500+ questions with Explanations

Java SE7 Associate Practice Exams

1Z0-803

With Exam Refresher

Hanumant Deshmukh

ENTHUWARE

OCAJP Oracle© Certified Associate Java SE 7 Programmer Practice Exams

(Exam Code 1Z0-803)

©Hanumant Deshmukh
www.enthuware.com

Preface

At Enthuware, we have been training Students for various Java certifications for the past ten years. Our highly advanced mock exam simulator is a well respected study tool in terms of quality, quantity, price, and features. While it is a full blown desktop application that offers unparalleled features, we realize that it may not be possible for everybody to study while being tied to a regular PC. We have received numerous requests to provide the same content in an eBook format. This book is an attempt to help people access the same mock exams on their mobile devices.

Table of Contents

Introduction

Oracle ® has significantly altered the Java Certification track with the release of Java SE 7. The entry level certification for Java Programmers is now broken up into two levels.

The first level is called Java Programmer - I, and confers an Associate Level certificate. The full name of this certification is *Oracle Certified **Associate**, Java SE 7 Programmer* (Exam Number: 1Z0-803).
The second level is called, well, Java Programmer - II, and confers a Professional Level certificate. The full name of this certification is *Oracle Certified **Professional**, Java SE 7 Programmer* (Exam Number: 1Z0-804). One must first acquire the Associate Level certificate before going for the Professional Level. This book focuses on the Level 1 certification exam, i.e. OCA - Java SE 7 Programmer Certification.

Who should use this book

This book is for OCA-JP SE7 certification aspirants. If you are a Java programmer with a couple of years of experience and if you are confident about your basic Java programming skills, you should take the mock exams in this book before attempting the real exam. The breadth of topics covered in this exam isn't much and the toughness level isn't too high. If you know basic Java programming, and are at least familiar with all the exam objectives, you will sail through the exam. You don't need any specific Certification Study Guide for this exam. The only caveat is that the exam is really lengthy. You have to answer 90 questions in 2.5 hours. If you haven't taken a lot of online tests recently, you need to practice. This is exactly what this book is for. It will make you ready for the real exam in a couple of weeks.

If you are a Java beginner, you should use this book as a supplement to which ever regular Java programming book you are going through. This book is not a tutorial or a guide and it is not meant to replace a regular Java book. It is not meant to teach you the basics of Java programming. You should use this book to check how well you are learning the concepts by answering questions for any given exam objective. For

example, if you are done studying the topic of Constructors from another book, you should attempt the questions given in this book on this topic and check how much you've learnt. If you have already gone through a book, you may attempt a complete mock exam and see how you score. If you pass the mock exam, you may proceed with the next mock exam otherwise, you need to go back to your regular Java programming book and read up on the topics on which you scored less.

How to prepare for OCA-JP

As mentioned before, if you are a Java programmer with a couple of years of experience, you don't need any specific Certification Study Guide. You can start with the mock exams in this book straight away.

If you are a complete beginner, you should first go through any Java book for beginners. Write a lot of short and simple programs to understand the concepts. Check out the exam objectives and read up on these topics from any book or online tutorial. We recommend Thinking in Java by **Bruce Eckel**. It is, in our opinion, the best book for learning Java. The first seven chapters are free and a must read for a Java beginner. Remember, the exam isn't tough in terms of the complexity of the questions. Once you are comfortable with all the topics, you may start with the first Standard Test.

A note about Old SCJA Resources

While the name of the new certification is similar to the name of the previous entry level certification called SCJA, which is short for "Sun Certified Java Associate for Java SE5/SE6", they have little in common. The exam objectives, style of questions, the toughness level of questions, number of questions, and time limit, are all different. So if you have a book for the old version of the certification, be careful not to get distracted by topics such as UML, EJB, JSP/Servlet, JDBC, HTML, and Swing.

How is this book organized?

This book contains full sized mock exams that mimic the style and toughness level of the real exam. All such mock exams are under Standard Tests.

Every question is also categorized under the exam objective that it covers. So if you are following a book and want to reinforce your understanding about a topic, you may attempt to answer questions on that particular topic.

Duplicate Questions
Standard Tests and Objective wise sets contain the same questions. So, depending on your mode of preparation, you should either attempt Standard Tests or Objective wise questions. If you have already attempted Objective wise questions, taking Standard Tests is of no use. You would have seen all the questions and your score will not be a real measure of your preparation.

Last Day Test, however, is an exception. We have made this test completely unique. Questions in this test are not included in Objective wise sets and so even if you have attempted all the objective wise questions, you may still attempt this test.

Taking the mock exams
You should start with the first Standard Test. Your score on this test will give you a fair idea of how well you are prepared for the exam. Ideally, you should score more than 77%* on this exam before moving on to the next exam. We have included a lot of reading material with the questions and you should go through the detailed explanation for each question...even for questions that you've answered correctly. Your objective should be to improve your score on the topics on which you scored less in this test.

If you fail in a standard test, you should not move on to the next test. Instead, first read up on the topics in which you failed from any book, write some sample programs to reinforce the concepts, and then attempt the next test.

*Oracle has recently changed the passing marks from 75% to 77% and it may change

the passing marks again at any time. So it is a good idea to check the current passing percentage at the time of your exam.

At the end of your preparation, you should attempt the "Last Day Test". If you pass this test, you are ready for the real exam. Most of our users have scored 10% higher on the real exam.

That is all there is to this book. Happy Learning!

-Hanumant Deshmukh and the rest of Enthuware Team.

P.S.1 If you have any doubt or feedback about any question, just click on the question id at the top of the question to see any discussion associated with that question on Enthuware discussion forum. If it hasn't been discussed before, feel free to post a message and we will try our best to help.

P.S.2 If you like this book, please do leave a feedback here. This will motivate us to create ebooks for other certifications as well.

Exam Objectives

The following are the exam objectives as of this writing. Oracle may tweak the objectives at any time so please verify the current objectives published at OCA-JP Certification Page at Oracle.

1. **Java Basics**
    - Define the scope of variables
    - Define the structure of a Java class
    - Create executable Java applications with a main method
    - Import other Java packages to make them accessible in your code
2. **Working With Java Data Types**
    - Declare and initialize variables
    - Differentiate between object reference variables and primitive variables
    - Read or write to object fields
    - Explain an object's lifecycle
    - Call methods on objects
    - Manipulate data using the StringBuilder class and its methods
    - Create and manipulate strings
3. **Using Operators and Decision Constructs**
    - Use Java operators
    - Use parentheses to override operator precedence
    - Test equality between strings and other objects using == and equals ()
    - Create if and if/else constructs
    - Use a switch statement
4. **Creating and Using Arrays**
    - Declare, instantiate, initialize and use a one-dimensional array
    - Declare, instantiate, initialize and use multi-dimensional array
    - Declare and use an ArrayList
5. **Using Loop Constructs**
    - Create and use while loops
    - Create and use for loops including the enhanced for loop
    - Create and use do/while loops

- Compare loop constructs
- Use break and continue

6. **Working with Methods and Encapsulation**
    - Create methods with arguments and return values
    - Apply the static keyword to methods and fields
    - Create an overloaded method
    - Differentiate between default and user-defined constructors
    - Create and overload constructors
    - Apply access modifiers
    - Apply encapsulation principles to a class
    - Determine the effect upon object references and primitive values when they are passed into methods that change the values
7. **Working with Inheritance**
    - Implement inheritance
    - Develop code that demonstrates the use of polymorphism
    - Differentiate between the type of a reference and the type of an object
    - Determine when casting is necessary
    - Use super and this to access objects and constructors
    - Use abstract classes and interfaces
8. **Handling Exceptions**
    - Differentiate among checked exceptions, RuntimeExceptions and Errors
    - Create a try-catch block and determine how exceptions alter normal program flow
    - Describe what exceptions are used for in Java
    - Invoke a method that throws an exception
    - Recognize common exception classes and categories

Taking the Actual Exam

The exam is conducted by Pearson VUE. You may pay for and schedule the exam online through their website.

Type of Questions

All the questions in the exam are multiple choice questions and every question tells you how many option you have to select. There are no drag and drop or fill in the blanks type questions.

Testing Software

The testing application is fairly straight forward. You can mark the questions, move forward and backward while answering the questions, change your answers, review questions, and finally submit the answers for evaluation. Before starting the test, it allows you to get acclimatized by presenting you with a test containing dummy questions. This dummy test does not eat up the time from the actual test so it is a good idea to use this feature and make yourself comfortable with the testing environment before starting the real test.

Overall, it is not something that you need to lose your sleep over. However, if you have never taken a computer based test before, we advise you to use our Mock Exam Simulator and simulate the real test on a computer at home. The simulator is meant for learning purpose and has a lot more features than the actual testing software, so it does not look exactly the same as the real test, but it will give you a decent idea of what to expect. Solving questions in a book without anybody keeping the time and taking a test on a computer are two different things. This test is particularly lengthy and it is easy to lose track of time. Using the simulator will help you in determining how quick or slow you are in answering the questions.

Finally, don't worry too much about the test. Practice all the questions in this book and you will be fine :)

Sample

This section contains only a few questions for sampling the book.

01. QID - 2.904

Given:

```
class Triangle{
    public int base;
    public int height;
    private static double ANGLE;

    public static double getAngle();

    public static void Main(String[] args) {
        System.out.println(getAngle());
    }
}
```

Identify the correct statements:

Select 1 option

A. It will not compile because it does not implement `setAngle` method.

B. It will not compile because `ANGLE` cannot be private.

C. It will not compile because `getAngle()` has no body.

D. It will not compile because `ANGLE` field is not initialized.

E. It will not compile because of the name of the method `Main` instead of `main`.

Check Answer

02. QID - 2.1184

Given:

```
StringBuilder b1 = new StringBuilder("snorkler");
StringBuilder b2 = new StringBuilder("yoodler");
```

Write the contents of b1 and b2 after the statements shown on the left are executed independent of each other.

Statements	Contents of b1	Contents of b2
`b1.append(b2.substring(2, 5) .toUpperCase());`		
`b2.insert(3, b1.append("a"));`		
`b1.replace(3, 4, b2.substring(4)) .append(b2.append(false));`		

Check Answer

03. QID - 2.1112

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}

class AnotherException extends Exception {}

public class ExceptionTest{
    public static void main(String[] args) throws Exception{
        try{
            m2();
        }
        finally{
            m3();
        }
        catch (NewException e){}
    }

    public static void m2() throws NewException { throw new NewExcep

    public static void m3() throws AnotherException{ throw new Anothe

}
```

Select 1 option

A. It will compile but will throw `AnotherException` when run.

B. It will compile but will throw `NewException` when run.

C. It will compile and run without throwing any exceptions.

D. It will not compile.

E. None of the above.

Check Answer

04. QID - 2.1116

Which of the following are true about the "default" constructor?

Select 2 options

A. It is provided by the compiler only if the class does not define any constructor.

B. It initializes the instance members of the class.

C. It calls the default 'no-args' constructor of the super class.

D. It initializes instance as well as class fields of the class.

E. It is provided by the compiler if the class does not define a 'no- args' constructor.

Check Answer

05. QID - 2.1258

Consider the following code:

```
class Test{
  public static void main(String[] args){
    for (int i = 0; i < args.length; i++)   System.out.print(i == 0
? args[i] : " " + args[i]);
  }
}
```

What will be the output when it is run using the following command:

java Test good bye friend!

Select 1 option

A. good bye friend!

B. good good good

C. goodgoodgood

D. good bye

E. None of the above.

Check Answer

06. QID - 2.962

Consider the following method...

```
public int setVar(int a, int b, float c) { ...}
```

Which of the following methods correctly overload the above method?

Select 2 options

A.
```
public int setVar(int a, float b, int c){
  return (int)(a + b + c);
}
```

B.
```
public int setVar(int a, float b, int c){
  return this(a, c, b);
}
```

C.
```
public int setVar(int x, int y, float z){
  return x+y;
}
```

D.
```
public float setVar(int a, int b, float c){
  return c*a;
}
```

E.
```
public float setVar(int a){
  return a;
}
```

Check Answer

07. QID - 2.1149

Given the following class definitions :

```
interface MyIface{};
class A {};
class B extends A implements MyIface{};
class C implements MyIface{};
```

and the following object instantiations:

```
A a = new A();
B b = new B();
C c = new C();
```

Which of the following assignments are legal at compile time?

Select 1 option

A. `b = c;`

B. `c = b;`

C. `MyIface i = c;`

D. `c = (C) b;`

E. `b = a;`

Check Answer

08. QID - 2.879

What will the following code print?

```
int[] scores1 = { 1, 2, 3, 4, 5, 6};
int[] scores2 = { 0, 0, 0, 0, 0, 0};
System.arraycopy(scores2, 2, scores1, 3, 2);
for(int i :  scores2) System.out.print(i);
```

Select 1 option

A. `123006`

B. `000000`

C. `000450`

D. It throw an exception at run time.

Check Answer

09. QID - 2.876

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){        this.side = length;    }

    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

    double getArea() {   return area;   }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Select 2 options

A. Make `setSide()` method private.

B. Make `getArea()` method private.

C. Make `side` and `area` fields private.

D. Make the side field private and remove the area field.

E. Change getArea method to:

```
public double getArea(){ return side*side; }
```

F. Add a `setArea()` method.

Check Answer

10. QID - 2.909

Which of the following declaration are valid:

```
1.  bool b = null;

2. boolean b = 1;

3. boolean b = true|false;

4 bool b = (10<11);

5. boolean b = true||false;
```

Select 1 option

A. 1 and 4

B. 2, 3, and 5

C. 2 and 3

D. 3 and 5

E. 5

Check Answer

11. QID - 2.1083

What will the following program print?

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

Select 1 option

A. 2

B. 3

C. 6

D. 7

E. 9

Check Answer

12. QID - 2.861

You want to find out whether two strings are equal or not, in terms of the actual characters within the strings. What is the best way to do this?

Select 1 option

A. use String's equals method.

B. use String's `equalsIgnoreCase` method.

C. Use == operator.

D. Use String's match method.

Check Answer

13. QID - 2.842

A new Java programmer has written the following method that takes an array of integers and sums up all the integers that are less than 100.

```
public void processArray(int[] values){
        int sum = 0;
        int i = 0;
        try{
            while(values[i]<100){
                sum = sum +values[i];
                i++;
            }
        }
        catch(Exception e){ }
        System.out.println("sum = "+sum);
    }
```

Which of the following are best practices to improve this code?

Select 2 options

A. Use `ArrayIndexOutOfBoundsException` for the catch argument.

B. Use `ArrayIndexOutOfBoundsException` for the catch argument and add code in the catch block to log or print the exception.

C. Add code in the catch block to handle the exception.

D. Use flow control to terminate the loop.

Check Answer

14. QID - 2.860

What will the following code print when run?

```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

Select 1 option

A. bat
big bat

B. big bat
none

C. big bat

D. bat

E. The code will not compile.

Check Answer

Sample Questions (Answered)

01. QID - 2.904 : Working with Methods

Given:

```
class Triangle{
    public int base;
    public int height;
    private static double ANGLE;

    public static double getAngle();

    public static void Main(String[] args) {
        System.out.println(getAngle());
    }
}
```

Identify the correct statements:

Correct Option is : C

~~**A.**~~ It will not compile because it does not implement `setAngle` method.

> There is no requirement that a class has to have a setter as well as a getter.

~~**B.**~~ It will not compile because `ANGLE` cannot be private.

> Any field can be made private.

C. It will not compile because `getAngle()` has no body.

~~**D.**~~ It will not compile because `ANGLE` field is not initialized.

> Since it is a static field, it will get a default value of 0.0.

~~**E.**~~ It will not compile because of the name of the method `Main` instead of `main`.

> A class can have a method named Main. Although, since it is not same as main, it will not be considered the standard main method that the JVM can invoke when the program is executed.

Back to Question without Answer

02. QID - 2.1184 : Working with Java Data Types - String, StringBuilder

Given:

```
StringBuilder b1 = new StringBuilder("snorkler");
StringBuilder b2 = new StringBuilder("yoodler");
```

Write the contents of b1 and b2 after the statements shown on the left are executed independent of each other.

Statements	Contents of b1	Contents of b2
`b1.append(b2.substring(2, 5) .toUpperCase());`	snorklerODL	yoodler
`b2.insert(3, b1.append("a"));`	snorklera	yoosnorkleradler
`b1.replace(3, 4, b2.substring(4)) .append(b2.append(false));`	snolerkleryoodlerfalse	yoodlerfalse

Explanation:

You need to understand how `append, insert, delete,` and `substring` methods of `StringBuilder/StringBuffer` work. Please go through JavaDoc API for these methods. This is very important for the exam. Observe that `substring()` does not modify the object it is invoked on but `append, insert` and `delete` do.

In the exam, you will find questions that use such quirky syntax, where multiple calls are chained together. For example: `sb.append("a").append("asdf").insert(2, "asdf")`. Make yourself familiar with this technique. If in doubt, just break it down into multiple calls. For example, the aforementioned statement can be thought of as:

```
sb.append("a");
sb.append("asdf");
sb.insert(2, "asdf")
```

Note that the method `substring()` in `StringBuilder/StringBuffer` returns a `String` (and not a reference to itself, unlike `append, insert,` and `delete`). So

another StringBuilder method cannot be chained to it. For example, the following is not valid: `sb.append("a").substring(0, 4).insert(2, "asdf");`

The following is valid though: `String str = sb.append("a").insert(2, "asdf").substring(0, 4);`

Back to Question without Answer

03. QID - 2.1112 : Handling Exceptions

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}

class AnotherException extends Exception {}

public class ExceptionTest{
    public static void main(String[] args) throws Exception{
        try{
            m2();
        }
        finally{
            m3();
        }
        catch (NewException e){}
    }

    public static void m2() throws NewException { throw new NewExcep

    public static void m3() throws AnotherException{ throw new Anothe

}
```

Correct Option is : D

~~**A.**~~ It will compile but will throw `AnotherException` when run.

~~**B.**~~ It will compile but will throw `NewException` when run.

~~**C.**~~ It will compile and run without throwing any exceptions.

D. It will not compile.

> Because a catch block cannot follow a finally block!

~~**E.**~~ None of the above.

Explanation:

Syntax of try/catch/finally is:

```
try{
}
catch(Exception1 e) {... }
catch(Exception2 e) {... }
...
catch(ExceptionN e) {... }
finally { ...  }
```

With a try, either a catch and or finally or both can occur.
A try MUST be followed by at least one catch or finally. (Unless it is a try with resources statement, which is not in scope for this exam.)

In Java 7, you can collapse the catch blocks into a single one:

```
try {
  ...
}
catch (SQLException | IOException | RuntimeException e) {
  //In this block, the class of the actual exception object will be
whatever exception is thrown at runtime.
  //But the class of the reference e will be the closest common
super class of all the exceptions in the catch block.
  //In this case, it will be java.lang.Exception because that is the
most specific class that is a super class for all the three
exceptions.
  e.printStackTrace();
}
```

Back to Question without Answer

04. QID - 2.1116 : Constructors

Which of the following are true about the "default" constructor?

Correct Options are : A C

A. It is provided by the compiler only if the class does not define any constructor.

~~**B.**~~ It initializes the instance members of the class.

C. It calls the default 'no-args' constructor of the super class.

~~**D.**~~ It initializes instance as well as class fields of the class.

~~**E.**~~ It is provided by the compiler if the class does not define a 'no- args' constructor.

It is not provided even if the class declares any other constructor.

Explanation:

The default constructor is provided by the compiler only when a class does not define ANY constructor explicitly. For example,

```
public class A{
  public A()  //This constructor is automatically inserted by the cor
    super();  //Note that it calls the super class' default no-args c
  }
}
public class A{
  //Compiler will not generate any constructor because the programme
  public A(int i){
     //do something
```

```
    }
}
```

Back to Question without Answer

05. QID - 2.1258 : Java Basics

Consider the following code:

```
class Test{
  public static void main(String[] args){
    for (int i = 0; i < args.length; i++)   System.out.print(i == 0
? args[i] : " " + args[i]);
  }
}
```

What will be the output when it is run using the following command:

java Test good bye friend!

Correct Option is : A

A. good bye friend!

~~**B.**~~ good good good

~~**C.**~~ goodgoodgood

~~**D.**~~ good bye

~~**E.**~~ None of the above.

Explanation:

The arguments passed on the command line can be accessed using the args array. The first argument (i.e. good) is stored in args[0], second argument (i.e. bye) is stored in

args[1] and so on.

Here, we are passing 3 arguments. Therefore, args.length is 3 and the for loop will run 3 times. For the first iteration, i is 0 and so the first operand of the ternary operator (?) will be returned, which is args[i]. For the next two iterations, " "+args[i] will be returned. Hence, the program will print three strings: "good", " bye", and " friend" on the same line.

Notice that unlike in C++, program name is not the first parameter in the argument list. Java does not need to know the program name because the .class file name and the java class name are always same (for a public class). So the java code always knows the program name it is running in. So there is no need to pass the program name as the first parameter of the argument list.
Note that in C/C++, the binary file name may be anything so the code does not know what binary file it is going to end up in. That's why the program name is also sent (automatically) in parameter list.

Back to Question without Answer

06. QID - 2.962 : Overloading methods

Consider the following method...

```
public int setVar(int a, int b, float c) { ...}
```

Which of the following methods correctly overload the above method?

Correct Options are : A E

A.
```
public int setVar(int a, float b, int c){
  return (int)(a + b + c);
}
```

~~**B.**~~
```
public int setVar(int a, float b, int c){
  return this(a, c, b);
}
```

this(...) can only be called in a constructor and that too as a first statement.

~~**C.**~~
```
public int setVar(int x, int y, float z){
  return x+y;
}
```

It will not compile because it is same as the original method. The name of parameters do not matter.

~~**D.**~~
```
public float setVar(int a, int b, float c){
  return c*a;
}
```

It will not compile as it is same as the original method. The return type does not matter.

E.
```
public float setVar(int a){
  return a;
}
```

Explanation:

A method is said to be overloaded when the other method's name is same and parameters (either the number or their order) are different.
Option 2 is not valid Because of the line: `return this(a, c, b);` This is the syntax of calling a constructor and not a method. It should have been: `return this.setVar(a, c, b);`

Back to Question without Answer

07. QID - 2.1149 : Working with Inheritance

Given the following class definitions :

```
interface MyIface{};
class A {};
class B extends A implements MyIface{};
class C implements MyIface{};
```

and the following object instantiations:

```
A a = new A();
B b = new B();
C c = new C();
```

Which of the following assignments are legal at compile time?

Correct Option is : C

~~**A.**~~ `b = c;`

There is no relation between b and c.

~~**B.**~~ `c = b;`

There is no relation between b and c.

C. `MyIface i = c;`

Because C implements I.

~~**D.**~~ `c = (C) b;`

> Compiler can see that in no case can an object referred to by b can be of class c. So it is a compile time error.

~~**E.**~~ `b = a;`

> It will fail at compile time because a is of class A and can potentially refer to an object of class A, which cannot be assigned to b, which is a variable of class B. To make it compile, you have to put an explicit cast, which assures the compiler that a will point to an object of class B (or a subclass of B) at run time. Note that, in this case, an explicit cast can take it through the compiler but it will then fail at run time because a does not actually refer to an object of class B (or a subclass of B), so the JVM will throw a ClassCastException.

Explanation:

The statements `c = b` and `b = c` are illegal, since neither of the classes C and B is a subclass of the other. Even though a cast is provided, the statement `c = (C) b` is illegal because the object referred to by b cannot ever be of type C.

Back to Question without Answer

08. QID - 2.879 : Creating and Using Arrays

What will the following code print?

```
int[] scores1 = { 1, 2, 3, 4, 5, 6};
int[] scores2 = { 0, 0, 0, 0, 0, 0};
System.arraycopy(scores2, 2, scores1, 3, 2);
for(int i :  scores2) System.out.print(i);
```

Correct Option is : B

~~**A.**~~ `123006`

B. `000006`

> Source is `scores2` and destination is `scores1`. So `scores1` will become `1 2 3 0 0 6`. However, you are printing `scores2`, which is still `{0, 0, 0, 0, 0, 0}`.

~~**C.**~~ `000450`

~~**D.**~~ It throw an exception at run time.

Explanation:

The arraycopy method basically copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. The last parameter is the number of elements that you want to copy.

There are questions in the exam on `System.arraycopy` so you should go through the following JavaDoc description for this method:

```
public static void arraycopy(Object src,
             int srcPos,
             Object dest,
             int destPos,
             int length)
```

Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. A subsequence of array components is copied from the source array referenced by src to the destination array referenced by dest. The number of components copied is equal to the length argument. The components at positions srcPos through srcPos+length-1 in the source array are copied into positions destPos through destPos+length-1, respectively, of the destination array. If the src and dest arguments refer to the same array object, then the copying is performed as if the components at positions srcPos through srcPos+length-1 were first copied to a temporary array with length components and then the contents of the temporary array were copied into positions destPos through destPos+length-1 of the destination array.

If dest is null, then a `NullPointerException` is thrown.

If src is null, then a `NullPointerException` is thrown and the destination array is not modified.

Otherwise, if any of the following is true, an ArrayStoreException is thrown and the destination is not modified:

The src argument refers to an object that is not an array.
The dest argument refers to an object that is not an array.
The src argument and dest argument refer to arrays whose component types are different primitive types.
The src argument refers to an array with a primitive component type and the dest argument refers to an array with a reference component type.
The src argument refers to an array with a reference component type and the dest

argument refers to an array with a primitive component type.
Otherwise, if any of the following is true, an IndexOutOfBoundsException is thrown and the destination is not modified:

The srcPos argument is negative.
The destPos argument is negative.
The length argument is negative.
srcPos+length is greater than src.length, the length of the source array.
destPos+length is greater than dest.length, the length of the destination array.
Otherwise, if any actual component of the source array from position srcPos through srcPos+length-1 cannot be converted to the component type of the destination array by assignment conversion, an ArrayStoreException is thrown. In this case, let k be the smallest nonnegative integer less than length such that src[srcPos+k] cannot be converted to the component type of the destination array; when the exception is thrown, source array components from positions srcPos through srcPos+k-1 will already have been copied to destination array positions destPos through destPos+k-1 and no other positions of the destination array will have been modified. (Because of the restrictions already itemized, this paragraph effectively applies only to the situation where both arrays have component types that are reference types.)

Parameters:
src - the source array.
srcPos - starting position in the source array.
dest - the destination array.
destPos - starting position in the destination data.
length - the number of array elements to be copied.
Throws:
IndexOutOfBoundsException - if copying would cause access of data outside array bounds.
ArrayStoreException - if an element in the src array could not be stored into the dest array because of a type mismatch.
NullPointerException - if either src or dest is null.

Back to Question without Answer

09. QID - 2.876 : Encapsulation

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){        this.side = length;    }

    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

    double getArea() {   return area;   }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Correct Options are : D E

~~**A.**~~ Make `setSide()` method private.

~~**B.**~~ Make `getArea()` method private.

It should be made public so that other classes can get the area.

~~**C.**~~ Make `side` and `area` fields private.

> There is no need to keep the area field because that would amount to duplicating the data. If you change side, the value of area will become obsolete.

D. Make the side field private and remove the area field.

E. Change getArea method to:

```
public double getArea(){ return side*side; }
```

~~**F.**~~ Add a `setArea()` method.

> This is not required because area is calculated using the side. So if you allow other classes to set the area, it could make side and area inconsistent with each other.

Explanation:

There can be multiple ways to accomplish this. The exam asks you questions on the similar pattern.
The key is that your data variable should be private and the functionality that is to be exposed outside should be public. Further, your setter methods should be coded such that they don't leave the data members inconsistent with each other.

Back to Question without Answer

10. QID - 2.909 : Working with Java Data Types - Variables and Objects

Which of the following declaration are valid:

```
1.  bool b = null;

2. boolean b = 1;

3. boolean b = true|false;

4 bool b = (10<11);

5. boolean b = true||false;
```

Correct Option is : D

~~**A.** 1 and 4~~

~~**B.** 2, 3, and 5~~

~~**C.** 2 and 3~~

D. 3 and 5

~~**E.** 5~~

Explanation:

`bool` is an invalid keyword. Therefore, 1 and 4 can't be right. (Although 1 could be right if `bool` were a user defined class but as per Java coding conventions, a class name should start with a capital letter.)

`boolean b = 1;` is wrong because you can only assign `true` or `false` to a `boolean` variable. 1 is an integral value it cannot be converted to `boolean`. Also note that `boolean b = null;` would be invalid as well because `null` is not a true or false value. A primitive (whether it is a boolean or an int or a double), can never be assigned `null`.

`boolean b = true|false;` and `boolean b = true||false;` are both valid and the difference between true|false and true||false is not material in this case. However, there is a lot of difference between | (and &) and || (and &&) as explained below:

|| and && perform short circuit evaluation, while & and | do not. Which means, if you use the || and && forms, Java will not bother to evaluate the right-hand operand if the result of the expression can be known by just evaluating the left hand operand.

Consider the following example.

```
Boolean b = true;
if(b || foo.timeConsumingCall()) {
  //entered here without calling timeConsumingCall()
}
```

Another example:

```
String s = null;
if(s != null && s.isEmpty())  //No NullPointerException because
string.isEmpty() is not called.
//If you use & instead of && , s.isEmpty will be called and a
NullPointerException will be thrown.{
   ...
}
```

Back to Question without Answer

11. QID - 2.1083 : Using Loop Constructs

What will the following program print?

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

Correct Option is : B

~~**A.** 2~~

B. 3

~~**C.** 6~~

~~**D.** 7~~

~~E.~~ 9

Explanation:

To understand how this loop works let us put some extra print statements in the innermost loop:

```
System.out.println("i="+i+" j="+j+" k="+k);
if(k-j>0){
     System.out.println("breaking middle "+j);
     break middle;
}
counter++;
```

This is what it prints:

```
i=0 j=0 k=0
i=0 j=0 k=1
breaking middle 0
i=1 j=0 k=0
i=1 j=0 k=1
breaking middle 0
i=2 j=0 k=0
i=2 j=0 k=1
breaking middle 0
3
```

The key is that the middle loop is broken as soon as k-j becomes > 0. This happens on every second iteration of inner loop when k is 1 and j is 0. Now, when middle is broken inner cannot continue. So the next iteration of outer starts.

Back to Question without Answer

12. QID - 2.861 : Working with Java Data Types - String, StringBuilder

You want to find out whether two strings are equal or not, in terms of the actual characters within the strings. What is the best way to do this?

Correct Option is : A

A. use String's equals method.

For example:

```
String x1 = "a";
String x2 = new String("a");
```

`x1.equals(x2)` will return true. Because even though x1 and x2 are pointing to different objects, the content of the objects are same, which is what String's equals method checks.

`x1 == x2` will return false, because == only checks if the two references are pointing to the same object or not. In this case, they are not.

~~**B.**~~ use String's `equalsIgnoreCase` method.

If you use this method, `"a"` will be considered equal to `"A"`, which is not what the question is asking for.

~~**C.**~~ Use == operator.

== checks for the equality of the references and not for the equality of the objects themselves. Therefore, this will return true only if two string references are pointing to the same String object, which is not what the question is asking for.

~~**D.**~~ Use String's match method.

There is no method named match in String class.
There is a matches method, which checks whether the String matches a regular expression but that is beyond the scope of this exam.

```
public boolean matches(String regex)
```

Tells whether or not this string matches the given regular expression.
An invocation of this method of the form str.matches(regex) yields exactly the same result as the expression Pattern.matches(regex, str)

Back to Question without Answer

13. QID - 2.842 : Handling Exceptions

A new Java programmer has written the following method that takes an array of integers and sums up all the integers that are less than 100.

```
public void processArray(int[] values){
        int sum = 0;
        int i = 0;
        try{
            while(values[i]<100){
                sum = sum +values[i];
                i++;
            }
        }
        catch(Exception e){ }
        System.out.println("sum = "+sum);
    }
```

Which of the following are best practices to improve this code?

Correct Options are : B D

~~**A.**~~ Use `ArrayIndexOutOfBoundsException` for the catch argument.

B. Use `ArrayIndexOutOfBoundsException` for the catch argument and add code in the catch block to log or print the exception.

> Empty catch blocks are a bad practice because at run time, if the exception is thrown, the program will not show any sign of the exception and may produce bad results that will be hard to debug. Therefore, it is a good practice to at least print out the exception if you don't want to do any thing upon encountering an exception.

~~**C.**~~ Add code in the catch block to handle the exception.

There are a few questions in the exam that are difficult to interpret. In this case, for example, it is not clear what is meant by handling the exception. The catch block itself is meant to handle the exception. Once you get the exception, you can do what ever is required in the catch block.

D. Use flow control to terminate the loop.

It is considered a bad practice to use exceptions to control the flow of execution. In this case, `values[i]` will throw an `ArrayIndexOutOfBoundsException` once it goes beyond the array length and the programmer is using this fact to control the loop. Instead of doing this, the programmer should use something like: `for(int i=0; i<values.length; i++)` to control the execution of the loop.

Back to Question without Answer

14. QID - 2.860 : Using Operators and Decision Constructs

What will the following code print when run?

```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

Correct Option is : B

~~**A.** bat~~
big bat

B. big bat
none

Since there is a case condition that matches the input string "B", that case statement will be executed directly. This prints "big bat". Since there is no break after this case statement and the next case statement, the control will fall through the next one (which is default :) and so "none" will be printed as well.

Note that "b" and "B" are different strings. "B" is not equal to "b".

~~C.~~ big bat

~~D.~~ bat

~~E.~~ The code will not compile.

Explanation:

As of JDK 7 release, you can use a String object in the expression of a switch statement:

```
public String getTypeOfDayWithSwitchStatement(String dayOfWeekArg) {
     String typeOfDay;
     switch (dayOfWeekArg) {
         case "Monday":
             typeOfDay = "Start of work week";
             break;
         case "Tuesday":
         case "Wednesday":
         case "Thursday":
             typeOfDay = "Midweek";
             break;
         case "Friday":
             typeOfDay = "End of work week";
             break;
         case "Saturday":
         case "Sunday":
             typeOfDay = "Weekend";
             break;
         default:
             throw new IllegalArgumentException("Invalid day of the
     }
     return typeOfDay;
}
```

The switch statement compares the String object in its expression with the expressions

associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

Back to Question without Answer

Standard Tests

This test environment mimics the real exam environment. You should take the tests in this environment if you are fully prepared for the final exam. You should be able to score more than 77% on these tests. If you score less on a topic, you should study it again before taking the next test.

DO NOT ATTEMPT NEXT STANDARD TEST BEFORE GOING THROUGH A BOOK IF YOU SCORE LESS THAN 77%.

Test 1

01. QID - 2.1177

Which of the following statements are true?

Select 1 option

A. For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `true`.

B. For any non-null reference `o1`, the expression `(o1 instanceof Object)` will always yield `true`.

C. For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `false`.

D. For any non-null reference `o1`, the expression `(o1 instanceof Object)` may yield `false`.

E. None of the above.

Check Answer

02. QID - 2.1320

What will the following code print when compiled and run?

```
class ABCD{
   int x = 10;
   static int y = 20;
}
class MNOP extends ABCD{
   int x = 30;
   static int y = 40;
}

public class TestClass {
   public static void main(String[] args) {
     System.out.println(new MNOP().x+", "+new MNOP().y);
   }
}
```

Select 1 option

A. 10, 40

B. 30, 20

C. 10, 20

D. 30, 40

E. 20, 30

F. Compilation error.

Check Answer

03. QID - 2.1229

Which of these array declarations and instantiations are legal?

Select 4 options

A. `int[ ] a[ ] = new int [5][4] ;`

B. `int a[ ][ ] = new int [5][4] ;`

C. `int a[ ][ ] = new int [ ][4] ;`

D. `int[ ] a[ ] = new int[4][ ] ;`

E. `int[ ][ ] a = new int[5][4] ;`

Check Answer

04. QID - 2.1232

Which of the following lines can be inserted at line 1 to make the program run?

```
//line 1
public class TestClass{
  public static void main(String[] args){
     PrintWriter pw = new PrintWriter(System.out);
     OutputStreamWriter osw  =  new OutputStreamWriter( System.out
);
     pw.print("hello");
  }
}
```

Assume that `PrintWriter` and `OutputStreamWriter` are valid classes in `java.io` package.

Select 1 option

A. `import java.lang.*;`

B. `import java.io.*;`

C. `import java.io.OutputStreamWriter;`

D. `include java.io.*;`

E. `include java.lang.System;`

Check Answer

05. QID - 2.1044

What will be the output if you run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0 ; j < 1 ; ++j , i++){
         System.out.println( i + " " + j );
      }
      System.out.println( i + " " + j );
   }
}
```

Select 1 option

A. `0 0` will be printed twice.

B. `1 1` will be printed once.

C. `0 1` will be printed followed by `1 2`.

D. `0 0` will be printed followed by `1 1`.

E. It will print `0 0` and then `0 1`.

Check Answer

06. QID - 2.1039

What will the following method return if called with an argument of 7?

```
public int transformNumber(int n){
   int radix = 2;
   int output = 0;
   output += radix*n;
   radix = output/radix;
   if(output<14){
       return output;
   }
   else{
       output = output*radix/2;
       return output;
   }
   else {
       return output/2;
   }
}
```

Select 1 option

A. 7

B. 14

C. 49

D. Compilation fails.

Check Answer

07. QID - 2.1347

Consider the following code...

```
public class TestClass{
     class MyException extends Exception {}
     public void myMethod() throws XXXX{
         throw new MyException();
     }
}
```

What can replace XXXX?

Select 3 options

A. MyException

B. Exception

C. No throws clause is necessary

D. Throwable

E. RuntimeException

Check Answer

08. QID - 2.1225

Which of these statements concerning the `charAt()` method of the `String` class are true?

Select 2 options

A. The `charAt( )` method can take a char value as an argument.

B. The `charAt( )` method returns a Character object.

C. The expression `char ch = "12345".charAt(3)` will assign 3 to ch.

D. The expression `char ch = str.charAt(str.length())` where str is "12345", will assign 3 to ch.

E. The index of the first character is 0.

F. It throws `StringIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

G. It throws `ArrayIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

Check Answer

09. QID - 2.934

You can call only public and protected constructors of the super class from a subclass if the subclass is not in the same package because only those are inherited.

Select 1 option

A. True

B. False

Check Answer

10. QID - 2.996

Consider the following program:

```
class Game {
  public void play() throws Exception   {
    System.out.println("Playing...");
  }
}

class Soccer extends Game {
   public void play(String ball)    {
      System.out.println("Playing Soccer with "+ball);
   }
}

public class TestClass {
   public static void main(String[] args) throws Exception  {
       Game g = new Soccer();
       // 1
       Soccer s = (Soccer) g;
       // 2
   }
}
```

Which of the given options can be inserted at //1 and //2?

Select 2 options

A. It will not compile as it is.

B. It will throw an `Exception` at runtime if it is run as it is.

C. `g.play();` at //1 and `s.play("cosco");` at //2

D. `g.play();` at //1 and `s.play();` at //2

E. `g.play("cosco");` at //1 and `s.play("cosco");` at //2

Check Answer

11. QID - 2.1021

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      String s = "hello";
      StringBuilder sb = new StringBuilder( "hello" );
      sb.reverse();
      s.reverse();
      if( s == sb.toString() )  System.out.println( "Equal" );
      else System.out.println( "Not Equal" );
   }
}
```

Select 1 option

A. Compilation error.

B. It will print 'Equal'.

C. It will print 'Not Equal'.

D. Runtime error.

E. None of the above.

Check Answer

12. QID - 2.915

Given the following contents of two java source files:

```
package util.log4j;
public class Logger  {
  public void log(String msg){
      System.out.println(msg);
  }
}
```

and

```
package util;
public class TestClass {
    public static void main(String[] args) throws Exception {
        Logger logger = new Logger();
        logger.log("hello");
    }
}
```

What changes, when made independently, will enable the code to compile and run?

Select 2 options

A. Replace `Logger logger = new Logger();` with:

```
log4j.Logger logger = new log4j.Logger();
```

B. Replace package `util.log4j;` with

```
package util;
```

C. Replace `Logger logger = new Logger();` with:

```
util.log4j.Logger logger = new util.log4j.Logger();
```

D. Remove package `util.log4j;` from Logger.

E. Add `import log4j;` to TestClass.

Check Answer

13. QID - 2.1333

Which of the following are not legal Java identifiers?

Select 1 option

A. goto

B. unsigned

C. String

D. _xyz

E. $_abc

F. iLikeVeryVeryVeryVeryVeryLongIdentifiersThatDontMakeAnySenseAtAll (65 characters)

Check Answer

14. QID - 2.1088

Consider:
`o1` and `o2` denote two object references to two different objects of same class.
Which of the following statements are true?

Select 2 options

A. `o1.equals(o2)` will always be false.

B. `o1.hashCode() == o2.hashCode()` will always be false.

C. `o1 == o2` will always be false.

D. Nothing can be said about `o1.equals(o2)` regarding what it will return based on the given information.

E. Nothing can be said about `o1 == o2`.

Check Answer

15. QID - 2.924

Consider the following class...

```
class TestClass{
    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(Object x) { System.out.println("In Object"); } //3

    void probe(Long x) { System.out.println("In Long"); } //4

    public static void main(String[] args){
        String a = "hello";
        new TestClass().probe(a);
    }
}
```

What will be printed?

Select 1 option

A. In Integer

B. In Object

C. In Long

D. It will not compile

Check Answer

16. QID - 2.1343

Consider the following class...

```
class TestClass{
   int x;
   public static void main(String[] args){
      // lot of code.
   }
}
```

Select 1 option

A. By declaring x as static, main can access `this.x`

B. By declaring x as public, main can access `this.x`

C. By declaring x as protected, main can access `this.x`

D. main cannot access `this.x` as it is declared now.

E. By declaring x as private, main can access `this.x`

Check Answer

17. QID - 2.1246

What will the following statement return?

```
"    hello java guru   ".trim();
```

Select 1 option

A. The line of code will not compile.

B. `"hellojavaguru"`

C. `"hello java guru"`

D. `"hello java guru   "`

E. None of the above

Check Answer

18. QID - 2.851

Identify the correct statements about ArrayList?

Select 3 options

A. Standard JDK provides no subclasses of ArrayList.

B. You cannot store primitives in an ArrayList.

C. It allows constant time access to all its elements.

D. ArrayList cannot resize dynamically if you add more number of elements than its capacity.

E. An ArrayList is backed by an array.

Check Answer

19. QID - 2.1013

Using a break in a while loop causes the loop to break the current iteration and start the next iteration of the loop.

Select 1 option

A. True

B. False

Check Answer

20. QID - 2.1193

Which of these statements are valid when occurring by themselves?

Select 3 options

A. `while ( ) break ;`

B. `do { break ; } while (true) ;`

C. `if (true) { break ; }` (When not inside a switch block or a loop)

D. `switch (1) { default : break; }`

E. `for ( ; true ; ) break ;`

Check Answer

21. QID - 2.1136

Which of the given statements are correct about the following code?

```
//Filename: TestClass.java
class TestClass{
   public static void main(String[] args){
      A a = new A();
      B b = new B();
   };
}
class A implements T1, T2{}
class B extends A implements T1{}
interface T1 { }
interface T2 { }
```

Select 4 options

A. `(a instanceof T1)` will return true.

B. `(a instanceof T2)` will return true.

C. `(b instanceof T1)` will return true.

D. `(b instanceof T2)` will return true.

E. `(b instanceof A)` will return false.

Check Answer

22. QID - 2.1368

Identify the exceptions that are usually thrown by the JVM and the exceptions usually thrown by an application.

Select 1 option

A. JVM : IllegalStateException, IllegalArgumentException
Application : ClassCastException, NullPointerException, SecurityException

B. JVM : IllegalStateException, IllegalArgumentException, ClassCastException,
Application : NullPointerException, SecurityException

C. JVM : IllegalStateException, IllegalArgumentException, ClassCastException, NullPointerException
Application : SecurityException

D. JVM : ClassCastException, NullPointerException, SecurityException
Application : IllegalStateException, IllegalArgumentException

E. JVM : ClassCastException, NullPointerException
Application : IllegalStateException, IllegalArgumentException, SecurityException

F. JVM : ClassCastException, NullPointerException, IllegalStateException
Application : IllegalArgumentException, SecurityException

Check Answer

23. QID - 2.1173

Consider the following class :

```
public class Test{
   public static void main(String[] args){
      if (args[0].equals("open"))
         if (args[1].equals("someone"))
            System.out.println("Hello!");
      else System.out.println("Go away "+ args[1]);
    }
}
```

Which of the following statements are true if the above program is run with the command line :
java Test closed

Select 1 option

A. It will throw `ArrayIndexOutOfBoundsException` at runtime.

B. It will end without exceptions and will print nothing.

C. It will print `Go away`

D. It will print `Go away` and then will throw `ArrayIndexOutOfBoundsException`.

E. None of the above.

Check Answer

24. QID - 2.1064

Consider this code:

```
interface X1{ }
interface X2{ }
class A { }
class B extends A implements X1{ }
class C extends B implements X2{
   D d = new D();
}
class D { }
```

Which of the following statements are true?

Select 3 options

A. D is-a B.

B. B has-a D.

C. C is-a A

D. C is-a X1

E. C is-a X2

Check Answer

25. QID - 2.907

Consider the following code appearing in Eagle.java

```
class Bird {
    private Bird(){     }
}
class Eagle extends Bird {
    public String name;
    public Eagle(String name){
        this.name = name;
    }

    public static void main(String[] args) {
        System.out.println(new Eagle("Bald Eagle").name);
    }
}
```

What needs to be done to make this code compile?

Select 1 option

A. Nothing, it will compile as it is.

B. Make Eagle class declaration public:

```
public class Eagle { ... }
```

C. Make the Eagle constructor private:

```
private Eagle(String name){ ... }
```

D. Make Bird constructor public:

```
public Bird() { ... }
```

E. Insert super(); as the first line in Eagle constructor:

```
public Eagle(String name){
    super();
    this.name = name;
}
```

Check Answer

26. QID - 2.1256

Which of these statements are true?

Select 2 options

A. A `super( )` or `this( )` call must always be provided explicitly as the first statement in the body of the constructor.

B. If a subclass does not have any declared constructors, the implicit default constructor of the subclass will have a call to `super( )`.

C. If neither `super( )` or `this( )` is declared as the first statement of the body of a constructor, then `this( )` will implicitly be inserted as the first statement.

D. `super(...)` can only be called in the first line of the constructor but `this(...)` can be called from anywhere.

E. You can either call `super(...)` or `this(...)` but not both.

Check Answer

27. QID - 2.1112

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}

class AnotherException extends Exception {}

public class ExceptionTest{
    public static void main(String[] args) throws Exception{
        try{
            m2();
        }
        finally{
            m3();
        }
        catch (NewException e){}
    }

    public static void m2() throws NewException { throw new NewExcep

    public static void m3() throws AnotherException{ throw new Anothe

}
```

Select 1 option

A. It will compile but will throw `AnotherException` when run.

B. It will compile but will throw `NewException` when run.

C. It will compile and run without throwing any exceptions.

D. It will not compile.

E. None of the above.

Check Answer

28. QID - 2.939

Which of the following are true regarding overloading of a method?

Select 1 option

A. An overloading method must have a different parameter list and same return type as that of the overloaded method.

B. If there is another method with the same name but with a different number of arguments in a class then that method can be called as overloaded.

C. If there is another method with the same name and same number and type of arguments but with a different return type in a class then that method can be called as overloaded.

D. An overloaded method means a method with the same name and same number and type of arguments exists in the super class and sub class.

Check Answer

29. QID - 2.1328

Consider the following classes :

```
interface I{
}
class A implements I{
}

class B extends A {
}

class C extends B{
}
```

And the following declarations:

```
A a = new A();
B b = new B();
```

Identify options that will compile and run without error.

Select 1 option

A. `a = (B)(I)b;`

B. `b = (B)(I) a;`

C. `a = (I) b;`

D. `I i = (C) a;`

Check Answer

30. QID - 2.891

What can you do to make the following code compile?

```
public class TestClass {
    public static void main(String[] args) {
        int[] values = { 10, 20, 30 };
        for( /* put code here */ ){
        }
    }
}
```

Select 2 options

A. int k : values

B. int k in values

C. int k; k<0; k++

D. ;;

E. ; k<values.length;k++

Check Answer

31. QID - 2.942

Consider the following code to count objects and save the most recent object ...

```
int i = 0 ;
Object prevObject ;
public void saveObject(List e ){
    prevObject = e ;
    i++ ;
}
```

Which of the following calls will work without throwing an exception?

Select 3 options

A. `saveObject( new ArrayList() );`

B. `Collection c = new ArrayList(); saveObject( c );`

C. `List l = new ArrayList(); saveObject(l);`

D. `saveObject(null);`

E. `saveObject(0);` //The argument is the number zero and not the letter o

Check Answer

32. QID - 2.1189

Which of the following statements are acceptable?

Select 4 options

A. `Object o = new java.io.File("a.txt");`
(Assume that java.io.File is a valid class.)

B. `Boolean bool = false;`

C. `char ch = 10;`

D. `Thread t = new Runnable();`
(Assume that Runnable is a valid interface.)

E. `Runnable r = new Thread();`
(Assume that Thread is a class that implements Runnable interface)

Check Answer

33. QID - 2.940

Which of the statements regarding the following code are correct?

```
public class TestClass{
   static int a;
   int b;
   public TestClass(){
      int c;
      c = a;
      a++;
      b += c;
   }
   public static void main(String args[]) {   new TestClass();   }
}
```

Select 1 option

A. The code will fail to compile because the constructor is trying to access static members.

B. The code will fail to compile because the constructor is trying to use static member variable a before it has been initialized.

C. The code will fail to compile because the constructor is trying to use member variable b before it has been initialized.

D. The code will fail to compile because the constructor is trying to use local variable c before it has been initialized.

E. The code will compile and run without any problem.

Check Answer

34. QID - 2.1073

Which line(s) of code in the following program will cause a compilation error?

```
public class TestClass{
   static int value = 0; //1
   public static void main(String args[]) //2
   {
      int 2ndArgument = Integer.parseInt(args[2]); //3
      if( true == 2 > 10 ) //4
      {
         value = -10;
      }
      else{
         value =  2ndArgument;
      }
      for( ; value>0; value--) System.out.println("A"); //5
   }
}
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

35. QID - 2.1001

Consider the following class:

```
public class Test{
    public int id;
}
```

Which of the following is the correct way to make the variable 'id' read only for any other class?

Select 1 option

A. Make 'id' private.

B. Make 'id' private and provide a public method getId() which will return its value.

C. Make 'id' static and provide a public static method getId() which will return its value.

D. Make id 'protected'

Check Answer

36. QID - 2.1164

Which of the following statements will compile without any error?

Select 4 options

A. `System.out.println("a"+'b'+63);`

B. `System.out.println("a"+63);`

C. `System.out.println('b'+new Integer(63));`

D. `String s = 'b'+63+"a";`

E. `String s = 63 + new Integer(10);`

Check Answer

37. QID - 2.1348

Which digits and in what order will be printed when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int k = 0;
      try{
         int i = 5/k;
      }
      catch (ArithmeticException e){
         System.out.println("1");
      }
      catch (RuntimeException e){
         System.out.println("2");
         return ;
      }
      catch (Exception e){
         System.out.println("3");
      }
      finally{
         System.out.println("4");
      }
      System.out.println("5");
   }
}
```

Select 1 option

A. The program will print 5.

B. The program will print 1 and 4, in that order.

C. The program will print 1, 2 and 4, in that order.

D. The program will print 1, 4 and 5, in that order.

E. The program will print 1,2, 4 and 5, in that order.

Check Answer

38. QID - 2.1174

What will the following program print?

```
public class TestClass{
  static String str = "Hello World";
  public static void changeIt(String s){
    s = "Good bye world";
  }
  public static void main(String[] args){
    changeIt(str);
    System.out.println(str);
  }
}
```

Select 1 option

A. `"Hello World"`

B. `"Good bye world"`

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

39. QID - 2.1061

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i, j, k;
      i = j = k = 9;
      System.out.println(i);
   }
}
```

Select 2 options

A. The code will not compile because unlike in c++, operator '=' cannot be chained i.e. a = b = c = d is invalid.

B. The code will not compile as 'j' is being used before getting initialized.

C. The code will compile correctly and will display '9' when run.

D. The code will not compile as 'j' and 'i' are being used before getting initialized.

E. All the variables will get a value of 9.

Check Answer

40. QID - 2.983

Consider the following classes:

```
class A implements Runnable{ ...}
class B extends A implements Observer { ...}
```

(Assume that Observer has no relation to Runnable.)

and the declarations :

```
A a = new A() ;
B b = new B();
```

Which of the following Java code fragments will compile and execute without throwing exceptions?

Select 2 options

A. `Object o = a; Runnable r = o;`

B. `Object o = a; Runnable r = (Runnable) o;`

C. `Object o = a; Observer ob = (Observer) o ;`

D. `Object o = b; Observer o2 = o;`

E. `Object o = b; Runnable r = (Runnable) b;`

Check Answer

41. QID - 2.1238

What will the following code print when run without any arguments ...

```
public class TestClass {

    public static int m1(int i){
        return ++i;
    }

    public static void main(String[] args) {

        int k = m1(args.length);
        k += 3 + ++k;
        System.out.println(k);
    }

}
```

Select 1 option

A. It will throw `ArrayIndexOutofBoundsException`.

B. It will throw `NullPointerException`.

C. `6`

D. `5`

E. `7`

F. 2

G. None of these.

Check Answer

42. QID - 2.1084

Which of the following are valid operators in Java?

Select 4 options

A. !

B. ~

C. &

D. %=

E. $

Check Answer

43. QID - 2.1310

Consider the following method...

```
public static void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   if (flag)    //3
   System.out.println("False True");
   else         //4
   System.out.println("True False");
   else         //5
   System.out.println("True True");
   else         //6
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Select 2 options

A. If run with an argument of 'false', it will print 'False False'

B. If run with an argument of 'false', it will print 'True True'

C. If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

E. It will not compile.

Check Answer

44. QID - 2.1342

Consider that `str` is a variable of class `java.lang.String`.
Which of the following lines of code may throw a `NullPointerException` in certain situations?

Or a tougher version of the question could be :
Which of the following lines of code are not an example of robust design ?

Select 3 options

A. `if ( (str != null) | ( i == str.length() ) )`

B. `if ( (str == null) | ( i == str.length() ) )`

C. `if ( (str != null) || (i == str.length() ) )`

D. `if ( (str == null) || (i == str.length() ) )`

Check Answer

45. QID - 2.1278

Which of the following four constructs are valid?

```
1.
   switch(5)
  {
      default :
   }

2.
   switch(5)
   {
      default : break;
   }

3.
  switch(8);

4.
 int x = 0;
 switch(x){
 }
```

Select 1 option

A. 1, 3

B. 1, 2, 3

C. 3, 4

D. 1, 2, 4

E. All are valid.

Check Answer

46. QID - 2.1316

Which of the following statements will correctly create and initialize an array of Strings to non null elements?

Select 4 options

A. `String[] sA = new String[1] { "aaa"};`

B. `String[] sA = new String[] { "aaa"};`

C. `String[] sA = new String[1] ; sA[0] = "aaa";`

D. `String[] sA = {new String( "aaa")};`

E. `String[] sA = { "aaa"};`

Check Answer

47. QID - 2.1275

How can you declare 'i' so that it is not visible outside the package `test`.

```
package test;
public class Test{
   XXX int i;
   /*  irrelevant code */
}
```

Select 2 options

A. private

B. public

C. protected

D. No access modifier

E. friend

Check Answer

48. QID - 2.826

What will be the output when the following program is run?

```
package exceptions;
public class TestClass {
    public static void main(String[] args) {
        try{
            doTest();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void doTest() throws MyException{
        int[] array = new int[10];
        array[10] = 1000;
        doAnotherTest();
    }

    static void doAnotherTest() throws MyException{
        throw new MyException("Exception from doAnotherTest");
    }
}
class MyException extends Exception {
    public MyException(String msg){
     super(msg);
    }
}
```

Select 1 option

A.
```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 10
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

B.
```
Error in thread "main" java.lang.ArrayIndexOutOfBoundsException
```

C.
```
exceptions.MyException: Exception from doAnotherTest
```

D.
```
exceptions.MyException: Exception from doAnotherTest
    at exceptions.TestClass.doAnotherTest(TestClass.java:29)
    at exceptions.TestClass.doTest(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

Check Answer

49. QID - 2.1095

What will be the result of compiling and running the following code?

```
class Base{
   public short getValue(){ return 1; } //1
}
class Base2 extends Base{
   public byte getValue(){ return 2; } //2
}
public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Select 1 option

A. It will print 1

B. It will print 2.

C. Compile time error at //1

D. Compile time error at //2

E. Compile time error at //3

Check Answer

50. QID - 2.1137

What will the following code snippet print?

```
int index = 1;
String[] strArr = new String[5];
String   myStr  = strArr[index];
System.out.println(myStr);
```

Select 1 option

A. nothing

B. `null`

C. It will throw `ArrayIndexOutOfBounds` at runtime.

D. It will print some junk value.

E. None of the above.

Check Answer

51. QID - 2.884

Consider the following code snippet:

```
public class Test{
  void test(){
      MyClass obj = new MyClass();
      obj.name = "jack";
      // 1 insert code here
  }
}

//In MyClass.java
public class MyClass{
  int value;
  String name;
}
```

What can be inserted at // 1, which will make the object referred to by obj eligible for garbage collection?

Select 1 option

A. `obj.destroy();`

B. `Runtime.getRuntime().gc();`

C. `obj = null;`

D. `obj.finalize()`

E. `obj.name = null; as well as obj = null;`

Check Answer

52. QID - 2.1244

Given the following program, which statements are true?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      A[] a, a1;
      B[] b;
      a = new A[10]; a1  = a;
      b =  new B[20];
      a = b;  // 1
      b = (B[]) a;  // 2
      b = (B[]) a1; // 3
   }
}
class A { }
class B extends A { }
```

Select 2 options

A. Compile time error at line 3.

B. The program will throw a java.lang.ClassCastException at the line labelled 2 when run.

C. The program will throw a java.lang.ClassCastException at the line labelled 3 when run.

D. The program will compile and run if the (B[ ]) cast in the line 2 and the whole line 3 is removed.

E. The cast at line 2 is needed.

Check Answer

53. QID - 2.1356

How many times will the line marked //1 be called in the following code?

```
int x = 10;
do{
 x--;
 System.out.println(x);  // 1
}while(x<10);
```

Select 1 option

A. 0

B. 1

C. 9

D. 10

E. None of these.

Check Answer

54. QID - 2.1035

Which of the following statements are true?

Select 2 options

A. The `extends` keyword is used to specify inheritance.

B. subclass of a non-abstract class cannot be declared `abstract`.

C. subclass of an abstract class can be declared `abstract`.

D. subclass of a final class cannot be `abstract`.

E. A class, in which all the members are declared `private`, cannot be declared `public`.

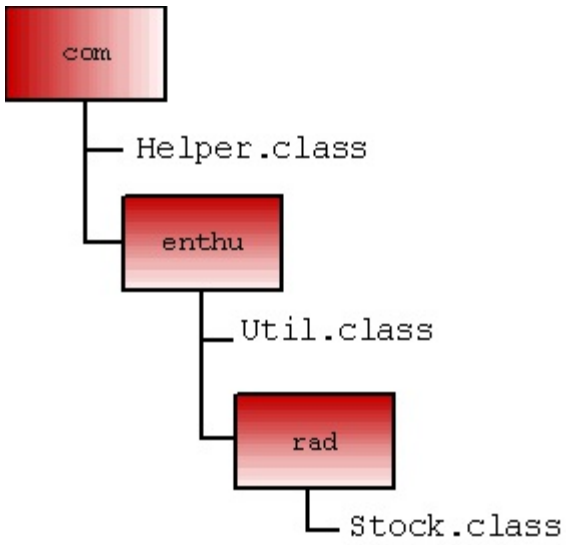
Check Answer

55. QID - 2.1066

Consider the following directory structure shown in Image 1 that displays available folders and classes and the code given below.

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements must be added to the above class?



Select 2 options

A. `import com.enthu.*;`

B. `import com.*.*;`

C. `import *.*.*;`

D. `import com.*;`

E. `import com.enthu.rad.*;`

F. `import all;`

Check Answer

56. QID - 2.1093

Which statements regarding the following code are correct ?

```
class Base{
   void method1() throws java.io.IOException, NullPointerException{
      someMethod("arguments");
      // some I/O operations
   }
   int someMethod(String str){
      if(str == null) throw new NullPointerException();
      else return str.length();
   }
}
public class NewBase extends Base{
      void method1(){
           someMethod("args");
      }
}
```

Select 2 options

A. method1 in class NewBase does not need to specify any exceptions.

B. The code will not compile because RuntimeExceptions cannot be given in throws clause.

C. method1 in class NewBase must at least give `IOException` in its throws clause.

D. method1 in class NewBase must at least give `NullpointerException` in its throws clause.

E. There is no problem with the code.

Check Answer

57. QID - 2.1190

What will the following code snippet print?

```
Object t = new Integer(107);
int k = (Integer) t.intValue()/9;
System.out.println(k);
```

Select 1 option

A. 11

B. 12

C. It will not compile.

D. It will throw an exception at runtime.

Check Answer

58. QID - 2.980

Given the following class definitions, the expression

```
(obj instanceof A) && ! (obj instanceof C) && ! (obj instanceof D)
```

correctly identifies whether the object referred to by obj was created by instantiating class B rather than classes A, C and D?

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}
```

Select 1 option

A. True

B. False

Check Answer

59. QID - 2.1235

A Java programmer is developing a desktop application. Which of the following exceptions would be appropriate for him to throw explicitly from his code?

Select 1 option

A. `NullPointerException`

B. `ClassCastException`

C. `ArrayIndexOutofBoundsException`

D. `Exception`

E. `NoClassDefFoundError`

Check Answer

60. QID - 2.986

The following method will compile and run without any problems.

```
public void switchTest(byte x){
   switch(x){
      case 'b':   // 1
      default :   // 2
      case -2:    // 3
      case 80:    // 4
   }
}
```

Select 1 option

A. True

B. False

Check Answer

61. QID - 2.885

Given:

```
//Insert code here

   public abstract void draw();
}


//Insert code here
   public void draw(){  System.out.println("in draw..."); }
}
```

Which of the following lines of code can be used to complete the above code?

Select 2 options

A. class Shape {

and

class Circle extends Shape {

B. public class Shape {

and

class Circle extends Shape {

C. abstract Shape {

and

public class Circle extends Shape {

D. public abstract class Shape {

and

class Circle extends Shape {

E. public abstract class Shape {

and

class Circle implements Shape {

F. public interface Shape {

and

class Circle implements Shape {

Check Answer

62. QID - 2.1105

Consider the following method...

```
public void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   System.out.println("True False");
   else         // 3
   System.out.println("True True");
   else         // 4
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Select 3 options

A. If run with an argument of 'false', it will print 'False False'

B. If run with an argument of 'false', it will print 'True True'

C. If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

E. It will not compile.

Check Answer

63. QID - 2.1367

What should be inserted in the code given below at line marked //10:

```
class MyClass{
}

class MyComparable implements Comparable<MyClass>{
   public int compareTo(  *INSERT CODE HERE*  x ){ //10
         return 0;
   }
}
```

Select 1 option

A. `Object`

B. `MyClass`

C. `Object<MyClass>`

D. `Comparable<MyClass>`

E. Comparable

Check Answer

64. QID - 2.1100

Consider the following class...

```
class TestClass{
    void probe(int... x) { System.out.println("In ..."); }  //1

    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(long x) { System.out.println("In long"); } //3

    void probe(Long x) { System.out.println("In LONG"); } //4

    public static void main(String[] args){
        Integer a = 4; new TestClass().probe(a); //5
        int b = 4; new TestClass().probe(b); //6
    }
}
```

What will it print when compiled and run?

Select 2 options

A. In Integer and In long

B. In ... and In LONG, if //2 and //3 are commented out.

C. In Integer and In ..., if //4 is commented out.

D. It will not compile, if //1, //2, and //3 are commented out.

E. In LONG and In long, if //1 and //2 are commented out.

Check Answer

65. QID - 2.872

Consider the following classes:

```
class A {
      public int getCode(){ return 2;}
}

class AA extends A {
  public void doStuff() {
  }
}
```

Given the following two declarations, which of the options will compile?

```
A a = null;
AA aa = null;
```

Select 4 options

A. `a = (AA)aa;`

B. `a = new AA();`

C. `aa = new A();`

D. `aa = (AA) a;`

E. `aa = a;`

F. `((AA)a).doStuff();`

Check Answer

66. QID - 2.1323

What is the result of compiling and running this code?

```
class MyException extends Throwable{}
class MyException1 extends MyException{}
class MyException2 extends MyException{}
class MyException3 extends MyException2{}
public class ExceptionTest{
   void myMethod() throws MyException{
      throw new MyException3();
   }
   public static void main(String[] args){
      ExceptionTest et = new ExceptionTest();
      try{
         et.myMethod();
      }
      catch(MyException me){
         System.out.println("MyException thrown");
      }
      catch(MyException3 me3){
         System.out.println("MyException3 thrown");
      }
      finally{
         System.out.println(" Done");
      }
   }
}
```

Select 1 option

A. MyException thrown

B. MyException3 thrown

C. MyException thrown Done

D. MyException3 thrown Done

E. It fails to compile

Check Answer

67. QID - 2.945

Which of the following statements are true?

Select 2 options

A. Private methods cannot be overridden in subclasses.

B. A subclass can override any method in a non-final superclass.

C. An overriding method can declare that it throws a wider spectrum of checked exceptions than the method it is overriding.

D. The parameter list of an overriding method must be a subset of the parameter list of the method that it is overriding.

E. The overriding method may opt not to declare any throws clause even if the original method has a throws clause.

Check Answer

68. QID - 2.848

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        do{
            System.out.println(k);
        }while(--k>0);
    }
}
```

Select 1 option

A.
```
1
```

B.
```
1
0
```

C.
```
2
1
```

D.
```
2
1
0
```

E. It will keeping printing numbers in an infinite loop.

F. It will not compile.

Check Answer

69. QID - 2.1079

Consider the following two classes defined in two .java files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1  <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       System.out.println(X.LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Select 1 option

A. `import static X;`

B. `import static com.foo.*;`

C. `import static com.foo.X.*;`

D. `import com.foo.*;`

E. `import com.foo.X.LOGICID;`

Check Answer

70. QID - 2.1363

In java, Strings are immutable. A direct implication of this is...

Select 2 options

A. you cannot call methods like "1234".replace('1', '9'); and expect to change the original String.

B. you cannot change a String object, once it is created.

C. you can change a String object only by the means of its methods.

D. you cannot extend String class.

E. you cannot compare String objects.

Check Answer

71. QID - 2.1135

Which of the following statements is/are true?

Select 1 option

A. Subclasses must define all the abstract methods that the superclass defines.

B. A class implementing an interface must define all the methods of that interface.

C. A class cannot override the super class's constructor.

D. It is possible for two classes to be the superclass of each other.

E. An interface can implement multiple interfaces.

Check Answer

72. QID - 2.987

What will be the result of attempting to run the following program?

```
public class StringArrayTest{
   public static void main(String args[]){
      String[][][] arr  ={{ { "a", "b" , "c"}, { "d", "e", null } },
{ {"x"}, null },{{"y"}},{ { "z","p"}, {} }
      };
      System.out.println(arr[0][1][2]);
   }
}
```

Select 1 option

A. It will throw NullPointerBoundsException.

B. It will throw ArrayIndexOutOfBoundsException.

C. It will print null.

D. It will run without any error but will print nothing.

E. None of the above.

Check Answer

73. QID - 2.894

The following are the complete contents of TestClass.java file. Which packages are automatically imported?

```
class TestClass{
   public static void main(String[] args){
     System.out.println("hello");
   }
}
```

Select 2 options

A. java.util

B. System

C. java.lang

D. java.io

E. String

F. The package with no name.

Check Answer

74. QID - 2.1012

Which lines contain a valid constructor in the following code?

```
public class TestClass{
   public TestClass(int a, int b) { } // 1
   public void TestClass(int a) { }   // 2
   public TestClass(String s); // 3
   private TestClass(String s, int a) { }     //4
   public TestClass(String s1, String s2) { }; //5
}
```

Select 3 options

A. Line // 1

B. Line // 2

C. Line // 3

D. Line // 4

E. Line // 5

Check Answer

75. QID - 2.1279

Which of the following code snippets will compile without any errors?
(Assume that the statement `int x = 0;` exists prior to the statements below.)

Select 3 options

A. `while (false) { x=3; }`

B. `if (false) { x=3; }`

C. `do{ x = 3; } while(false);`

D. `for( int i = 0; i< 0; i++) x = 3;`

Check Answer

76. QID - 2.1098

What will be the result of attempting to compile the following program?

```
public class TestClass{
   long l1;
   public void TestClass(long pLong) { l1 = pLong ; }  //(1)
   public static void main(String args[]){
      TestClass a, b ;
      a = new TestClass();  //(2)
      b = new TestClass(5);  //(3)
   }
}
```

Select 1 option

A. A compilation error will be encountered at (1), since constructors should not specify a return value.

B. A compilation error will be encountered at (2), since the class does not have a default constructor.

C. A compilation error will be encountered at (3).

D. The program will compile correctly.

E. It will not compile because parameter type of the constructor is different than the type of value passed to it.

Check Answer

77. QID - 2.998

What will the following program print when compiled and run?

```
class Game{
  public void play() throws Exception{
    System.out.println("Playing...");
  }
}

public class Soccer extends Game{
   public void play(){
      System.out.println("Playing Soccer...");
   }
   public static void main(String[] args){
       Game g = new Soccer();
       g.play();
   }
}
```

Select 1 option

A. It will not compile.

B. It will throw an Exception at runtime.

C. Playing Soccer...

D. Playing...

E. None of these.

Check Answer

78. QID - 2.1016

What will the following code snippet print?

```
int count = 0, sum = 0;
do{
        if(count % 3 == 0) continue;
        sum+=count;
}
while(count++ < 11);
System.out.println(sum);
```

Select 1 option

A. 49

B. 48

C. 37

D. 36

E. 38

Check Answer

79. QID - 2.1107

Which of the following are correct about "encapsulation"?

Select 2 options

A. Encapsulation is same as polymorphism.

B. It helps make sure that clients have no accidental dependence on the choice of representation

C. It helps avoiding name clashes as internal variables are not visible outside.

D. Encapsulation makes sure that messages are sent to the right object at run time.

E. Encapsulation helps you inherit the properties of another class.

Check Answer

80. QID - 2.1298

Which of the following are valid declarations within a class?

Select 2 options

A. `volatile int k;`

B. `abstract boolean bool;`

C. `native float radix;`

D. `friendly int ArrayList;`

E. `int friendly, ArrayList;`

Check Answer

81. QID - 2.1270

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      boolean b = false;
      int i = 1;
      do{
         i++ ;
      } while (b = !b);
      System.out.println( i );
   }
}
```

Select 1 option

A. The code will fail to compile, 'while' has an invalid condition expression.

B. It will compile but will throw an exception at runtime.

C. It will print 3.

D. It will go in an infinite loop.

E. It will print 1.

Check Answer

82. QID - 2.1065

What will the following code print?

```
boolean flag = true;
if(flag = false){
   System.out.println("1");
}else if(flag){
   System.out.println("2");
}else if(!flag){
   System.out.println("3");
}else    System.out.println("4");
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. Compilation error.

Check Answer

83. QID - 2.1134

Which of the following are valid at line 1?

```
public class X{
   //line 1: insert code here.
}
```

Select 2 options

A. String s;

B. String s = 'asdf';

C. String s = 'a';

D. String s = this.toString();

E. String s = asdf;

Check Answer

84. QID - 2.1208

Given the following class definition:

```
class A{
  protected int i;
  A(int i) {    this.i = i;    }

}
// 1 : Insert code here
```

Which of the following would be a valid class that can be inserted at //1 ?

Select 2 options

A. `class B {}`

B. `class B extends A {}`

C. `class B extends A {  B()  {  System.out.println("i = " + i); }  }`

D. `class B { B() {} }`

Check Answer

85. QID - 2.1083

What will the following program print?

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

Select 1 option

A. 2

B. 3

C. 6

D. 7

E. 9

Check Answer

86. QID - 2.964

What letters, and in what order, will be printed when the following program is compiled and run?

```
public class FinallyTest{
   public static void main(String args[]) throws Exception{
       try{
          m1();
          System.out.println("A");
       }
       finally{
          System.out.println("B");
       }
       System.out.println("C");
   }
   public static void m1() throws Exception { throw new Exception();
}
```

Select 1 option

A. It will print C and B, in that order.

B. It will print A and B, in that order.

C. It will print B and throw Exception.

D. It will print A, B and C in that order.

E. Compile time error.

Check Answer

87. QID - 2.887

Given:

```
class OverloadingTest{

    void m1(int x){
        System.out.println("m1 int");
    }

    void m1(double x){
        System.out.println("m1 double");
    }

    void m1(String x){
        System.out.println("m1 String");
    }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        OverloadingTest ot = new OverloadingTest();
        ot.m1(1.0);
    }
}
```

What will be the output?

Select 1 option

A. It will fail to compile.

B. m1 int

C. m1 double

D. m1 String

Check Answer

88. QID - 2.890

Given:

```
class Square {
    private double side = 0;
    String color;
    public Square(double length){
        this.side = length;
    }
    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Square mysq = new Square(10);
        mysq.color = "red";

        //set mysq's side to 20
    }
}
```

Which of the following statements will set mysq's side to 20?

Select 1 option

A. `mysq.side = 20;`

B. `mysq = new Square(20);`

C. `mysq.setSide(20);`

D. `side = 20;`

E. `Square.mysql.side = 20;`

Check Answer

89. QID - 2.1271

What will be the result of attempting to compile and run the following code?

```
public class PromotionTest{
   public static void main(String args[]){
      int i = 5;
      float f = 5.5f;
      double d = 3.8;
      char c = 'a';
      if (i == f) c++;
      if (((int) (f + d)) == ((int) f + (int) d)) c += 2;
      System.out.println(c);
   }
}
```

Select 1 option

A. The code will fail to compile.

B. It will print d.

C. It will print c.

D. It will print b

E. It will print a.

Check Answer

90. QID - 2.842

A new Java programmer has written the following method that takes an array of integers and sums up all the integers that are less than 100.

```
public void processArray(int[] values){
        int sum = 0;
        int i = 0;
        try{
            while(values[i]<100){
                sum = sum +values[i];
                i++;
            }
        }
        catch(Exception e){ }
        System.out.println("sum = "+sum);
    }
```

Which of the following are best practices to improve this code?

Select 2 options

A. Use `ArrayIndexOutOfBoundsException` for the catch argument.

B. Use `ArrayIndexOutOfBoundsException` for the catch argument and add code in the catch block to log or print the exception.

C. Add code in the catch block to handle the exception.

D. Use flow control to terminate the loop.

Check Answer

Test 1 (Answered)

01. QID - 2.1177 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Option is : B

~~**A.**~~ For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `true`.

> `instanceof` operator does not accept objects as the right hand side operand. The operand at the right hand side should be a class. Therefore, this expression will not compile.

B. For any non-null reference `o1`, the expression `(o1 instanceof Object)` will always yield `true`.

> Because all objects in Java derive from Object class.

~~**C.**~~ For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `false`.

> It is wrong for the same reason as option 1.

~~**D.**~~ For any non-null reference `o1`, the expression `(o1 instanceof Object)` may yield `false`.

> Since all objects in Java derive from Object class, there is no way it will ever return `false`.

~~**E.**~~ None of the above.

Explanation:

You should understand here that `instanceof` operator returns true even if the Right Hand Side is a super class.
For example,

```
class Animal {}
class Dog extends Animal { }
Dog d = new Dog();
```

Now, `d instanceof Animal` will be true because even though `d` is actually an instance of `Dog`, since `Dog` is a subclass of `Animal`, `Dog` IS-A `Animal`.

Back to Question without Answer

02. QID - 2.1320 : Working with Inheritance

What will the following code print when compiled and run?

```
class ABCD{
   int x = 10;
   static int y = 20;
}
class MNOP extends ABCD{
   int x = 30;
   static int y = 40;
}

public class TestClass {
   public static void main(String[] args) {
     System.out.println(new MNOP().x+", "+new MNOP().y);
   }
}
```

Correct Option is : D

~~**A.**~~ 10, 40

~~**B.**~~ 30, 20

~~**C.**~~ 10, 20

D. 30, 40

~~**E.**~~ 20, 30

~~**F.**~~ Compilation error.

Explanation:

Access to static and instance fields and static methods depends on the class of reference variable and not the actual object to which the variable points to. Observe that this is opposite of what happens in the case of instance methods. In case of instance methods the method of the actual class of the object is called.

Therefore, in case of `System.out.println(new MNOP().x);` the reference is of type MNOP and so MNOP's x will be accessed.

Had it been like this:

```
ABCD a = new MNOP();
System.out.println(a.x);
System.out.println(a.y);
```

ABCD's x and y would have been accessed because a is of type ABCD even though the actual object is of type MNOP.

Back to Question without Answer

03. QID - 2.1229 : Creating and Using Arrays

Which of these array declarations and instantiations are legal?

Correct Options are : A B D E

A. `int[ ] a[ ] = new int [5][4] ;`

This will create an array of length 5. Each element of this array will be an array of 4 ints.

B. `int a[ ][ ] = new int [5][4] ;`

This will create an array of length 5. Each element of this array will be an array of 4 ints.

C. ~~`int a[ ][ ] = new int [ ][4] ;`~~

The statement int[ ][4] will not compile, because the dimensions must be created from left to right.

D. `int[ ] a[ ] = new int[4][ ] ;`

This will create an array of length 4. Each element of this array will be `null`. But you can assign an array of ints of any length to any of the elements. For example:

```
a[0] = new int[10];//valid
a[1] = new int[4];//valid
a[2] = new int[]; //invalid because you must specify the length
a[3] = new Object[] //invalid because a[3] can only refer to an
array of ints.
```

This shows that while creating a one dimensional array, the length must be

specified but while creating multidimensional arrays, the length of the last dimension can be left unspecified. Further, the length of multiple higher dimensions after the first one can also be left unspecified if none of the dimensions are specified after it. So for example,
`a[][][][] = new int[4][3][3][5];` is same as `a[][][][] = new int[4][][][];` (Note that the first dimension must be specified.)

Thus, multidimensional arrays do not have to be symmetrical.

E. `int[ ][ ] a = new int[5][4] ;`

This will create an array of length 5. Each element of this array will be an array of 4 ints.

Explanation:

The [] notation can be placed both before and after the variable name in an array declaration.

```
int[] ia, ba;  // here ia and ba both are int arrays.
int ia[], ba; //here only ia is int array and ba is an int.
```

Multidimensional arrays are created by creating arrays that can contain references to other arrays .

Back to Question without Answer

04. QID - 2.1232 : Java Basics

Which of the following lines can be inserted at line 1 to make the program run?

```
//line 1
public class TestClass{
  public static void main(String[] args){
     PrintWriter pw = new PrintWriter(System.out);
     OutputStreamWriter osw  =  new OutputStreamWriter( System.out
);
     pw.print("hello");
  }
}
```

Assume that `PrintWriter` and `OutputStreamWriter` are valid classes in `java.io` package.

Correct Option is : B

~~**A.**~~ `import java.lang.*;`

> Although you can import java.lang package explicitly, it is not required because this package is always imported by the compiler.

B. `import java.io.*;`

> This will make all the classes of java.io package available.

~~**C.**~~ `import java.io.OutputStreamWriter;`

> This will only make OutputStreamwriter available. PrintWriter will still be unavailable.

~~**D.**~~ `include java.io.*;`

include is not valid keyword in Java.

E. ~~`include java.lang.System;`~~

Back to Question without Answer

05. QID - 2.1044 : Using Loop Constructs

What will be the output if you run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0 ; j < 1 ; ++j , i++){
         System.out.println( i + " " + j );
      }
      System.out.println( i + " " + j );
   }
}
```

Correct Option is : D

~~**A.**~~ `0 0` will be printed twice.

~~**B.**~~ `1 1` will be printed once.

~~**C.**~~ `0 1` will be printed followed by `1 2`.

D. `0 0` will be printed followed by `1 1`.

~~**E.**~~ It will print `0 0` and then `0 1`.

Explanation:

`j` will be less than `1` for only the first iteration. So, first it will print `0, 0`. Next, `i` and

`j` are incremented.

Because `j` is not less than `1` at the start of the loop, the condition fails and it comes out of the loop. Finally, it will print `1,1`.

Back to Question without Answer

06. QID - 2.1039 : Using Operators and Decision Constructs

What will the following method return if called with an argument of 7?

```
public int transformNumber(int n){
   int radix = 2;
   int output = 0;
   output += radix*n;
   radix = output/radix;
   if(output<14){
       return output;
   }
   else{
       output = output*radix/2;
       return output;
   }
   else {
       return output/2;
   }
}
```

Correct Option is : D

~~**A.**~~ 7

~~**B.**~~ 14

~~**C.**~~ 49

D. Compilation fails.

The if-else-else is invalid. It should be if , else if, else.

Back to Question without Answer

07. QID - 2.1347 : Handling Exceptions

Consider the following code...

```
public class TestClass{
     class MyException extends Exception {}
     public void myMethod() throws XXXX{
         throw new MyException();
     }
}
```

What can replace XXXX?

Correct Options are : A B D

A. MyException

B. Exception

> Because Exception is a superclass of MyException.

~~**C.**~~ No throws clause is necessary

> It is needed because MyException is a checked exception. Any exception that extends java.lang.Exception but is not a subclass of java.lang.RuntimeException is a checked exception.

D. Throwable

> Because Throwable is a super class of Exception.

~~**E.**~~ RuntimeException

Explanation:

You can use `Throwable` as well as `Exception` as both of them are super classes of `MyException`.
`RuntimeException` (and its subclasses such as `NullPointerException` and `ArrayIndexOutOfBoundsException`) is not a checked exception. So it cannot cover for `MyException` which is a checked exception.
You cannot use `Error` as well because it is not in the hierarchy of `MyException`, which is Object <- Throwable <- Exception <- MyException.

Back to Question without Answer

08. QID - 2.1225 : Working with Java Data Types - String, StringBuilder

Which of these statements concerning the `charAt()` method of the `String` class are true?

Correct Options are : A E

A. The `charAt( )` method can take a char value as an argument.

> Yes, it can because it takes an int and char will be implicitly promoted to int.

~~**B.**~~ The `charAt( )` method returns a Character object.

> It returns char.

~~**C.**~~ The expression `char ch = "12345".charAt(3)` will assign 3 to ch.

> It will assign 4 as indexing starts from 0.

~~**D.**~~ The expression `char ch = str.charAt(str.length())` where str is "12345", will assign 3 to ch.

> It will throw IndexOutOfBoundsException as `str.length()` is 5 and there is no `str.charAt(5);`

E. The index of the first character is 0.

~~**F.**~~ It throws `StringIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

~~G.~~ It throws `ArrayIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

Explanation:

Since indexing starts with 0, the maximum value that you can pass to `charAt` is length-1.

As per the API documentation for `charAt`, it throws `IndexOutOfBoundsException` if you pass an invalid value.

Both - `ArrayIndexOutOfBoundsException` and `StringIndexOutOfBoundsException`, extend `IndexOutOfBoundsException` and although in practice, the `charAt` method throws `StringIndexOutOfBoundsException`, it is not a valid option because the implementation is free to throw some other exception as long as it is an `IndexOutOfBoundsException`. There are questions in the exam on this aspect.

Back to Question without Answer

09. QID - 2.934 : Constructors

You can call only public and protected constructors of the super class from a subclass if the subclass is not in the same package because only those are inherited.

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

Most of the statement is correct but remember that constructors are NEVER (whether public or otherwise) inherited. (The above statement is true for other methods though.) So, if you have a class :

```
class A{
     public A(int i){}
}
```

and another class:

```
class B extends A{
}
```

You cannot do : new B(10); because that is A's constructor and it is not inherited by B. To invoke A's constructor you have to do:

```
class B extends A{
     public B(int i){  super(i); }
```

```
}
```

Back to Question without Answer

10. QID - 2.996 : Working with Inheritance

Consider the following program:

```
class Game {
  public void play() throws Exception   {
    System.out.println("Playing...");
  }
}

class Soccer extends Game {
   public void play(String ball)    {
      System.out.println("Playing Soccer with "+ball);
   }
}

public class TestClass {
   public static void main(String[] args) throws Exception  {
       Game g = new Soccer();
       // 1
       Soccer s = (Soccer) g;
       // 2
   }
}
```

Which of the given options can be inserted at //1 and //2?

Correct Options are : C D

~~**A.**~~ It will not compile as it is.

> There is no problem with the existing code.

~~**B.**~~ It will throw an `Exception` at runtime if it is run as it is.

> `Soccer s = (Soccer) g;` is a valid because `g` does refer to an object of class

`Soccer` at run time. So there will be no exception at run time.

C. `g.play();` at //1 and `s.play("cosco");` at //2

This is valid because `g` is of type `Game`, which has the no-args play method and `s` is of type `Soccer`, which has defined `play(String )` method.

D. `g.play();` at //1 and `s.play();` at //2

This is valid because `g` is of type `Game`, which has the no-args play method and `s` is of type `Soccer`, which inherits that method.

E. ~~`g.play("cosco");`~~ at //1 and `s.play("cosco");` at //2

`g.play("cosco")` is not valid because even though the object referred to by `g` is of class `Soccer`, the reference type of `g` is `Game`, which does not have `play(String )` method.

Back to Question without Answer

11. QID - 2.1021 : Working with Java Data Types - String, StringBuilder

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      String s = "hello";
      StringBuilder sb = new StringBuilder( "hello" );
      sb.reverse();
      s.reverse();
      if( s == sb.toString() )  System.out.println( "Equal" );
      else System.out.println( "Not Equal" );
   }
}
```

Correct Option is : A

A. Compilation error.

> There is no reverse() method in String class.

~~**B.**~~ It will print 'Equal'.

~~**C.**~~ It will print 'Not Equal'.

~~**D.**~~ Runtime error.

~~**E.**~~ None of the above.

Back to Question without Answer

12. QID - 2.915 : Java Basics

Given the following contents of two java source files:

```
package util.log4j;
public class Logger  {
  public void log(String msg){
      System.out.println(msg);
  }
}
```

and

```
package util;
public class TestClass {
    public static void main(String[] args) throws Exception {
        Logger logger = new Logger();
        logger.log("hello");
    }
}
```

What changes, when made independently, will enable the code to compile and run?

Correct Options are : B C

~~**A.**~~ Replace `Logger logger = new Logger();` with:

```
log4j.Logger logger = new log4j.Logger();
```

> If you are not importing a class or the package of the class, you need to use the class's fully qualified name while using it. Here, you need to use `util.log4j.Logger` instead of just `log4j.Logger`:
>
> ```
> util.log4j.Logger logger = new util.log4j.Logger();
> ```

B. Replace package `util.log4j;` with

```
package util;
```

This will put both the classes in the same package and TestClass can then directly use Logger class without importing anything.

C. Replace `Logger logger = new Logger();` with:
`util.log4j.Logger logger = new util.log4j.Logger();`

Using a fully qualified class name always works irrespective of whether you import the package or not. In this case, you are importing package util but Logger is in util.log4j. Therefore, the use of fully qualified class name for Logger, which is util.log4j.Logger, makes it work.

~~**D.**~~ Remove package `util.log4j`; from Logger.

Remember that you can never access a class that is defined in the default package (i.e. the package with no name) from a class in any other package. So if you remove the package statement from Logger, you can't access it from util package, which is where TestClass is.

~~**E.**~~ Add `import log4j;` to TestClass.

This will not help because Logger is in `util.log4j` package and not in `log4j` package.

Back to Question without Answer

13. QID - 2.1333 : Java Basics

Which of the following are not legal Java identifiers?

Correct Option is : A

A. goto

> Even though it is not used anywhere in Java code, it has been kept as a reserved word to prevent its usage in any form.

~~**B.**~~ unsigned

> It is not a reserved word or keyword.

~~**C.**~~ String

> Yes, it is valid. This is a valid statement: String String = "String";

~~**D.**~~ _xyz

> An identifier can start with a _ sign or a $ sign.

~~**E.**~~ $_abc

> Can have _ or $.

~~**F.**~~ iLikeVeryVeryVeryVeryVeryLongIdentifiersThatDontMakeAnySenseAtAll (65 characters)

> There is no restriction on the length of an identifier.

Explanation:

A valid java identifier is composed of a sequence of java letters and digits, the first of which must be a letter. It cannot be same as any java Keywords or literals (i.e. true, false or null). Java letters include uppercase and lowercase ASCII latin letters and _ , $. (According to JLS.) . In fact, class names can serve as a valid identifier as in 'String' in one of the options above.

Back to Question without Answer

14. QID - 2.1088 : Using Operators and Decision Constructs

Consider:
`o1` and `o2` denote two object references to two different objects of same class.
Which of the following statements are true?

Correct Options are : C D

~~**A.**~~ `o1.equals(o2)` will always be false.

> It depends on how the equals method is overridden. If it is not overridden, then it will return false.

~~**B.**~~ `o1.hashCode() == o2.hashCode()` will always be false.

> `hashCode()` can be overridden and so the given statements is not true.

C. `o1 == o2` will always be false.

> The == operator compares whether the two references are pointing to the same object or not. Here, they are not, so it returns false.

D. Nothing can be said about `o1.equals(o2)` regarding what it will return based on the given information.

> It depends on how the class implements this method.

~~**E.**~~ Nothing can be said about `o1 == o2`.

> It will always return false if references are to two different objects.

Explanation:

Note that both `equals()` and `hashCode()` methods can be overridden by the programmer so you can't say anything about what they will return without looking at the code.

Back to Question without Answer

15. QID - 2.924 : Overloading methods

Consider the following class...

```
class TestClass{
    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(Object x) { System.out.println("In Object"); } //3

    void probe(Long x) { System.out.println("In Long"); } //4

    public static void main(String[] args){
        String a = "hello";
        new TestClass().probe(a);
    }
}
```

What will be printed?

Correct Option is : B

~~**A.**~~ In Integer

B. In Object

~~**C.**~~ In Long

~~**D.**~~ It will not compile

Explanation:

Here, we have three overloaded probe methods but there is no probe method that takes

a String parameter. The only one that is able to accept a String is the one that takes Object as a parameter. So that method will be called.

A String cannot be assigned to a variable of class Integer or Long variable, but it can be assigned to a variable of class Object.

Back to Question without Answer

16. QID - 2.1343 : Working with Methods

Consider the following class...

```
class TestClass{
   int x;
   public static void main(String[] args){
      // lot of code.
   }
}
```

Correct Option is : D

~~**A.**~~ By declaring x as static, main can access `this.x`

~~**B.**~~ By declaring x as public, main can access `this.x`

~~**C.**~~ By declaring x as protected, main can access `this.x`

D. main cannot access `this.x` as it is declared now.

Because main() is a static method. It does not have 'this'!

~~**E.**~~ By declaring x as private, main can access `this.x`

Explanation:

It is not possible to access x from main without making it static. Because main is a static method and only static members are accessible from static methods. There is no 'this' available in main so none of the this.x are valid.

Back to Question without Answer

17. QID - 2.1246 : Working with Java Data Types - String, StringBuilder

What will the following statement return?

```
"    hello java guru   ".trim();
```

Correct Option is : C

~~**A.**~~ The line of code will not compile.

> `"    hello java guru   "` is a valid String and `trim()` is a valid method in String class.

~~**B.**~~ `"hellojavaguru"`

> `trim()` does not remove spaces in within the string but the spaces at the beginning and at the end.

C. `"hello java guru"`

~~**D.**~~ `"hello java guru   "`

> It returns a string in which both the leading and trailing white space of the original string are removed.

~~**E.**~~ None of the above

Back to Question without Answer

18. QID - 2.851 : Creating and Using Arrays

Identify the correct statements about ArrayList?

Correct Options are : B C E

~~**A.**~~ Standard JDK provides no subclasses of ArrayList.

> It does.
> Direct Known Subclasses:
> AttributeList, RoleList, RoleUnresolvedList

B. You cannot store primitives in an ArrayList.

> This is true because only Objects can be stored in it.

C. It allows constant time access to all its elements.

> This is true because it implements java.util.RandomAccess interface, which is a marker interface that signifies that you can directly access any element of this collection. This also implies that it takes the same amount of time to access any element.
>
> (Compare that with a LinkedList, which takes more time to access the last element than the first element.)

~~**D.**~~ ArrayList cannot resize dynamically if you add more number of elements than its capacity.

> It does resize dynamically. Compare that to an array, which cannot be resized once created.

E. An ArrayList is backed by an array.

> This is true. The elements are actually stored in an array and that is why is it called an ArrayList.
> (The expression "backed by an array" means that the implementation of ArrayList actually uses an array to store elements.)

Explanation:

ArrayList is a subclass of AbstractList.

```
java.lang.Object
-  java.util.AbstractCollection<E>
   -    java.util.AbstractList<E>
     -      java.util.ArrayList<E>
```

All Implemented Interfaces:
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:
AttributeList, RoleList, RoleUnresolvedList

Back to Question without Answer

19. QID - 2.1013 : Using Loop Constructs

Using a break in a while loop causes the loop to break the current iteration and start the next iteration of the loop.

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

The break statement is to break out of any loop completely. So the current iteration and any other remaining iterations of the loop will not execute.
Control is transferred to the first statement after the loop.

Back to Question without Answer

20. QID - 2.1193 : Using Loop Constructs

Which of these statements are valid when occurring by themselves?

Correct Options are : B D E

~~**A.**~~ `while ( ) break ;`

> The condition expression in a while header is required.

B. `do { break ; } while (true) ;`

~~**C.**~~ `if (true) { break ; }` (When not inside a switch block or a loop)

> Cannot have break or continue in an 'if' or 'else' block.

D. `switch (1) { default : break; }`

> You can use a constant in `switch(...);`

E. `for ( ; true ; ) break ;`

Explanation:

It is not possible to break out of an if statement. But if the if statement is placed within a labelled block or a switch statement or a loop construct, the usage of break in option 3 would be valid.

Back to Question without Answer

21. QID - 2.1136 : Working with Inheritance

Which of the given statements are correct about the following code?

```
//Filename: TestClass.java
class TestClass{
   public static void main(String[] args){
      A a = new A();
      B b = new B();
   };
}
class A implements T1, T2{}
class B extends A implements T1{}
interface T1 { }
interface T2 { }
```

Correct Options are : A B C D

A. `(a instanceof T1)` will return true.

B. `(a instanceof T2)` will return true.

C. `(b instanceof T1)` will return true.

D. `(b instanceof T2)` will return true.

~~**E.**~~ `(b instanceof A)` will return false.

> It will return true because B extends A and 'b' is referring to an object of class B.

Explanation:

Since A implements both T1 and T2, 1 and 2 are correct.
b instanceof A will return true as B is a subclass of A. Note that it is 'A' and not 'a'.
`( b instanceof a )` will not compile.

Back to Question without Answer

22. QID - 2.1368 : Handling Exceptions

Identify the exceptions that are usually thrown by the JVM and the exceptions usually thrown by an application.

Correct Option is : D

~~A.~~ JVM : IllegalStateException, IllegalArgumentException
Application : ClassCastException, NullPointerException, SecurityException

~~B.~~ JVM : IllegalStateException, IllegalArgumentException, ClassCastException,
Application : NullPointerException, SecurityException

~~C.~~ JVM : IllegalStateException, IllegalArgumentException, ClassCastException, NullPointerException
Application : SecurityException

D. JVM : ClassCastException, NullPointerException, SecurityException
Application : IllegalStateException, IllegalArgumentException

~~E.~~ JVM : ClassCastException, NullPointerException
Application : IllegalStateException, IllegalArgumentException, SecurityException

~~F.~~ JVM : ClassCastException, NullPointerException, IllegalStateException
Application : IllegalArgumentException, SecurityException

Explanation:

Note: The terminology "thrown by the JVM" and "thrown programatically or by the

application" is not precise but is used by popular books. If it helps, you can think of the exception categories as "thrown implicitly" and "thrown explicitly". An exception that is thrown even when there is no `throw` statement, is said to be thrown implicitly. For example, calling a method on null will cause a `NullPointerException` to be thrown automatically, even though there is no `throw` statement. On the other hand, a code may throw an exception explicitly by using the `throw` statement. For example, a method code might check an argument for validity and if it finds the argument inappropriate, it may throw an exception by executing `throw new IllegalArgumentException();`.

A quick way to determine who should throw an exception is to see if the exception extends `java.lang.Error`. Errors are always thrown only by the JVM.

Generally, `RuntimeException`s are also thrown by the JVM. However, it is ok for an application code to throw a `RuntimeException` if it makes sense for the application to throw a `RuntimeException` in a given situation.

You should know about the following common exception classes:

`IndexOutOfBoundsException extends RuntimeException`:
Usually thrown by the JVM. Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range. Applications can subclass this class to indicate similar exceptions.
`ArrayIndexOutOfBoundsException and StringIndexOutOfBoundsException` both extend `IndexOutOfBoundsException`.

`IllegalArgumentException` extends `RuntimeException`: If a parameter passed to a method is not valid. Usually thrown by the application.

`IllegalStateException extends RuntimeException`: Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation. Usually thrown by the application.

`ClassCastException extends RuntimeException`: Usually thrown by the JVM. Thrown to indicate that the code has attempted to cast an object to a subclass of which it is not an instance. For example, the following code generates a `ClassCastException`:

```
Object x = new Integer(0);
System.out.println((String)x);
```

`NullPointerException extends RuntimeException`: Usually thrown by the JVM. Thrown when an application attempts to use `null` in a case where an object is required. These include:

- Calling the instance method of a null object.
- Accessing or modifying the field of a null object.
- Taking the length of null as if it were an array.
- Accessing or modifying the slots of null as if it were an array.
- Throwing null as if it were a Throwable value.

Applications should throw instances of this class to indicate other illegal uses of the null object.

`SecurityException extends RuntimeException`: Usually thrown by the JVM. It is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception.

Back to Question without Answer

23. QID - 2.1173 : Using Operators and Decision Constructs

Consider the following class :

```
public class Test{
   public static void main(String[] args){
      if (args[0].equals("open"))
         if (args[1].equals("someone"))
            System.out.println("Hello!");
      else System.out.println("Go away "+ args[1]);
    }
}
```

Which of the following statements are true if the above program is run with the command line :
java Test closed

Correct Option is : B

~~**A.**~~ It will throw `ArrayIndexOutOfBoundsException` at runtime.

B. It will end without exceptions and will print nothing.

~~**C.**~~ It will print `Go away`

~~**D.**~~ It will print `Go away` and then will throw `ArrayIndexOutOfBoundsException`.

~~**E.**~~ None of the above.

Explanation:

As in C and C++, the Java if statement suffers from the so-called "dangling else problem," The problem is that both the outer if statement and the inner if statement might conceivably own the else clause.
In this example, one might be tempted to assume that the programmer intended the else clause to belong to the outer if statement.

The Java language, like C and C++ and many languages before them, arbitrarily decree that an else clause belongs to the innermost if so as the first if() condition fails (args[0] not being "open") there is no else associated to execute. So, the program does nothing. The else actually is associated with the second if. So had the command line been :
`java Test open`, it would have executed the second if and thrown `ArrayIndexOutOfBoundsException`.
If the command line had been:
`java Test open xyz`, it would execute the else part(which is associated with the second if) and would have printed `"Go away xyz"`.

Back to Question without Answer

24. QID - 2.1064 : Working with Inheritance

Consider this code:

```
interface X1{ }
interface X2{ }
class A { }
class B extends A implements X1{ }
class C extends B implements X2{
   D d = new D();
}
class D { }
```

Which of the following statements are true?

Correct Options are : C D E

~~**A.**~~ D is-a B.

~~**B.**~~ B has-a D.

C has-a D.

C. C is-a A

Because C 'is-a' B and B 'is-a' A.

D. C is-a X1

Because C is-a B and B is-a X1.

E. C is-a X2

Because C implements X2

Explanation:

Consider this code:

```
class A extends B implements I{
  D d = new D();
}
```

Now, Inheritance defines an is-a relation , so A is-a B because A extends B. This actually means that A can be used in all the places where B is used. A can substitute for B anywhere because A is-a B. As all objects have Object as their super class, every object 'is-a' Object.

Since A implements I, it is sometimes said that A 'is-like-a' I. That is, although A is not a I but for all purposes A is like an I. The distinction between is-a and is-like-a is not important for the exam. For the purpose of the exam, is-like-a is same as is-a.

Aggregation defines a has-a relation. Here, D is a member object in A so D is contained in A. So it is said that A 'has-a' D.

All Java objects have the class Object as the ultimate superclass, and so Object is always at the root of any inheritance hierarchy.

Back to Question without Answer

25. QID - 2.907 : Working with Inheritance

Consider the following code appearing in Eagle.java

```
class Bird {
    private Bird(){     }
}
class Eagle extends Bird {
    public String name;
    public Eagle(String name){
        this.name = name;
    }

    public static void main(String[] args) {
        System.out.println(new Eagle("Bald Eagle").name);
    }
}
```

What needs to be done to make this code compile?

Correct Option is : D

~~**A.**~~ Nothing, it will compile as it is.

~~**B.**~~ Make Eagle class declaration public:

```
public class Eagle { ... }
```

~~**C.**~~ Make the Eagle constructor private:

```
private Eagle(String name){ ... }
```

D. Make Bird constructor public:

```
public Bird() { ... }
```

~~**E.**~~ Insert super(); as the first line in Eagle constructor:

```
public Eagle(String name){
    super();
    this.name = name;
}
```

Explanation:

Note that if a subclass class constructor doesn't explicitly call the super class constructor, the compiler automatically inserts super(); as the first statement of the base class constructor. So option 5 is not needed.

Since the constructor of Bird is private, the subclass cannot access it and therefore, it needs to be made public.

Back to Question without Answer

26. QID - 2.1256 : Working with Inheritance

Which of these statements are true?

Correct Options are : B E

~~**A.**~~ A `super( )` or `this( )` call must always be provided explicitly as the first statement in the body of the constructor.

> `super()` is automatically added if the sub class constructor doesn't call any of the super class's constructor.

B. If a subclass does not have any declared constructors, the implicit default constructor of the subclass will have a call to `super( )`.

~~**C.**~~ If neither `super( )` or `this( )` is declared as the first statement of the body of a constructor, then `this( )` will implicitly be inserted as the first statement.

> `super()` is added and not `this()`

~~**D.**~~ `super(...)` can only be called in the first line of the constructor but `this(...)` can be called from anywhere.

E. You can either call `super(...)` or `this(...)` but not both.

Explanation:

Note that calling `super()` will not always work because if the super class has defined a constructor with arguments and has not defined a no args constructor then no args constructor will not be provided by the compiler. It is provided only to the class that does not define ANY constructor explicitly.

Back to Question without Answer

27. QID - 2.1112 : Handling Exceptions

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}

class AnotherException extends Exception {}

public class ExceptionTest{
    public static void main(String[] args) throws Exception{
        try{
            m2();
        }
        finally{
            m3();
        }
        catch (NewException e){}
    }

    public static void m2() throws NewException { throw new NewExcep

    public static void m3() throws AnotherException{ throw new Anothe

}
```

Correct Option is : D

~~**A.**~~ It will compile but will throw `AnotherException` when run.

~~**B.**~~ It will compile but will throw `NewException` when run.

~~**C.**~~ It will compile and run without throwing any exceptions.

D. It will not compile.

> Because a catch block cannot follow a finally block!

~~**E.**~~ None of the above.

Explanation:

Syntax of try/catch/finally is:

```
try{
}
catch(Exception1 e) {... }
catch(Exception2 e) {... }
...
catch(ExceptionN e) {... }
finally { ...  }
```

With a try, either a catch and or finally or both can occur.
A try MUST be followed by at least one catch or finally. (Unless it is a try with resources statement, which is not in scope for this exam.)

In Java 7, you can collapse the catch blocks into a single one:

```
try {
  ...
}
catch (SQLException | IOException | RuntimeException e) {
  //In this block, the class of the actual exception object will be
whatever exception is thrown at runtime.
  //But the class of the reference e will be the closest common
super class of all the exceptions in the catch block.
  //In this case, it will be java.lang.Exception because that is the
most specific class that is a super class for all the three
exceptions.
  e.printStackTrace();
}
```

Back to Question without Answer

28. QID - 2.939 : Overloading methods

Which of the following are true regarding overloading of a method?

Correct Option is : B

~~**A.**~~ An overloading method must have a different parameter list and same return type as that of the overloaded method.

> There is no restriction on the return type. If the parameters are different then the methods are totally different (other than the name) so their return types can be anything.

B. If there is another method with the same name but with a different number of arguments in a class then that method can be called as overloaded.

~~**C.**~~ If there is another method with the same name and same number and type of arguments but with a different return type in a class then that method can be called as overloaded.

> For overloading a method, the "signature" of the overloaded methods must be different. In simple terms, a method signature includes method name and the number and type of arguments that it takes. So if the parameter list of the two methods with the same name are different either in terms of number or in terms of the types of the parameters, then they are overloaded.
>
> For example:
> Method m1 is overloaded if you have two methods : `void m1(int k); and void m1(double d);`
> `or if you have: void m1(int k); and void m1(int k, double d);`
>
> Note that return type is not considered a part of the method signature.

~~D.~~ An overloaded method means a method with the same name and same number and type of arguments exists in the super class and sub class.

This is called overriding and not overloading.

Back to Question without Answer

29. QID - 2.1328 : Working with Inheritance

Consider the following classes :

```
interface I{
}
class A implements I{
}

class B extends A {
}

class C extends B{
}
```

And the following declarations:

```
A a = new A();
B b = new B();
```

Identify options that will compile and run without error.

Correct Option is : A

A. `a = (B)(I)b;`

> class B does implement I because it extends A, which implements I. A reference of type I can be cast to any class at compile time. Since B is-a A, it can be assigned to a.

B. `b = (B)(I) a;`

> This will fail at run time because a does not point to an object of class B.

C. `a = (I) b;`

An I is not an A. Therefore, it will not compile.

D. `I i = (C) a;`

It will compile because a C is-a A, which is-a I, and a reference of class A can point to an object of class C. But it will fail at runtime because a does not point to an object of class C.

Back to Question without Answer

30. QID - 2.891 : Using Loop Constructs

What can you do to make the following code compile?

```
public class TestClass {
    public static void main(String[] args) {
        int[] values = { 10, 20, 30 };
        for( /* put code here */ ){
        }
    }
}
```

Correct Options are : A D

A. int k : values

~~**B.**~~ int k in values

~~**C.**~~ int k; k<0; k++

> k must be initialized first. So it should be: int k=0; k<0; k++

D. ;;

> It will cause an infinite loop, but it is valid.

~~**E.**~~ ; k<values.length;k++

> k needs to be declared first.

Back to Question without Answer

31. QID - 2.942 : Creating and Using Arrays

Consider the following code to count objects and save the most recent object ...

```
int i = 0 ;
Object prevObject ;
public void saveObject(List e ){
    prevObject = e ;
    i++ ;
}
```

Which of the following calls will work without throwing an exception?

Correct Options are : A C D

A. `saveObject( new ArrayList() );`

> Because an `ArrayList` is a List.

~~B.~~ `Collection c = new ArrayList(); saveObject( c );`

> `saveObject()` cannot accept c because c is declared of type Collection, which is a superclass of List, but the `saveObject()` method expects a List.

C. `List l = new ArrayList(); saveObject(l);`

D. `saveObject(null);`

> In this case `prevObj` will be set to null.

~~E.~~ `saveObject(0);` //The argument is the number zero and not the letter o

> `0` is an int, which means it is a primitive. So it will be boxed into an `Integer`

object when you pass it to a method that expects an `Object`. However, `Integer` cannot be passed to a method that expects a `List`. Therefore, this option is not valid. Had the method bean `saveObject(Object obj)`, it would have been valid because an `Integer` is an `Object`.

Back to Question without Answer

32. QID - 2.1189 : Working with Java Data Types - Variables and Objects

Which of the following statements are acceptable?

Correct Options are : A B C E

A. `Object o = new java.io.File("a.txt");`
(Assume that java.io.File is a valid class.)

> This is valid because every object in Java is an Object.

B. `Boolean bool = false;`

> bool is a variable of type Boolean and not of a primitive type boolean however this is still valid because Java performs auto-boxing (and unboxing) for primitives and their wrapper types which allows false to be automatically be boxed into a Boolean false object.

C. `char ch = 10;`

> Because 10 can fit into a char.

~~D.~~ `Thread t = new Runnable();`
(Assume that Runnable is a valid interface.)

> Since Runnable is an interface, it cannot be instantiated like this. But you can do :
>
> ```
> Runnable r = new Runnable(){
>                         public void run(){ }
>                     };
> ```

E. `Runnable r = new Thread();`
(Assume that Thread is a class that implements Runnable interface)

Since Thread implements Runnable, this is a valid assignment.

Back to Question without Answer

33. QID - 2.940 : Java Basics

Which of the statements regarding the following code are correct?

```
public class TestClass{
   static int a;
   int b;
   public TestClass(){
      int c;
      c = a;
      a++;
      b += c;
   }
   public static void main(String args[]) {   new TestClass();   }
}
```

Correct Option is : E

~~**A.**~~ The code will fail to compile because the constructor is trying to access static members.

> A constructor (or any other method) can access static members.

~~**B.**~~ The code will fail to compile because the constructor is trying to use static member variable a before it has been initialized.

> static fields are always initialized automatically if you do not initialize them explicitly. So are instance fields.

~~**C.**~~ The code will fail to compile because the constructor is trying to use member variable b before it has been initialized.

~~**D.**~~ The code will fail to compile because the constructor is trying to use local variable

c before it has been initialized.

```
c is getting initialized at line 2: c = a;
```

E. The code will compile and run without any problem.

Explanation:

All the instance or static variables are given a default values if not explicitly initialized. All numeric variable are given a value of zero or equivalent to zero (i.e. 0.0 for double or float).
booleans are initialized to false and objects references are initialized to null.

Back to Question without Answer

34. QID - 2.1073 : Working with Java Data Types - Variables and Objects

Which line(s) of code in the following program will cause a compilation error?

```
public class TestClass{
   static int value = 0; //1
   public static void main(String args[]) //2
   {
      int 2ndArgument = Integer.parseInt(args[2]); //3
      if( true == 2 > 10 ) //4
      {
         value = -10;
      }
      else{
         value =  2ndArgument;
      }
      for( ; value>0; value--) System.out.println("A"); //5
   }
}
```

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 2

C. 3

'2ndArgument' is not a valid identifier name because an identified must not start with a digit (although it can contain a digit.)

~~**D.**~~ 4

== has less precedence than > so true == 2 > 10 is same as true == (2 > 10)

~~E.~~ 5

Back to Question without Answer

35. QID - 2.1001 : Working with Methods

Consider the following class:

```
public class Test{
    public int id;
}
```

Which of the following is the correct way to make the variable 'id' read only for any other class?

Correct Option is : B

~~**A.**~~ Make 'id' private.

> This will not allow others to read or write.

B. Make 'id' private and provide a public method getId() which will return its value.

~~**C.**~~ Make 'id' static and provide a public static method getId() which will return its value.

~~**D.**~~ Make id 'protected'

Explanation:

This is a standard way of providing read only access to internal variables.

Back to Question without Answer

36. QID - 2.1164 : Using Operators and Decision Constructs

Which of the following statements will compile without any error?

Correct Options are : A B C D

A. `System.out.println("a"+'b'+63);`

> Since the first operand is a String all others (one by one) will be converted to String."ab" + 63 => "ab63"

B. `System.out.println("a"+63);`

> Since the first operand is a String all others (one by one) will be converted to String."a" + 'b' => "a63"

C. `System.out.println('b'+new Integer(63));`

> Since the first operand of + one is of numeric type, its numeric value of 98 will be used. Integer 63 will be unboxed and added to 98. Therefore, the final value will be int 161.

D. `String s = 'b'+63+"a";`

> Since the first one is numeric type so, 'b'+63 = 161, 161+"a" = 161a.

E. `String s = 63 + new Integer(10);`

> Since none of '+' the operands is a String, the + operator will not generate a String. However, due to auto-unboxing, it will generate an int value of 73.

Explanation:

+ is overloaded such that if any one of its two operands is a String then it will convert the other operand to a String and create a new string by concatenating the two.
Therefore, in 63+"a" and "a"+63, 63 is converted to "63" and 'b' +"a" and "a"+'b', 'b' is converted to "b".
Note that in 'b'+ 63 , 'b' is promoted to an int i.e. 98 giving 161.

Back to Question without Answer

37. QID - 2.1348 : Handling Exceptions

Which digits and in what order will be printed when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int k = 0;
      try{
         int i = 5/k;
      }
      catch (ArithmeticException e){
         System.out.println("1");
      }
      catch (RuntimeException e){
         System.out.println("2");
         return ;
      }
      catch (Exception e){
         System.out.println("3");
      }
      finally{
         System.out.println("4");
      }
      System.out.println("5");
   }
}
```

Correct Option is : D

~~A.~~ The program will print 5.

~~B.~~ The program will print 1 and 4, in that order.

~~C.~~ The program will print 1, 2 and 4, in that order.

D. The program will print 1, 4 and 5, in that order.

~~E.~~ The program will print 1,2, 4 and 5, in that order.

Explanation:

Division by 0 throws a java.lang.ArithmeticException, which is a RuntimeException. This is caught by the first catch clause because it is the first block that can handle ArithmeticException. This prints 1. Now, as the exception is already handled, control goes to finally which prints 4 and then the try/catch/finally ends and 5 is printed. Remember : finally is always executed even if try or catch return; (Except when there is System.exit() in try.)

Back to Question without Answer

38. QID - 2.1174 : Working with Java Data Types - String, StringBuilder

What will the following program print?

```
public class TestClass{
  static String str = "Hello World";
  public static void changeIt(String s){
    s = "Good bye world";
  }
  public static void main(String[] args){
    changeIt(str);
    System.out.println(str);
  }
}
```

Correct Option is : A

A. `"Hello World"`

~~**B.**~~ `"Good bye world"`

~~**C.**~~ It will not compile.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

Theoretically, java supports Pass by Value for everything (i.e. primitives as well as Objects).

. Primitives are always passed by value.
. Object "references" are passed by value. So it looks like the object is passed by reference but actually it is the value of the reference that is passed.

An example:

```
    Object o1 = new Object(); //Let us say, the object is stored at
memory location 15000.
        //Since o1 actually stores the address of the memory
location where the object is stored, it contains 15000.
```

Now, when you call someMethod(o1); the value 15000 is passed to the method.

Inside the method someMethod():

```
    someMethod( Object localVar) {
            /*localVar now contains 15000, which means it also
points to the same memory location where the object is stored.
            Therefore, when you call a method on localVar, it will
be executed on the same object.
            However, when you change the value of localVar itself,
for example if you do localVar=null,
            it then it starts pointing to a different memory
location. But the original variable o1 still
            contains 15000 so it still points to the same object. */
        }
```

This is what happens in the this question.
In the method changeIt(...) you are giving a new value to the local variable but the original reference remains the same.
If you need even more detailed explanation, please check
http://www.javaranch.com/campfire/StoryPassBy.jsp

Back to Question without Answer

39. QID - 2.1061 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i, j, k;
      i = j = k = 9;
      System.out.println(i);
   }
}
```

Correct Options are : C E

~~**A.**~~ The code will not compile because unlike in c++, operator '=' cannot be chained i.e. a = b = c = d is invalid.

> It can be chained. I.e. assuming all the variables are declared appropriately, `a = b = c = d;` is valid.

~~**B.**~~ The code will not compile as 'j' is being used before getting initialized.

> j is being initialize by the expression k = 9, which evaluates to 9.

C. The code will compile correctly and will display '9' when run.

~~**D.**~~ The code will not compile as 'j' and 'i' are being used before getting initialized.

E. All the variables will get a value of 9.

Explanation:

Every expression has a value, in this case the value of the expression is the value that is assigned to the Right Hand Side of the equation.
k has a value of 9 which is assigned to j and then to i.

Another implication of this is :

```
boolean b = false;
if( b = true) { System.out.println("TRUE");}
```

The above code is valid and will print TRUE. Because `b = true` has a boolean value, which is what an if statement expects.

Note that `if( i = 5) { ... }` is not valid because the value of the expression `i = 5` is an int (5) and not a boolean.

Back to Question without Answer

40. QID - 2.983 : Working with Inheritance

Consider the following classes:

```
class A implements Runnable{ ...}
class B extends A implements Observer { ...}
```

(Assume that Observer has no relation to Runnable.)

and the declarations :

```
A a = new A() ;
B b = new B();
```

Which of the following Java code fragments will compile and execute without throwing exceptions?

Correct Options are : B E

~~**A.**~~ `Object o = a; Runnable r = o;`

B. `Object o = a; Runnable r = (Runnable) o;`

> Here you are explicitly telling the compiler that o refers to an object that is Runnable.

~~**C.**~~ `Object o = a; Observer ob = (Observer) o ;`

> It will compile but will fail at run time as at runtime 'a' does not refer to an object which is Observable

~~**D.**~~ `Object o = b; Observer o2 = o;`

> 'o' is declared as an Object. so same case as in choice 1.

E. `Object o = b; Runnable r = (Runnable) b;`

> Since b is declared of a type that indirectly implements `Runnable`, the compiler can figure out that b will always point to an object that is assignable to a `Runnable`. Therefore, explicit cast is not required here. It is will still work fine with the explicit cast though.

Explanation:

Although o refers to an object which is Runnable but the compiler doesn't know about it. You have to do: `Runnable r = (Runnable) o;`
You can assign a subclass object reference to superclass reference without a cast but to assign a super class object reference to a subclass (or interface) reference you need an explicit cast as in option 2.

Back to Question without Answer

41. QID - 2.1238 : Using Operators and Decision Constructs

What will the following code print when run without any arguments ...

```
public class TestClass {

    public static int m1(int i){
        return ++i;
    }

    public static void main(String[] args) {

        int k = m1(args.length);
        k += 3 + ++k;
        System.out.println(k);
    }

}
```

Correct Option is : C

~~**A.**~~ It will throw `ArrayIndexOutofBoundsException`.

~~**B.**~~ It will throw `NullPointerException`.

C. 6

~~**D.**~~ 5

~~**E.**~~ 7

~~**F.**~~ 2

~~**G.**~~ None of these.

Explanation:

When the program is run without any arguments, `args` gets assigned a string array of size 0. So `NullPointerException` or `ArrayIndexOutofBoundsException` are out of question. Thus, the first call becomes :

```
int k = m1(0);
```

Follow through the code like this:
1. Method m1() uses pre-increment operation. Therefore, first i is incremented and then the new value of i is returned.
2. Thus, k gets the value of 1.

3. Expand the += operator as:

```
k = k + 3 + ++k;
```

This becomes (remember that k = 1 at this point):

`k = 1 + 3 + (++k)` i.e.

`k = 1 + 3 + 2;` (at this point value of k is 2 because of ++k). But the value of Right Hand Side has not yet been assigned to k.

`k = 6;` 6 is assigned to k thereby overwriting the value of 2.

Therefore, the final value of k is 6.

Back to Question without Answer

42. QID - 2.1084 : Using Operators and Decision Constructs

Which of the following are valid operators in Java?

Correct Options are : A B C D

A. !

operates only on booleans

B. ~

bitwise negation. Operates only on integral types.

C. &

bitwise AND

D. %=

similar to += or /=

~~**E.**~~ $

It is not an operator!

Back to Question without Answer

43. QID - 2.1310 : Using Operators and Decision Constructs

Consider the following method...

```
public static void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   if (flag)    //3
   System.out.println("False True");
   else         //4
   System.out.println("True False");
   else         //5
   System.out.println("True True");
   else         //6
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Correct Options are : A D

A. If run with an argument of 'false', it will print 'False False'

~~**B.**~~ If run with an argument of 'false', it will print 'True True'

~~**C.**~~ If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

~~**E.**~~ It will not compile.

Explanation:

Look at it like this:

```
if (flag)       //1
 {
     if (flag)        // 2
     {
          if (flag)         //3
          {
                System.out.println("False True");
          }
          else             //4
          {
                System.out.println("True False");
          }
     }
     else            //5
     {
           System.out.println("True True");
     }
 }
else            //6
{
      System.out.println("False False");
 }
```

Note that if and else do not cascade. They are like opening an closing brackets. So, else at //4 is associated with if at //3 and else at //5 is associated with if at //2

Back to Question without Answer

44. QID - 2.1342 : Using Operators and Decision Constructs

Consider that `str` is a variable of class `java.lang.String`.
Which of the following lines of code may throw a `NullPointerException` in certain situations?

Or a tougher version of the question could be :
Which of the following lines of code are not an example of robust design ?

Correct Options are : A B C

A. `if ( (str != null) | ( i == str.length() ) )`

> `(i == str.length())` will always be executed so if 'str' is null, then str.length() will throw a NullPointerException.

B. `if ( (str == null) | ( i == str.length() ) )`

> `(i == str.length())` will always be executed so if 'str' is null, then str.length() will throw a NullPointerException.

C. `if ( (str != null) || (i == str.length() ) )`

> `(i == str.length())` will only be evaluated if (str != null) is false, and (str != null) will be false if 'str' is null. So it will also throw a NullPointerException.

~~D.~~ ~~`if ( (str == null) || (i == str.length() ) )`~~

> `(i == str.length())` will only be evaluated if (str == null) is false, and (str == null) will be false if 'str' is NOT null. So it will NEVER throw a NullPointerException.

Explanation:

The concept is : `||` and `&&` are short circuiting operation i.e. if the value of the expression can be known by just seeing the first part then the remaining part is not evaluated while `|` and `&` will always let all the parts evaluates.
Let's break this down in two cases:

1. Say `str = null;`

for a, the first part is false and `str.length()` throws `NullPointerException` because str is null.

for b, the first part of it is true but it will still evaluate the second part and as str is null, str.length() throws NullPointerException. Had it been || instead of |, the second part would not have been evaluated and no exception would have been thrown.

for c, the first part of it is false and it will also evaluate the second part which will throw a NullPointerException as str is null.

for d, the first part is true, so the second part is not evaluated.

2. Say, `str = "somestring";` //i.e. str is not null.

for a, the first part is true, so is the second part. No exception is thrown. Note that second part will still be evaluated although by looking at the first part itself we can tell that the whole expression will return true.

for b, the first part is false, and the second part is also true. No exception is thrown.

for c, first part is true, so second part is not evaluated at all. No exception is thrown.

for d, first part is false, so it will evaluate second part. No exception is thrown as str

is not null.

It would be nice if you try to run the following program to understand the concept : (Uncomment only one of the commented lines one by one).

```
public class TestClass {
   public static void main(String[] args) {
      int i = 0;
      String s = "";

      //s = null;
      if ((s != null) | ( i==s.length())) {}
      System.out.println("A");

      //s = null;
      if ((s == null) | ( i==s.length())) {}
      System.out.println("B");

      //s = null;
      if ((s != null) || (i==s.length())) {}
      System.out.println("C");

      s = null;
      if ((s == null) || (i==s.length())) {}
      System.out.println("D");
   }
}
```

Back to Question without Answer

45. QID - 2.1278 : Using Operators and Decision Constructs

Which of the following four constructs are valid?

```
1.
   switch(5)
  {
      default :
   }

2.
   switch(5)
   {
      default : break;
   }

3.
  switch(8);

4.
 int x = 0;
 switch(x){
 }
```

Correct Option is : D

~~**A.**~~ 1, 3

~~**B.**~~ 1, 2, 3

~~**C.**~~ 3, 4

D. 1, 2, 4

Code 3 is invalid because a switch statement must have a body. The body may even be empty as shown in Code 4.

E. All are valid.

Back to Question without Answer

46. QID - 2.1316 : Creating and Using Arrays

Which of the following statements will correctly create and initialize an array of Strings to non null elements?

Correct Options are : B C D E

A. ~~`String[] sA = new String[1] { "aaa"};`~~

> Array size cannot be given here as the array is being initialized in the declaration.

B. `String[] sA = new String[] { "aaa"};`

C. `String[] sA = new String[1] ; sA[0] = "aaa";`

D. `String[] sA = {new String( "aaa")};`

E. `String[] sA = { "aaa"};`

Back to Question without Answer

47. QID - 2.1275 : Working with Methods - Access Modifiers

How can you declare 'i' so that it is not visible outside the package `test`.

```
package test;
public class Test{
   XXX int i;
   /*  irrelevant code */
}
```

Correct Options are : A D

A. private

> Note that the question does not require that 'x' should be accessible from test package. So private is fine.

~~**B.**~~ public

> Marking it public will make it accessible from all classes in all packages.

~~**C.**~~ protected

> It will make it available to a subclass even if the subclass is in a different package.

D. No access modifier

~~**E.**~~ friend

> There is no such modifier in Java.

Back to Question without Answer

48. QID - 2.826 : Handling Exceptions

What will be the output when the following program is run?

```
package exceptions;
public class TestClass {
    public static void main(String[] args) {
        try{
            doTest();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void doTest() throws MyException{
        int[] array = new int[10];
        array[10] = 1000;
        doAnotherTest();
    }

    static void doAnotherTest() throws MyException{
        throw new MyException("Exception from doAnotherTest");
    }
}
class MyException extends Exception {
    public MyException(String msg){
     super(msg);
    }
}
```

Correct Option is : A

A.
```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 10
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

You are creating an array of length 10. Since array numbering starts with 0, the last element would be `array[9]`.
`array[10]` would be outside the range of the array and therefore an `ArrayIndexOutOfBoundsException` will be thrown, which cannot be caught by `catch(MyException )` clause.
The exception is thus thrown out of the main method and is handled by the JVM's uncaught exception handling mechanism, which prints the stack trace.

B. ~~`Error in thread "main" java.lang.ArrayIndexOutOfBoundsException`~~

java.lang.ArrayIndexOutOfBoundsException extends java.lang.RuntimeException, which in turn extends java.lang.Exception. Therefore, ArrayIndexOutOfBoundsException is an Exception and not an Error.

C. ~~`exceptions.MyException: Exception from doAnotherTest`~~

D. ~~`exceptions.MyException: Exception from doAnotherTest`~~

```
    at exceptions.TestClass.doAnotherTest(TestClass.java:29)
    at exceptions.TestClass.doTest(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

Explanation:

Note that there are questions in the exam that test your knowledge about how exception messages are printed.
When you use `System.out.println(exception)`, a stack trace is not printed. Just the name of the exception class and the message is printed.
When you use `exception.printStackTrace()`, a complete chain of the names of the methods called, along with the line numbers, is printed from the point where the exception was thrown and up to the point where the exception was caught.

Back to Question without Answer

49. QID - 2.1095 : Working with Inheritance

What will be the result of compiling and running the following code?

```
class Base{
   public short getValue(){ return 1; } //1
}
class Base2 extends Base{
   public byte getValue(){ return 2; } //2
}
public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Correct Option is : D

~~**A.**~~ It will print 1

~~**B.**~~ It will print 2.

~~**C.**~~ Compile time error at //1

D. Compile time error at //2

~~**E.**~~ Compile time error at //3

Explanation:

In case of overriding, the return type of the overriding method must match exactly to the return type of the overridden method if the return type is a primitive.
(In case of objects, the return type of the overriding method may be a subclass of the return type of the overridden method.)

Back to Question without Answer

50. QID - 2.1137 : Creating and Using Arrays

What will the following code snippet print?

```
int index = 1;
String[] strArr = new String[5];
String   myStr  = strArr[index];
System.out.println(myStr);
```

Correct Option is : B

~~**A.**~~ nothing

B. `null`

~~**C.**~~ It will throw `ArrayIndexOutOfBounds` at runtime.

~~**D.**~~ It will print some junk value.

~~**E.**~~ None of the above.

Explanation:

When you create an array of Objects (here, Strings) all the elements are initialized to null. So in the line 3, null is assigned to myStr.
Note that. empty string is "" (String str = "";) and is not same as null.

Back to Question without Answer

51. QID - 2.884 : Java Basics - Garbage Collection

Consider the following code snippet:

```
public class Test{
  void test(){
      MyClass obj = new MyClass();
      obj.name = "jack";
      // 1 insert code here
  }
}

//In MyClass.java
public class MyClass{
  int value;
  String name;
}
```

What can be inserted at // 1, which will make the object referred to by obj eligible for garbage collection?

Correct Option is : C

~~**A.**~~ `obj.destroy();`

~~**B.**~~ `Runtime.getRuntime().gc();`

> Calling System.gc() or Runtime.getRuntime().gc() will not necessarily run the garbage collector. It only requests the JVM to perform garbage collection but there is no guarantee that the JVM will do it.
>
> By the way, System.gc() is equivalent to Runtime.getRuntime().gc().

C. `obj = null;`

> This will make the object eligible for GC because there are no other references to it.

~~**D.**~~ `obj.finalize()`

~~**E.**~~ `obj.name = null; as well as obj = null;`

> You don't need to do obj.name=null;

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

Nothing can ensure that an object will definitely be destroyed by the garbage collector. You can at most make an object eligible for GC by making sure that there are no references to it.

Back to Question without Answer

52. QID - 2.1244 : Creating and Using Arrays

Given the following program, which statements are true?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      A[] a, a1;
      B[] b;
      a = new A[10]; a1  = a;
      b =  new B[20];
      a = b;  // 1
      b = (B[]) a;  // 2
      b = (B[]) a1; // 3
   }
}
class A { }
class B extends A { }
```

Correct Options are : C E

~~**A.**~~ Compile time error at line 3.

~~**B.**~~ The program will throw a java.lang.ClassCastException at the line labelled 2 when run.

C. The program will throw a java.lang.ClassCastException at the line labelled 3 when run.

~~**D.**~~ The program will compile and run if the (B[ ]) cast in the line 2 and the whole line 3 is removed.

E. The cast at line 2 is needed.

Explanation:

The line 1 will be allowed during compilation, since assignment is done from a subclass reference to a superclass reference.
The cast in line 2 is needed because a superclass reference is assigned to a subclass reference variable. And this works at runtime because the object referenced to by a is actually of an array of B.
Now, the cast at line 3 tells the compiler not to worry, that I'm a good programmer and I know what I am doing and the object referenced by the super class reference (a1) will actually be of class B at run time. So there is no compile time error. But at run time, this fails because the actual object is not an array of B but is an array of A.

Back to Question without Answer

53. QID - 2.1356 : Using Loop Constructs

How many times will the line marked //1 be called in the following code?

```
int x = 10;
do{
 x--;
 System.out.println(x);  // 1
}while(x<10);
```

Correct Option is : E

~~**A.**~~ 0

~~**B.**~~ 1

~~**C.**~~ 9

~~**D.**~~ 10

E. None of these.

Explanation:

A do-while loop is always executed at least once. So in the first iteration, x is decremented and becomes 9. Now the while condition is tested, which returns true because 9 is less than 10. So the loop is executed again with x = 9. In the loop, x is decremented to 8 and the condition is tested again, which again returns true because 8 is less than 10.

As you can see, x keeps on decreasing by one in each iteration and every time the condition x<10 returns true. However, after x reaches -2147483648, which is its MIN_VALUE, it cannot decrease any further and at this time when x-- is executed, the value rolls over to 2147483647, which is Integer.MAX_VALUE. At this time, the condition x<10 fails and the loop terminates.

Back to Question without Answer

54. QID - 2.1035 : Working with Inheritance

Which of the following statements are true?

Correct Options are : A C

A. The `extends` keyword is used to specify inheritance.

~~**B.**~~ subclass of a non-abstract class cannot be declared `abstract`.

C. subclass of an abstract class can be declared `abstract`.

~~**D.**~~ subclass of a final class cannot be `abstract`.

> final class cannot be subclassed.

~~**E.**~~ A class, in which all the members are declared `private`, cannot be declared `public`.

> There is no such rule.

Explanation:

The `extends` clause is used to specify that a class extends another class and thereby inherits all non-private instance members of that class.

A subclass can be declared `abstract` regardless of whether the superclass was declared abstract. A class cannot be declared `abstract` and `final` at the same time. This restriction makes sense because abstract classes need to be subclassed to be useful and final forbids subclasses.

The visibility of the class is not limited by the visibility of its members. A class with all the members declared private can still be declared public or a class having all public members may be declared private.

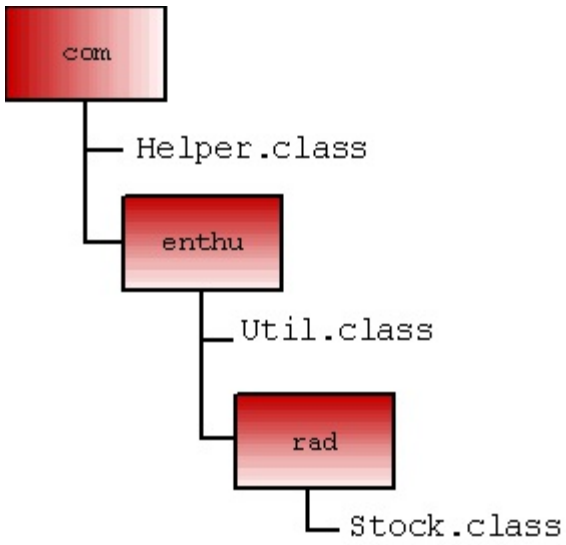
Back to Question without Answer

55. QID - 2.1066 : Java Basics

Consider the following directory structure shown in Image 1 that displays available folders and classes and the code given below.

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements must be added to the above class?



Correct Options are : D E

~~A.~~ `import com.enthu.*;`

This is not required because the code is not using any class from this package.

B. ~~`import com.*.*;`~~

> Bad Syntax. You can only import one package (i.e. all classes in that package) using a * or one class in an import statement.

C. ~~`import *.*.*;`~~

> Bad syntax.

D. `import com.*;`

> This is required because the code is using Helper.class, which exists in com package.

E. `import com.enthu.rad.*;`

> This is required because the code is using Stock.class, which exists in com.enthu.rad package.

F. ~~`import all;`~~

Explanation:

Since the given class does not have any package declaration, it belongs to the default package and therefore it must import com.Helper and com.enthu.rad.Stock classes.

Back to Question without Answer

56. QID - 2.1093 : Handling Exceptions

Which statements regarding the following code are correct ?

```
class Base{
   void method1() throws java.io.IOException, NullPointerException{
      someMethod("arguments");
      // some I/O operations
   }
   int someMethod(String str){
      if(str == null) throw new NullPointerException();
      else return str.length();
   }
}
public class NewBase extends Base{
      void method1(){
           someMethod("args");
      }
}
```

Correct Options are : A E

A. method1 in class NewBase does not need to specify any exceptions.

~~**B.**~~ The code will not compile because RuntimeExceptions cannot be given in throws clause.

> Any Exception can be specified in the throws clause.

~~**C.**~~ method1 in class NewBase must at least give `IOException` in its throws clause.

~~**D.**~~ method1 in class NewBase must at least give `NullpointerException` in its throws clause.

This is not needed because NullPointerException is a RuntimeException.

E. There is no problem with the code.

Explanation:

Overriding method only needs to specify a subset of the list of exception classes the overridden method can throw. A set of no classes is a valid subset of that list.

Remember that NullPointerException is a subclass of RuntimeException, while IOException is a subclass of Exception.

Back to Question without Answer

57. QID - 2.1190 : Using Operators and Decision Constructs

What will the following code snippet print?

```
Object t = new Integer(107);
int k = (Integer) t.intValue()/9;
System.out.println(k);
```

Correct Option is : C

~~**A.**~~ 11

~~**B.**~~ 12

C. It will not compile.

~~**D.**~~ It will throw an exception at runtime.

Explanation:

Compiler will complain that the method intValue() is not available in Object. This is because the . operator has more precedence than the cast operator. So you have to write it like this:

```
int k = ((Integer) t).intValue()/9;
```

Now, since both the operands of / are ints, it is an integer division. This means the resulting value is truncated (and not rounded). Therefore, the above statement will print 11 and not 12.

Back to Question without Answer

58. QID - 2.980 : Using Operators and Decision Constructs

Given the following class definitions, the expression

```
(obj instanceof A) && ! (obj instanceof C) && ! (obj instanceof D)
```

correctly identifies whether the object referred to by obj was created by instantiating class B rather than classes A, C and D?

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}
```

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

The given expression will not be able to distinguish between an object of class A and an object of class B. It will return true in both the cases. Also, The last part `! (obj instanceof D)` of the given expression is redundant because anything which is not instance of C cannot be an instanceof D either!

Correct expression would be `(obj instanceof B) && ! (obj instanceof C)`. This will return true only if obj points to an object of class B and not of A, C, or D.

Back to Question without Answer

59. QID - 2.1235 : Handling Exceptions

A Java programmer is developing a desktop application. Which of the following exceptions would be appropriate for him to throw explicitly from his code?

Correct Option is : D

~~**A.**~~ `NullPointerException`

~~**B.**~~ `ClassCastException`

~~**C.**~~ `ArrayIndexOutofBoundsException`

D. `Exception`

~~**E.**~~ `NoClassDefFoundError`

> `NoClassDefFoundError` is thrown by the JVM when it attempts to load a class and is unable to find the class file.
> Note that it extends `java.lang.Error` and Errors are always thrown by the JVM. A programmer should never throw an Error explicitly.

Explanation:

Observe that all the exceptions given in the options other than `Exception` and `NoClassDefFoundError` are `RuntimeException`s. These are usually thrown implicitly. A programmer should not throw these exceptions explicitly. `java.lang.Exception` and its subclasses (except `RuntimeException`) should be used by the programmer to reflect known exceptional situations, while RuntimeExceptions are used to reflect unforseen or unrecoverable exceptional

situations.

Note: There is no hard and fast rule that says `RuntimeException`s (such as the ones mentioned in this questions) must not be thrown explicitly. It is ok to throw these exceptions explicitly in certain situations. For example, framework/library classes such as Struts, Spring, and Hibernate, and standard JDK classes throw these exceptions explicitly. But for the purpose of the exam, it is a good way to determine if a given application should be thrown expliclity by the programmer or not.

Back to Question without Answer

60. QID - 2.986 : Using Operators and Decision Constructs

The following method will compile and run without any problems.

```
public void switchTest(byte x){
   switch(x){
      case 'b':   // 1
      default :   // 2
      case -2:    // 3
      case 80:    // 4
   }
}
```

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

The following types can be used as a switch variable:

`byte`, `char`, `short`, `int`, `String`, and enums. Note that `long`, `float`, `double`, and `boolean` are not allowed.

All the case constants should be assignable to the switch variable type. i.e. had there been a case label of 128 (case 128 : //some code), it would not have compiled. Because the range of a byte is from -128 to 127 and so 128 is not assignable to 'x'.

The interal value of 'b' is 98, which is less than 127 so Line //1 is fine.

Note: Although it is not required for the exam to know the integral values of characters, it is good to know that all English letters (upper case as well as lower case) as well as 0-9 are below 127 and so are assignable to byte.

Back to Question without Answer

61. QID - 2.885 : Working with Inheritance

Given:

```
//Insert code here

   public abstract void draw();
}


//Insert code here
   public void draw(){  System.out.println("in draw..."); }
}
```

Which of the following lines of code can be used to complete the above code?

Correct Options are : D F

~~**A.**~~ class Shape {

and

class Circle extends Shape {

> Since there is an abstract method in the first class, the class must be declared abstract.

~~**B.**~~ public class Shape {

and

class Circle extends Shape {

~~**C.** abstract Shape {~~

and

public class Circle extends Shape {

> class keyword is missing from the first declaration.

D. public abstract class Shape {

and

class Circle extends Shape {

~~**E.** public abstract class Shape {~~

and

class Circle implements Shape {

> You can only implement an interface not a class. So Circle implements shape is wrong.

F. public interface Shape {

and

class Circle implements Shape {

> By default all the methods of an interface are public and abstract so there is no need to explicitly specify the "abstract" keyword for the draw() method if you make Shape an interface. But it is not wrong to do so.

Back to Question without Answer

62. QID - 2.1105 : Using Operators and Decision Constructs

Consider the following method...

```
public void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   System.out.println("True False");
   else         // 3
   System.out.println("True True");
   else         // 4
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Correct Options are : A C D

A. If run with an argument of 'false', it will print 'False False'

~~**B.**~~ If run with an argument of 'false', it will print 'True True'

C. If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

~~**E.**~~ It will not compile.

Explanation:

Note that if and else do not cascade. They are like opening and closing braces.

```
if (flag)   //1
     if (flag)   //2
        System.out.println("True False");
     else        // 3 This closes //2
        System.out.println("True True");
 else        // 4 This closes //1
    System.out.println("False False");
```

So, else at //3 is associated with if at //2 and else at //4 is associated with if at //1

Back to Question without Answer

63. QID - 2.1367 : Working with Inheritance

What should be inserted in the code given below at line marked //10:

```
class MyClass{
}

class MyComparable implements Comparable<MyClass>{
   public int compareTo(  *INSERT CODE HERE*  x ){ //10
         return 0;
   }
}
```

Correct Option is : B

~~**A.**~~ `Object`

B. `MyClass`

~~**C.**~~ `Object<MyClass>`

~~**D.**~~ `Comparable<MyClass>`

~~**E.**~~ Comparable

Explanation:

Since MyComparable class specifies that it implements the Comparable interface that has been typed to MyClass, it must implement compareTo method that takes a MyClass.
Had it not declared a typed Comparable in its implements clause, compareTo(Object

x) would have been correct.

Back to Question without Answer

64. QID - 2.1100 : Overloading methods

Consider the following class...

```
class TestClass{
    void probe(int... x) { System.out.println("In ..."); }  //1

    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(long x) { System.out.println("In long"); } //3

    void probe(Long x) { System.out.println("In LONG"); } //4

    public static void main(String[] args){
        Integer a = 4; new TestClass().probe(a); //5
        int b = 4; new TestClass().probe(b); //6
    }
}
```

What will it print when compiled and run?

Correct Options are : A D

A. In Integer and In long

~~**B.**~~ In ... and In LONG, if //2 and //3 are commented out.

~~**C.**~~ In Integer and In ..., if //4 is commented out.

D. It will not compile, if //1, //2, and //3 are commented out.

~~**E.**~~ In LONG and In long, if //1 and //2 are commented out.

Explanation:

To answer this type of questions, you need to know the following rules:

1. The compiler always tries to choose the most specific method available with least number of modifications to the arguments.

2. Java designers have decided that old code should work exactly as it used to work before boxing-unboxing functionality became available.

3. Widening is preferred to boxing/unboxing (because of rule 2), which in turn, is preferred over var-args.

Thus,
1.
`probe(Integer)` is bound to `probe(Integer)` (exact match), then with `probe(long)`, then with `probe(int...)` in that order of preference.
`probe(long)` is preferred over `probe(int...)` because unboxing an Integer gives an int and in pre 1.5 code `probe(long)` is compatible with an int (Rule 2).

It is never bound to `probe(Long )` because `Integer` and `Long` are different object types and there is no IS-A relation between them. (This holds true for any two wrapper classes).
It could, however, be bound to `probe(Object )` (if it existed), because `Integer` IS-A `Object`.

2.
`probe(int)` is bound to `probe(long)` (because of Rule 2) , then to `probe(Integer )` because boxing an `int` qives you an `Integer`, which matches exactly to `probe(Integer)`, and then to `probe(int...)`.

It is never bound to `probe(Long )` because int is not compatible with `Long`.

We advise you to run this program and try out various combinations. The exam has questions on this pattern but they are not this tough. If you have a basic understanding, you should be ok.

Back to Question without Answer

65. QID - 2.872 : Working with Inheritance

Consider the following classes:

```
class A {
      public int getCode(){ return 2;}
}

class AA extends A {
  public void doStuff() {
  }
}
```

Given the following two declarations, which of the options will compile?

```
A a = null;
AA aa = null;
```

Correct Options are : A B D F

A. `a = (AA)aa;`

B. `a = new AA();`

C. ~~`aa = new A();`~~

D. `aa = (AA) a;`

`a` is declared as a reference of class A and therefore, at run time, it is possible for `a` to point to an object of class AA (because A is a super class of AA). Hence, the compiler will not complain. Although if `a` does not point to an object of class AA at run time, a `ClassCastException` will be thrown.

E. ~~`aa = a;`~~

A cast is required because the compiler needs to be assured that at run time `a` will point to an object of class AA.

F. `((AA)a).doStuff();`

Once you cast `a` to AA, you can call methods defined in AA. Of course, if `a` does not point to an object of class AA at runtime, a `ClassCastException` will be thrown.

Back to Question without Answer

66. QID - 2.1323 : Handling Exceptions

What is the result of compiling and running this code?

```
class MyException extends Throwable{}
class MyException1 extends MyException{}
class MyException2 extends MyException{}
class MyException3 extends MyException2{}
public class ExceptionTest{
   void myMethod() throws MyException{
      throw new MyException3();
   }
   public static void main(String[] args){
      ExceptionTest et = new ExceptionTest();
      try{
         et.myMethod();
      }
      catch(MyException me){
         System.out.println("MyException thrown");
      }
      catch(MyException3 me3){
         System.out.println("MyException3 thrown");
      }
      finally{
         System.out.println(" Done");
      }
   }
}
```

Correct Option is : E

~~**A.**~~ MyException thrown

~~**B.**~~ MyException3 thrown

~~**C.**~~ MyException thrown Done

~~**D.**~~ MyException3 thrown Done

E. It fails to compile

Explanation:

You can have multiple catch blocks to catch different kinds of exceptions, including exceptions that are subclasses of other exceptions. However, the catch clause for more specific exceptions (i.e. a "SubClassException") should come before the catch clause for more general exceptions (i.e. a "SuperClassException"). Failure to do so results in a compiler error as the more specific exception is unreachable.

In this case, catch for MyException3 cannot follow catch for MyException because if MyException3 is thrown, it will be caught by the catch clause for MyException. And so, there is no way the catch clause for MyException3 can ever execute. And so it becomes an "unreachable" statement.

Back to Question without Answer

67. QID - 2.945 : Working with Inheritance

Which of the following statements are true?

Correct Options are : A E

A. Private methods cannot be overridden in subclasses.

> Only methods that are inherited can be overridden and private methods are not inherited.

~~**B.**~~ A subclass can override any method in a non-final superclass.

> Only the methods that are not declared to be final can be overridden. Further, private methods are not inherited so they cannot be overridden either.

~~**C.**~~ An overriding method can declare that it throws a wider spectrum of checked exceptions than the method it is overriding.

~~**D.**~~ The parameter list of an overriding method must be a subset of the parameter list of the method that it is overriding.

> An overriding method (the method that is trying to override the base class's method) must have the same parameters.

E. The overriding method may opt not to declare any throws clause even if the original method has a throws clause.

> No exception (i.e. an empty set of exceptions) is a valid subset of the set of exceptions thrown by the original method so an overriding method can choose to not have any throws clause.

Explanation:

A method can be overridden by defining a method with the same signature(i.e. name and parameter list) and return type as the method in a superclass. The return type can also be a subclass of the orginal method's return type.

Only methods that are accessible can be overridden. A private method cannot, therefore, be overridden in subclasses, but the subclasses are allowed to define a new method with exactly the same signature.

A final method cannot be overridden.

An overriding method cannot exhibit behavior that contradicts the declaration of the original method. An overriding method therefore cannot return a different type (except a subtype) or throw a wider spectrum of exceptions than the original method in the superclass.

A subclass may have a static method with the same signature as a static method in the base class but it is not called overriding. It is called shadowing because the concept of polymorphism doesn't apply to static members.

Back to Question without Answer

68. QID - 2.848 : Using Loop Constructs

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        do{
            System.out.println(k);
        }while(--k>0);
    }
}
```

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 1
0

C. 2
1

> `--k>0` implies, decrement the value of k and then compare with 0. Therefore, the loop will only execute twice, printing 2 and 1.
>
> Had it been `k-->0`, it would imply, first compare k with 0, and then decrement k. In this case, the loop would execute thrice, printing 2, 1, and 0.

~~**D.**~~ 2
1

0

~~**E.**~~ It will keeping printing numbers in an infinite loop.

~~**F.**~~ It will not compile.

Back to Question without Answer

69. QID - 2.1079 : Java Basics

Consider the following two classes defined in two .java files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1  <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       System.out.println(X.LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Correct Option is : D

A. `import static X;`

B. `import static com.foo.*;`

Bad syntax. Package import does not use static keyword.

C. `import static com.foo.X.*;`

This static import, although syntactically correct, will not help here because Y

is accessing class X in X.LOGICID.

D. `import com.foo.*;`

This is required because Y is accessing class X. static import of LOGICID is NOT required because Y is accessing LOGICID through X (X.LOGICID). Had it been just System.out.println(LOGICID), only one import statement: import static com.foo.X.*; would have worked.

E. `import com.foo.X.LOGICID;`

Bad Syntax. Syntax for importing static fields is: import static <package>.<classname>.*; or import static <package>.<classname>.<fieldname>;

Back to Question without Answer

70. QID - 2.1363 : Working with Java Data Types - String, StringBuilder

In java, Strings are immutable. A direct implication of this is...

Correct Options are : A B

A. you cannot call methods like "1234".replace('1', '9'); and expect to change the original String.

> calling such methods do not change this object. They create a new String object.

B. you cannot change a String object, once it is created.

~~**C.**~~ you can change a String object only by the means of its methods.

~~**D.**~~ you cannot extend String class.

> That's because it is final, not because it is immutable. You can have a final class whose objects are mutable.

~~**E.**~~ you cannot compare String objects.

> String class implements Comparable interface.

Back to Question without Answer

71. QID - 2.1135 : Working with Inheritance

Which of the following statements is/are true?

Correct Option is : C

~~**A.**~~ Subclasses must define all the abstract methods that the superclass defines.

> Not if the subclass is also defined abstract!

~~**B.**~~ A class implementing an interface must define all the methods of that interface.

> Not if the class is defined abstract.

C. A class cannot override the super class's constructor.

> Because constructors are not inherited.

~~**D.**~~ It is possible for two classes to be the superclass of each other.

~~**E.**~~ An interface can implement multiple interfaces.

> Interface cannot "implement" anything. It can extend multiple interfaces. The following is a valid declaration :
> interface I1 extends I2, I3, I4 { }

Back to Question without Answer

72. QID - 2.987 : Creating and Using Arrays

What will be the result of attempting to run the following program?

```
public class StringArrayTest{
   public static void main(String args[]){
      String[][][] arr  ={{ { "a", "b" , "c"}, { "d", "e", null } },
{ {"x"}, null },{{"y"}},{ { "z","p"}, {} }
      };
      System.out.println(arr[0][1][2]);
   }
}
```

Correct Option is : C

~~**A.**~~ It will throw NullPointerBoundsException.

> There is no such exception.

~~**B.**~~ It will throw ArrayIndexOutOfBoundsException.

C. It will print null.

~~**D.**~~ It will run without any error but will print nothing.

~~**E.**~~ None of the above.

Explanation:

arr[0][1][2] => [0] = { { "a", "b" , "c"}, { "d", "e", null } }, [1] = { "d", "e", null } and [2] = null.
So it will print null.

Back to Question without Answer

73. QID - 2.894 : Java Basics

The following are the complete contents of TestClass.java file. Which packages are automatically imported?

```
class TestClass{
   public static void main(String[] args){
     System.out.println("hello");
   }
}
```

Correct Options are : C F

~~**A.**~~ java.util

~~**B.**~~ System

> System is not a package. It is a class in java.lang package.

C. java.lang

~~**D.**~~ java.io

~~**E.**~~ String

> String is a class in java.lang package.

F. The package with no name.

> If there is no package statement in the source file, the class is assumed to be created in a default package that has no name.

Back to Question without Answer

74. QID - 2.1012 : Constructors

Which lines contain a valid constructor in the following code?

```
public class TestClass{
   public TestClass(int a, int b) { } // 1
   public void TestClass(int a) { }   // 2
   public TestClass(String s); // 3
   private TestClass(String s, int a) { }     //4
   public TestClass(String s1, String s2) { }; //5
}
```

Correct Options are : A D E

A. Line // 1

~~**B.**~~ Line // 2

> Constructors cannot return anything. Not even void.

~~**C.**~~ Line // 3

> Constructors cannot have empty bodies (i.e. they cannot be abstract)

D. Line // 4

> You can apply public, private, protected to a constructor. But not static, final, synchronized, native and abstract.

E. Line // 5

> The compiler ignores the extra semi-colon.

Explanation:

It is interesting to note that `public void TestClass(int a) {} // 2` will actually compile. It is not a constructor, but compiler considers it as a valid method!

Back to Question without Answer

75. QID - 2.1279 : Using Loop Constructs

Which of the following code snippets will compile without any errors?
(Assume that the statement `int x = 0;` exists prior to the statements below.)

Correct Options are : B C D

~~**A.**~~ `while (false) { x=3; }`

B. `if (false) { x=3; }`

C. `do{ x = 3; } while(false);`

> In a do- while, the block is ALWAYS executed at least once because the condition check is done after the block is executed. Unlike a while loop, where the condition is checked before the execution of the block.

D. `for( int i = 0; i< 0; i++) x = 3;`

Explanation:

while `(false) { x=3; }` is a compile-time error because the statement `x=3;` is not reachable;
Similarly, `for( int i = 0; false; i++) x = 3;` is also a compile time error because `x= 3` is unreachable.

In `if(false){ x=3; },` although the body of the condition is unreachable, this is not an error because the JLS explicitly defines this as an exception to the rule. It allows this construct to support optimizations through the conditional compilation. For example,

```
if(DEBUG){ System.out.println("beginning task 1"); }
```

Here, the DEBUG variable can be set to false in the code while generating the production version of the class file, which will allow the compiler to optimize the code by removing the whole if statement entirely from the class file.

Back to Question without Answer

76. QID - 2.1098 : Constructors

What will be the result of attempting to compile the following program?

```
public class TestClass{
   long l1;
   public void TestClass(long pLong) { l1 = pLong ; }  //(1)
   public static void main(String args[]){
      TestClass a, b ;
      a = new TestClass();  //(2)
      b = new TestClass(5);  //(3)
   }
}
```

Correct Option is : C

~~**A.**~~ A compilation error will be encountered at (1), since constructors should not specify a return value.

> But it becomes a valid method if you give a return type.

~~**B.**~~ A compilation error will be encountered at (2), since the class does not have a default constructor.

> The class has an implicit default constructor since the class doesn't have any constructor defined.

C. A compilation error will be encountered at (3).

> Because (1) is a method and not a constructor. So there is no constructor that take a parameter.

~~**D.**~~ The program will compile correctly.

E. It will not compile because parameter type of the constructor is different than the type of value passed to it.

> If (1) was a valid constructor 'int' would be promoted to long at the time of passing.

Explanation:

The declaration at `(1)` declares a method, not a constructor because it has a return value. The method happens to have the same name as the class, but that is ok.
The class has an implicit default constructor since the class contains no constructor declarations. This allows the instantiation at `(2)` to work.

Back to Question without Answer

77. QID - 2.998 : Working with Inheritance

What will the following program print when compiled and run?

```
class Game{
  public void play() throws Exception{
    System.out.println("Playing...");
  }
}

public class Soccer extends Game{
   public void play(){
      System.out.println("Playing Soccer...");
   }
   public static void main(String[] args){
       Game g = new Soccer();
       g.play();
   }
}
```

Correct Option is : A

A. It will not compile.

~~**B.**~~ It will throw an Exception at runtime.

~~**C.**~~ Playing Soccer...

~~**D.**~~ Playing...

~~**E.**~~ None of these.

Explanation:

Observe that play() in Game declares Exception in its throws clause. Further, class Soccer overrides the play() method without any throws clause. This is valid because a list of no exception is a valid subset of a list of exceptions thrown by the superclass method.
Now, even though the actual object referred to by 'g' is of class Soccer, the class of the variable g is of class Game. Therefore, at compile time, compiler assumes that g.play() might throw an exception, because Game's play method declares it, and thus expects this call to be either wrapped in a try-catch or the main method to have a throws clause for the main() method.

Back to Question without Answer

78. QID - 2.1016 : Using Loop Constructs

What will the following code snippet print?

```
int count = 0, sum = 0;
do{
        if(count % 3 == 0) continue;
        sum+=count;
}
while(count++ < 11);
System.out.println(sum);
```

Correct Option is : B

~~**A.** 49~~

B. 48

~~**C.** 37~~

~~**D.** 36~~

~~**E.** 38~~

Explanation:

1. The while condition uses post increment operator, which means count is first compared with 11 (and based on this comparison a decision is made whether to execute the loop again or not) and then incremented. So when count is 10, the condition 10<11 is true (that means the loop needs to be executed again) and count is

incremented to 11.

2. When count is completely divisible by 3, (i.e. when count is 0, 3, 6, 9) `sum+=count;` is not executed.

Thus, the result is the summation of:
1 2 4 5 7 8 10 11

Back to Question without Answer

79. QID - 2.1107 : Working with Inheritance

Which of the following are correct about "encapsulation"?

Correct Options are : B C

~~**A.**~~ Encapsulation is same as polymorphism.

B. It helps make sure that clients have no accidental dependence on the choice of representation

C. It helps avoiding name clashes as internal variables are not visible outside.

~~**D.**~~ Encapsulation makes sure that messages are sent to the right object at run time.

This is dynamic binding, an outcome of polymorphism.

~~**E.**~~ Encapsulation helps you inherit the properties of another class.

Explanation:

Encapsulation: Encapsulation is the technique used to package the information in such a way as to hide what should be hidden, and make visible what is intended to be visible. In simple terms, encapsulation generally means making the data variables private and providing public accessors.

Back to Question without Answer

80. QID - 2.1298 : Java Basics

Which of the following are valid declarations within a class?

Correct Options are : A E

A. `volatile int k;`

~~B.~~ `abstract boolean bool;`

fields cannot be abstract.

~~C.~~ `native float radix;`

variables cannot be native.

~~D.~~ `friendly int ArrayList;`

Here friendly is being used as if it were a keyword, but friendly is not a valid keyword in Java.

E. `int friendly, ArrayList;`

Here, friendly and ArrayList are variable names. Note that any class name can be used as a variable name.

Back to Question without Answer

81. QID - 2.1270 : Using Loop Constructs

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      boolean b = false;
      int i = 1;
      do{
         i++ ;
      } while (b = !b);
      System.out.println( i );
   }
}
```

Correct Option is : C

~~**A.**~~ The code will fail to compile, 'while' has an invalid condition expression.

> It is perfectly valid because b = !b; returns a boolean, which is what is needed for while condition.

~~**B.**~~ It will compile but will throw an exception at runtime.

C. It will print 3.

> The loop body is executed twice and the program will print 3.

~~**D.**~~ It will go in an infinite loop.

~~**E.**~~ It will print 1.

Explanation:

Unlike the 'while(){}' loop, the 'do {} while()' loop executes at least once because the condition is checked after the iteration.

Back to Question without Answer

82. QID - 2.1065 : Using Operators and Decision Constructs

What will the following code print?

```
boolean flag = true;
if(flag = false){
   System.out.println("1");
}else if(flag){
   System.out.println("2");
}else if(!flag){
   System.out.println("3");
}else    System.out.println("4");
```

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 2

C. 3

~~**D.**~~ 4

~~**E.**~~ Compilation error.

Explanation:

At the beginning, flag is true. In the first if, we do flag = false. Notice that it is not flag == false. It is a single =, which assigns false to flag. Thus, flag becomes false and the condition becomes false therefore 1 is not printed. In the first 'else if', again since flag is false, 2 is not printed. In second 'else if', !flag implies !false, which is true, so 3 is

printed. Finally, since an else-if condition has been satisfied, the last else is not executed.

Back to Question without Answer

83. QID - 2.1134 : Working with Methods

Which of the following are valid at line 1?

```
public class X{
   //line 1: insert code here.
}
```

Correct Options are : A D

A. String s;

~~**B.**~~ String s = 'asdf';

A string must be enclosed in double quotes ".

~~**C.**~~ String s = 'a';

'a' is a char. "a" is a String.

D. String s = this.toString();

Since every class directly or indirectly extends Object class and since Object class has a toString() method, that toString() method will be invoked and the String that it returns will be assigned to s.

~~**E.**~~ String s = asdf;

there is no variable asdf defined in the given class.

Back to Question without Answer

84. QID - 2.1208 : Working with Inheritance

Given the following class definition:

```
class A{
  protected int i;
  A(int i) {    this.i = i;    }

}
// 1 : Insert code here
```

Which of the following would be a valid class that can be inserted at //1 ?

Correct Options are : A D

A. `class B {}`

~~**B.**~~ `class B extends A {}`

Since class B does not have any constructor, the compiler will try to insert the default constructor, which will look like this:

```
B(){
    super();  //Notice that it is trying to call the no args
constructor of the super class, A.
}
```

Since A doesn't have any no-args constructor, the above code will fail to compile.

~~**C.**~~ `class B extends A {  B()  {  System.out.println("i = " + i); }  }`

It has the same problem as the one above.

D. `class B { B() {} }`

Explanation:

Notice that class A does not define a no-argument constructor. Also note that the class B does not define a constructor. Thus, class B relies on the default constructor B(). Class B's default constructor looks like this: public B() {} However, Constructors implicitly (if an explicit call to the superclass's constructor is not present) call their superclass's constructor super(). So, class B's default constructor actually looks like this:

```
public B(){
  super();
}
```

Now, since class A does not define a no-argument constructor the above code will not compile. However, class B would be correct if changed to:

```
class B extends A{
  B(){
    super(1); // pass it any integer
  }
  // or
  B(int number){
    super(number);
  }
}
```

You could also add a no-argument constructor to class A and leave class B as is.

Back to Question without Answer

85. QID - 2.1083 : Using Loop Constructs

What will the following program print?

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

Correct Option is : B

~~**A.** 2~~

B. 3

~~**C.** 6~~

~~**D.** 7~~

~~E.~~ 9

Explanation:

To understand how this loop works let us put some extra print statements in the innermost loop:

```
System.out.println("i="+i+" j="+j+" k="+k);
if(k-j>0){
     System.out.println("breaking middle "+j);
     break middle;
}
counter++;
```

This is what it prints:

```
i=0 j=0 k=0
i=0 j=0 k=1
breaking middle 0
i=1 j=0 k=0
i=1 j=0 k=1
breaking middle 0
i=2 j=0 k=0
i=2 j=0 k=1
breaking middle 0
3
```

The key is that the middle loop is broken as soon as k-j becomes > 0. This happens on every second iteration of inner loop when k is 1 and j is 0. Now, when middle is broken inner cannot continue. So the next iteration of outer starts.

Back to Question without Answer

86. QID - 2.964 : Handling Exceptions

What letters, and in what order, will be printed when the following program is compiled and run?

```
public class FinallyTest{
   public static void main(String args[]) throws Exception{
       try{
          m1();
          System.out.println("A");
       }
       finally{
          System.out.println("B");
       }
       System.out.println("C");
   }
   public static void m1() throws Exception { throw new Exception();
}
```

Correct Option is : C

~~**A.**~~ It will print C and B, in that order.

~~**B.**~~ It will print A and B, in that order.

C. It will print B and throw Exception.

~~**D.**~~ It will print A, B and C in that order.

~~**E.**~~ Compile time error.

Explanation:

An `Exception` is thrown in method `m1()` so `println("A")` will not be executed. As there is no catch block the exception will not be handled and the `main()` method will propagate the exception. So `println("C");` will also not be executed. 'finally' block is always executed (even if there is a return in try but not if there is System.exit()) so `println("B")` is executed.

Back to Question without Answer

87. QID - 2.887 : Overloading methods

Given:

```
class OverloadingTest{

    void m1(int x){
        System.out.println("m1 int");
    }

    void m1(double x){
        System.out.println("m1 double");
    }

    void m1(String x){
        System.out.println("m1 String");
    }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        OverloadingTest ot = new OverloadingTest();
        ot.m1(1.0);
    }
}
```

What will be the output?

Correct Option is : C

~~**A.**~~ It will fail to compile.

~~**B.**~~ m1 int

C. m1 double

~~**D.**~~ m1 String

Explanation:

Here, `m1()` is overloading for three different argument types. So when you call `ot.m1(1.0)`, the one with argument of type double will be invoked.

Back to Question without Answer

88. QID - 2.890 : Working with Java Data Types - Variables and Objects

Given:

```
class Square {
    private double side = 0;
    String color;
    public Square(double length){
        this.side = length;
    }
    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Square mysq = new Square(10);
        mysq.color = "red";

        //set mysq's side to 20
    }
}
```

Which of the following statements will set mysq's side to 20?

Correct Option is : C

~~**A.**~~ `mysq.side = 20;`

> Since side is a private variable, you cannot access it from outside Square class.

~~**B.**~~ `mysq = new Square(20);`

> This will create a new Square object.

C. `mysq.setSide(20);`

~~D.~~ `side = 20;`

~~E.~~ `Square.mysql.side = 20;`

Back to Question without Answer

89. QID - 2.1271 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following code?

```
public class PromotionTest{
   public static void main(String args[]){
      int i = 5;
      float f = 5.5f;
      double d = 3.8;
      char c = 'a';
      if (i == f) c++;
      if (((int) (f + d)) == ((int) f + (int) d)) c += 2;
      System.out.println(c);
   }
}
```

Correct Option is : E

~~**A.**~~ The code will fail to compile.

~~**B.**~~ It will print d.

~~**C.**~~ It will print c.

~~**D.**~~ It will print b

E. It will print a.

Explanation:

In the case of i == f, value of i will be promoted to a float i.e. 5.0, and so it returns false.
(int)f+(int)d = (int)5.5 + (int) 3.8 => 5 + 3 = 8

(int)(f + d) => (int) (5.5 + 3.8) => (int)(9.3) => 9, so this also return false.
So, c is not incremented at all. Hence c remains 'a'.

Back to Question without Answer

90. QID - 2.842 : Handling Exceptions

A new Java programmer has written the following method that takes an array of integers and sums up all the integers that are less than 100.

```
public void processArray(int[] values){
        int sum = 0;
        int i = 0;
        try{
            while(values[i]<100){
                sum = sum +values[i];
                i++;
            }
        }
        catch(Exception e){ }
        System.out.println("sum = "+sum);
    }
```

Which of the following are best practices to improve this code?

Correct Options are : B D

~~**A.**~~ Use `ArrayIndexOutOfBoundsException` for the catch argument.

B. Use `ArrayIndexOutOfBoundsException` for the catch argument and add code in the catch block to log or print the exception.

> Empty catch blocks are a bad practice because at run time, if the exception is thrown, the program will not show any sign of the exception and may produce bad results that will be hard to debug. Therefore, it is a good practice to at least print out the exception if you don't want to do any thing upon encountering an exception.

~~**C.**~~ Add code in the catch block to handle the exception.

There are a few questions in the exam that are difficult to interpret. In this case, for example, it is not clear what is meant by handling the exception. The catch block itself is meant to handle the exception. Once you get the exception, you can do what ever is required in the catch block.

D. Use flow control to terminate the loop.

It is considered a bad practice to use exceptions to control the flow of execution. In this case, `values[i]` will throw an `ArrayIndexOutOfBoundsException` once it goes beyond the array length and the programmer is using this fact to control the loop. Instead of doing this, the programmer should use something like: `for(int i=0; i<values.length; i++)` to control the execution of the loop.

Back to Question without Answer

Test 2

01. QID - 2.1045

Which line, if any, will give a compile time error ?

```
void test(byte x){
   switch(x){
      case 'a':    // 1
      case 256:    // 2
      case 0:      // 3
      default :    // 4
      case 80:     // 5
   }
}
```

Select 1 option

A. Line 1 as 'a' is not compatible with byte.

B. Line 2 as 256 cannot fit into a byte.

C. No compile time error but a run time error at line 2.

D. Line 4 as the default label must be the last label in the switch statement.

E. There is nothing wrong with the code.

Check Answer

02. QID - 2.1276

What is wrong with the following code?

```
class MyException extends Exception {}
public class TestClass{
   public static void main(String[] args){
      TestClass tc = new TestClass();
      try{
         tc.m1();
      }
      catch (MyException e){
         tc.m1();
      }
      finally{
         tc.m2();
      }
   }
   public void m1() throws MyException{
      throw new MyException();
   }
   public void m2() throws RuntimeException{
      throw new NullPointerException();
   }
}
```

Select 1 option

A. It will not compile because you cannot throw an exception in finally block.

B. It will not compile because you cannot throw an exception in catch block.

C. It will not compile because NullPointerException cannot be created this way.

D. It will not compile because of unhandled exception.

E. It will compile but will throw an exception when run.

Check Answer

03. QID - 2.1274

What will be the result of attempting to compile and run the following code?

```
class TestClass{
   public static void main(String args[] ){
      String str1 = "str1";
      String str2 = "str2";
      System.out.println( str1.concat(str2) );
      System.out.println(str1);
   }
}
```

Select 1 option

A. The code will fail to compile.

B. The program will print `str1` and `str1`.

C. The program will print `str1` and `str1str2`

D. The program will print `str1str2` and `str1`

E. The program will print `str1str2` and `str1str2`.

Check Answer

04. QID - 2.949

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2 = false;
if (b2 != b1 = !b2){
   System.out.println("true");
}
else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print `true`.

C. It will print `false`.

D. Runtime error.

E. It will print nothing.

Check Answer

05. QID - 2.873

Which of the following are benefits of ArrayList over an array?

Select 1 option

A. You do not have to worry about the size of the ArrayList while inserting elements.

B. It consumes less memory space.

C. You do not have to worry about thread safety.

D. It allows you to write type safe code.

Check Answer

06. QID - 2.923

Given the following code:

```
class M { }
class N{
   private M m = new M();
   public void makeItNull(M pM){
      pM = null;
   }
   public void makeThisNull(){
      makeItNull(m);
   }
   public static void main(String[] args){
      N n = new N();
      n.makeThisNull();
   }
}
```

Which of the following statements are correct?

Select 1 option

A. There are no instances of M to be garbage collected till the program ends.

B. A call to `n.makeThisNull()` marks the private instance of M for garbage collection.

C. Setting `pM = null;` in `makeItNull()`, marks the private instance of M for garbage collection.

D. `private` members of a class become eligible for garbage collection only when the instance of the class itself becomes eligible for garbage collection.

Check Answer

07. QID - 2.1263

Consider the following code in TestClass.java file:

```
package p;
private class TC extends java.util.HashMap{
   public TC(){
      super(100);
      System.out.println("TC created");
   }
}
public class TestClass extends TC{
   public TestClass(){
      System.out.println("TestClass created");
   }
   public static void main(String[] args){ new TestClass(); }
}
```

What will be the output when TestClass is run?

Select 1 option

A. `"TestClass created"` as well as `"TC created"`.

B. It will not compile because `HashMap` is a final class.

C. Only `"TestClass created"` will be printed.

D. Only `"TC created"` will be printed.

E. None of the above are correct.

Check Answer

08. QID - 2.1227

Given the following code, which of these statements are true?

```
class TestClass{
   public static void main(String args[]){
      int k = 0;
      int m = 0;
      for ( int i = 0; i <= 3; i++){
         k++;
         if ( i == 2){
            // line 1
         }
         m++;
      }
      System.out.println( k + ", " + m );
   }
}
```

Select 3 options

A. It will print 3, 2 when line 1 is replaced by break;

B. It will print 3, 2 when line 1 is replaced by continue.

C. It will print 4, 3 when line 1 is replaced by continue.

D. It will print 4, 4 when line 1 is replaced by i = m++;

E. It will print 3, 3 when line 1 is replaced by i = 4;

Check Answer

09. QID - 2.866

What can be the type of a `catch` argument ?

Select 1 option

A. Any class that extends `java.lang.Exception`

B. Any class that extends `java.lang.Exception` except any class that extends `java.lang.RuntimeException`

C. Any class that is-a `Throwable`.

D. Any Object

E. Any class that extends `Error`

Check Answer

10. QID - 2.1052

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10);        //1
        test1(10, 20); //2
   }

   public static void main(String[] args){
     new Varargs().test();
   }

   //insert method here.
}
```

Which of the following lines can be added independently to the above class so that it will run without any errors or exceptions?

Select 2 options

A. `public void test1(int i, int j){}`

B. `public void test1(int i, int... j){}`

C. `public void test1(int... i){}`

D. `public void test1(int i...){}`

E. `public void test1(int[] i){}`

Check Answer

11. QID - 2.1325

What will be the result of attempting to compile and run the following class?

```
public class IfTest{
   public static void main(String args[]){
      if (true)
      if (false)
      System.out.println("True False");
      else
      System.out.println("True True");
   }
}
```

Select 1 option

A. The code will fail to compile because the syntax of the `if` statement is not correct.

B. The code will fail to compile because the values in the condition bracket are invalid.

C. The code will compile correctly and will not display anything.

D. The code will compile correctly and will display `True True`.

E. The code will compile correctly but will display `True False`

Check Answer

12. QID - 2.958

Consider the following code:

```
public class Conversion{
   public static void main(String[] args){
     int i = 1234567890;
     float f = i;
     System.out.println(i - (int)f);
   }
}
```

What will it print when run?

Select 1 option

A. It will print 0.

B. It will not print 0.

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

13. QID - 2.1361

The following program will print

```
java.lang.ArithmeticException: / by zero
```

```
class Test{
   public static void main(String[] args){
      int d = 0;
      try{
         int i = 1 / (d* doIt());
      } catch (Exception e){
         System.out.println(e);
      }
   }
   public static int doIt() throws Exception{
      throw new Exception("Forget It");
   }
}
```

Select 1 option

A. True

B. False

Check Answer

14. QID - 2.1029

The following class will print 'index = 2' when compiled and run.

```
class Test{
   public static int[ ] getArray() {  return null;  }
   public static void main(String[] args){
      int index = 1;
      try{
         getArray()[index=2]++;
      }
      catch (Exception e){  }  //empty catch
      System.out.println("index = " + index);
   }
}
```

Select 1 option

A. True

B. False

Check Answer

15. QID - 2.1300

An abstract method cannot be overridden.

Select 1 option

A. True

B. False

Check Answer

16. QID - 2.1071

What will be the output of the following code snippet?

```
int a = 1;
int[] ia = new int[10];
int b = ia[a];
int c = b + a;
System.out.println(b = c);
```

Select 1 option

A. 0

B. 1

C. 2

D. true

E. false

Check Answer

17. QID - 2.1221

Which of the following is illegal ?

Select 1 option

A. char c = 320;

B. float f = 320;

C. double d = 320;

D. byte b = 320;

E. None of the above is illegal.

Check Answer

18. QID - 2.982

Given the following class, which of the given blocks can be inserted at line 1 without errors?

```
public class InitClass{
       private static int loop = 15 ;
       static final int INTERVAL = 10 ;
       boolean flag ;
       //line 1
}
```

Select 4 options

A. `static {System.out.println("Static"); }`

B. `static { loop = 1; }`

C. `static { loop += INTERVAL; }`

D. `static { INTERVAL = 10; }`

E. `{ flag = true; loop = 0; }`

Check Answer

19. QID - 2.836

Which of the following statements are correct?

Select 3 options

A. An abstract class can be extended by an abstract or a concrete class.

B. A concrete class can be extended by an abstract or a concrete class.

C. An interface can be extended by another interface.

D. An interface can be extended by an abstract class.

E. An interface can be extended by a concrete class.

F. An abstract class cannot implement an interface.

Check Answer

20. QID - 2.1130

You want to invoke the overridden method (the method in the base class) from the overriding method (the method in the derived class) named `m()`.
Which of the following constructs which will let you do that?

Select 1 option

A. `super.m();`

B. `super.this();`

C. `base.m();`

D. `parent.m();`

E. `super();`

Check Answer

21. QID - 2.1209

Which statements concerning the following code are true?

```
class A{
  public A() {} // A1
  public A(String s) {  this();  System.out.println("A :"+s);  }  //
}

class B extends A{
  public int B(String s) {  System.out.println("B :"+s);  return 0;
}
class C extends B{
    private C(){ super(); } // C1
    public C(String s){  this();  System.out.println("C :"+s);  } //
    public C(int i){} // C3
}
```

Select 4 options

A. At least one of the constructors of each class is called as a result of constructing an object of class C.

B. Constructor at //A2 will never be called in creation of an object of class C

C. Class C can be instantiated only in two ways by users of this class.

D. //B1 will never be called in creation of objects if class A, B, or C

E. The code will not compile.

Check Answer

22. QID - 2.1127

Consider the following class and interface definitions (in separate files):

```
public class Sample implements IInt{
 public static void main(String[] args){
  Sample s = new Sample();  //1
  int j = s.thevalue;       //2
  int k = IInt.thevalue;    //3
  int l = thevalue;         //4
 }
}
public interface IInt{
 int thevalue = 0;
}
```

What will happen when the above code is compiled and run?

Select 1 option

A. It will give an error at compile time at line //1.

B. It will give an error at compile time at line //2.

C. It will give an error at compile time at line //3

D. It will give an error at compile time at line //4.

E. It will compile and run without any problem.

Check Answer

23. QID - 2.1104

Consider the following lines of code:

```
Integer i = new Integer(42);
Long ln = new Long(42);
Double d = new Double(42.0);
```

Which of the following options are valid?

Select 3 options

A. `i == ln;`

B. `ln == d;`

C. `i.equals(d);`

D. `d.equals(ln);`

E. `ln.equals(42);`

Check Answer

24. QID - 2.1141

Which line(s) in the following code will cause a compilation error?

```
static import java.lang.System.*; //1
class $$ //2
{
    static public void main(String... _$_) //3
   {
       String _ = ""; //4
       for(int $=0; ++$ < _$_.length; )  //5
           _ += _$_[$];  //6
       out.println(_);  //7
    }
}
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. 5

F. 6

G. 7

H. None of the lines is invalid.

Check Answer

25. QID - 2.1041

Which of the following is not a primitive data value in Java?

Select 2 options

A. `"x"`

B. `'x'`

C. `10.2F`

D. `Object`

E. `false`

Check Answer

26. QID - 2.1032

What will the following code print?

```
public class BreakTest{
  public static void main(String[] args){
    int i = 0, j = 5;
    lab1 : for( ; ; i++){
      for( ; ; --j)  if( i >j ) break lab1;
    }
    System.out.println(" i = "+i+", j = "+j);
  }
}
```

Select 1 option

A. `i = 1, j = -1`

B. `i = 1, j = 4`

C. `i = 0, j = 4`

D. `i = 0, j = -1`

E. It will not compile.

Check Answer

27. QID - 2.1049

Consider that you are writing a set of classes related to a new Data Transmission Protocol and have created your own exception hierarchy derived from java.lang.Exception as follows:

```
enthu.trans.ChannelException
              +-- enthu.trans.DataFloodingException,
                    enthu.trans.FrameCollisionException
```

You have a TransSocket class that has the following method:

```
long connect(String ipAddr) throws ChannelException
```

Now, you also want to write another "AdvancedTransSocket" class, derived from "TransSocket" which overrides the above mentioned method. Which of the following are valid declaration of the overriding method?

Select 2 options

A. `int connect(String ipAddr) throws DataFloodingException`

B. `int connect(String ipAddr) throws ChannelException`

C. `long connect(String ipAddr) throws FrameCollisionException`

D. `long connect(String ipAddr) throws Exception`

E. `long connect(String str)`

Check Answer

28. QID - 2.1053

Compared to public, protected and private accessibility, default accessibility is....

Select 1 option

A. Less restrictive than public

B. More restrictive than public, but less restrictive than protected.

C. More restrictive than protected, but less restrictive than private.

D. More restrictive than private.

E. Less restrictive than protected from within a package, and more restrictive than protected from outside a package.

Check Answer

29. QID - 2.1118

Which of the changes given in options can be done (independent of each other) to let the following code compile and run without errors?

```
class SomeClass{
   String s1 = "green mile";   // 0
   public void generateReport( int n ){
      String local;   // 1
      if( n > 0 ) local = "good";   //2
      System.out.println( s1+" = " + local );   //3
   }
}
```

Select 2 options

A. Insert after line 2 : `else local = "bad";`

B. Insert after line 2 : `if(n <= 0) local = "bad";`

C. Move line 1 and place it after line 0.

D. change line 1 to : `final String local = "rocky";`

E. The program already is without any errors.

Check Answer

30. QID - 2.1293

Which of the following correctly defines a method named stringProcessor that can be called by other programmers as follows: `stringProcessor(str1)` or `stringProcessor(str1, str2)` or `stringProcessor(str1, str2, str3)`, where `str1`, `str2`, and `str3` are references to Strings.

Select 1 option

A.
```
public void stringProcessor(...String){
}
```

B.
```
public void stringProcessor(String... strs){
}
```

C.
```
public void stringProcessor(String[] strs){
}
```

D.
```
public void stringProcessor(String a, String b, String c){
}
```

E. Three separate methods need to be written.

Check Answer

31. QID - 2.994

Assume that a, b, and c refer to instances of primitive wrapper classes. Which of the following statements are correct?

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Select 2 options

A. a.equals(a) will always return true.

B. b.equals(c) may return false even if c.equals(b) returns true.

C. a.equals(b) returns same as a == b.

D. a.equals(b) throws an exception if they refer to instances of different classes.

E. a.equals(b) returns false if they refer to instances of different classes.

Check Answer

32. QID - 2.858

Which of the following are true about the "default" constructor?

Select 1 option

A. It is provided by the compiler only if the class and any of its super classes does not define any constructor.

B. It takes no arguments.

C. A default constructor is used to return a default value.

D. To define a default constructor, you must use the default keyword.

E. It is always public.

Check Answer

33. QID - 2.972

Consider the following program :

```
class Test{
  public static void main(String[] args){
    short s = 10;    // 1
    char c = s;      // 2
    s = c;           // 3
  }
}
```

Identify the correct statements.

Select 2 options

A. Line 3 is not valid.

B. Line 2 is not valid.

C. It will compile because both short and char can hold 10.

D. None of the lines 1, 2 and 3 is valid.

Check Answer

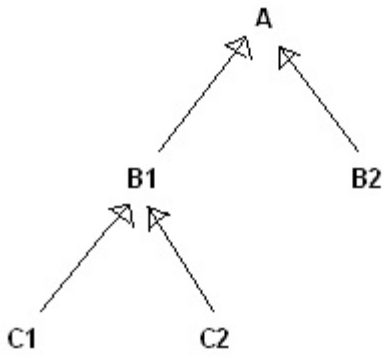
34. QID - 2.1287

Consider the following class hierarchy shown in the image. (B1 and B2 are subclasses of A and C1, C2 are subclasses of B1)

Assume that method `public void m1(){ ... }` is defined in all of these classes EXCEPT B1 and C1.

Assume that `"objectOfXX"` means a variable that points to an object of class XX. So, `objectOfC1` means a reference variable that is pointing to an object of class C1.

Which of the following statements are correct?



Select 1 option

A. `objectOfC1.m1();` will cause a compilation error.

B. `objectOfC2.m1();` will cause A's `m1()` to be called.

C. `objectOfC1.m1();` will cause A's `m1()` to be called.

D. `objectOfB1.m1();` will cause an exception at runtime.

E. `objectOfB2.m1();` will cause an exception at runtime.

Check Answer

35. QID - 2.999

Which of the following method calls can be applied to a String object?

Select 3 options

A. `equals(Object)`

B. `equalsIgnoreCase(String)`

C. `prune()`

D. `append()`

E. `intern()`

Check Answer

36. QID - 2.1179

The following code snippet will print 'true'.

```
short s = Short.MAX_VALUE;
char c = s;
System.out.println( c == Short.MAX_VALUE);
```

Select 1 option

A. True

B. False

Check Answer

37. QID - 2.1239

What letters will be printed by this program?

```
public class ForSwitch{
    public static void main(String args[]){
        char i;
        LOOP: for (i=0;i<5;i++){
            switch(i++){
                case '0': System.out.println("A");
                case 1: System.out.println("B"); break LOOP;
                case 2: System.out.println("C"); break;
                case 3: System.out.println("D"); break;
                case 4: System.out.println("E");
                case 'E' : System.out.println("F");
            }
        }
    }
}
```

Select 2 options

A. A

B. B

C. C

D. D

E. F

Check Answer

38. QID - 2.1358

What will the following program print when run using the command line: `java TestClass`

```
public class TestClass {

   public static void methodX() throws Exception {
      throw new AssertionError();
   }

   public static void main(String[] args) {
      try{
         methodX();
      }
      catch(Exception e) {
        System.out.println("EXCEPTION");
      }
   }
}
```

Select 1 option

A. It will throw `AssertionError` out of the main method.

B. It will print `EXCEPTION`.

C. It will not compile because of the throws clause in `methodX()`.

D. It will end without printing anything because assertions are disabled by default.

Check Answer

39. QID - 2.1351

Which of the following methods modify the object on which they are called?

Select 1 option

A. `setValue(int)` of java.lang.Integer class.

B. The `substring(int)` method of the String class

C. The `replace()` method of the String class.

D. The `reverse()` method of the StringBuffer class.

E. None of these.

Check Answer

40. QID - 2.1196

Consider the following code snippet:

```
XXXX m ;
  switch( m ){
     case 32  : System.out.println("32");   break;
     case 64  : System.out.println("64");   break;
     case 128 : System.out.println("128");  break;
  }
```

What type can 'm' be of so that the above code compiles and runs as expected ?

Select 3 options

A. `int m;`

B. `long m;`

C. `char m;`

D. `byte m;`

E. `short m;`

Check Answer

41. QID - 2.922

In the following code, after which statement (earliest), the object originally held in s, may be garbage collected ?

```
1. public class TestClass{
2.   public static void main (String args[]){
3.      Student s = new Student("Vaishali", "930012");
4.      s.grade();
5.      System.out.println(s.getName());
6.      s = null;
7.      s = new Student("Vaishali", "930012");
8.      s.grade();
9.      System.out.println(s.getName());
10      s = null;
     }
   }

public class Student{
   private String name, rollNumber;

   public Student(String name, String rollNumber) {
      this.name = name;
      this.rollNumber = rollNumber;
   }

   //valid setter and getter for name and rollNumber follow

   public void grade() {
   }

}
```

Select 1 option

A. It will not be Garbage Collected till the end of the program.

B. Line 5

C. Line 6

D. Line 7

E. Line 10

Check Answer

42. QID - 2.988

What will the following code print?

```
public class Test{
    public static void stringTest(String s){
        s.replace('h', 's');
    }
    public static void stringBuilderTest(StringBuilder s){
        s.append("o");
    }
    public static void main(String[] args){
        String s = "hell";
        StringBuilder sb = new StringBuilder("well");
        stringTest(s);
        stringBuilderTest(sb);
        System.out.println(s + sb);
    }
}
```

Select 1 option

A. sellwello

B. hellwello

C. hellwell

D. sellwell

E. None of these.

Check Answer

43. QID - 2.833

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showTwo(200);
        System.out.println(ct.myValue);
        ct.showOne(100);
        System.out.println(ct.myValue);
    }
}
```

Select 1 option

A. 0 followed by 100.

B. 100 followed by 100.

C. 0 followed by 200.

D. 100 followed by 200.

E. 200 followed by 200.

F. 200 followed by 100

Check Answer

44. QID - 2.1015

Which statement regarding the following code is correct?

```
class A{
   public int i = 10;
   private int j = 20;

}

class B extends A{
   private int i = 30; //1
   public int k = 40;

}

class C extends B{
}

public class TestClass{
   public static void main(String args[]){
      C c = new C();
      System.out.println(c.i); //2
      System.out.println(c.j); //3
      System.out.println(c.k);
   }
}
```

Select 1 option

A. It will print `10` and `40` if //3 is commented.

B. It will print `40` if //2 and //3 are commented.

C. It will not compile because of `//1`.

D. It will compile if `//2` is commented.

E. None of these.

Check Answer

45. QID - 2.997

Consider the following class written by a novice programmer.

```
class Elliptical{
    public int radiusA, radiusB;
    public int sum = 100;

    public void setRadius(int r){
        if(r>99) throw new IllegalArgumentException();
        radiusA = r;
        radiusB = sum - radiusA;

    }
}
```

After some time, the requirements changed and the programmer now wants to make sure that radiusB is always (200 - radiusA) instead of (100 - radiusA) without breaking existing code that other people have written. Which of the following will accomplish his goal?

Select 1 option

A. `Make sum = 200;`

B. `Make sum = 200` and make it private.

C. `Make sum = 200` and make all fields (radiusA, radiusB, and sum) private.

D. Write another method `setRadius2(int r)` and `set radiusB` accordingly in this method.

E. His goal cannot be accomplished.

F. This class will not compile.

Check Answer

46. QID - 2.1324

What will happen when the following program is compiled and run?

```
public class SM{
   public String checkIt(String s){
      if(s.length() == 0 || s == null){
         return "EMPTY";
      }
      else return "NOT EMPTY";
    }

    public static void main(String[] args){
      SM a = new SM();
      a.checkIt(null);
    }
}
```

Select 1 option

A. It will print `EMPTY`.

B. It will print `NOT EMPTY`.

C. It will throw `NullPointerException`.

D. It will print `EMPTY` if || is replaced with |.

Check Answer

47. QID - 2.1009

Consider the following code:

```
class Super { static String ID = "QBANK"; }

class Sub extends Super{
   static { System.out.print("In Sub"); }
}
public class Test{
   public static void main(String[] args){
      System.out.println(Sub.ID);
   }
}
```

What will be the output when class Test is run?

Select 1 option

A. It will print `In Sub` and `QBANK`.

B. It will print `QBANK`.

C. Depends on the implementation of JVM.

D. It will not even compile.

E. None of the above.

Check Answer

48. QID - 2.1042

Consider the following method which is called with an argument of 7:

```
public void method1(int i){
   int j = (i*30 - 2)/100;

   POINT1 : for(;j<10; j++){
       boolean flag  = false;
       while(!flag){
 if(Math.random()>0.5) break POINT1;
       }
   }
  while(j>0){
     System.out.println(j--);
     if(j == 4) break POINT1;
   }
}
```

What will it print?
(Assume that Math.random() return a double between 0.0 and 1.0, not including 1.0)

Select 1 option

A. It will print `1` and `2`

B. It will print `1` to N where N is a random number.

C. It will not compile.

D. It will throw an exception at runtime.

Check Answer

49. QID - 2.838

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Flyer) System.out.println("f is a Flyer");
        if(e instanceof Bird) System.out.println("e is a Bird");
        if(b instanceof Bird) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Select 1 option

A. It will not compile.

B. It will throw an exception when run.

C. f is a Flyer
e is a Bird

D. f is a Flyer

E. e is a Bird

Check Answer

50. QID - 2.1341

What will be result of attempting to compile this class?

```
import java.util.*;
package test;
public class TestClass{
    public OtherClass oc = new OtherClass();
}
class OtherClass{
    int value;
}
```

Select 1 option

A. The class will fail to compile, since the class OtherClass is used before it is defined.

B. There is no problem with the code.

C. The class will fail to compile, since the class OtherClass must be defined in a file called OtherClass.java

D. The class will fail to compile .

E. None of the above.

Check Answer

51. QID - 2.1149

Given the following class definitions :

```
interface MyIface{};
class A {};
class B extends A implements MyIface{};
class C implements MyIface{};
```

and the following object instantiations:

```
A a = new A();
B b = new B();
C c = new C();
```

Which of the following assignments are legal at compile time?

Select 1 option

A. `b = c;`

B. `c = b;`

C. `MyIface i = c;`

D. `c = (C) b;`

E. `b = a;`

Check Answer

52. QID - 2.1025

Identify the correct constructs.

Select 1 option

A.

```
try {
  for( ;; );
}finally { }
```

B.

```
try {
  File f = new File("c:\a.txt");
} catch {  f = null; }
```

C.

```
int k = 0;
try {
  k = callValidMethod();
}
System.out.println(k);
catch {  k = -1; }
```

D.

```
try {
  try {
     Socket s = new ServerSocket(3030);
  }catch(Exception e) {
```

```
    s = new ServerSocket(4040);
  }
}
```

E.

```
try {
       s = new ServerSocket(3030);
}
catch(Exception t){ t.printStackTrace(); }
catch(IOException e) {
    s = new ServerSocket(4040);
}
catch(Throwable t){ t.printStackTrace();  }
```

F.

```
int x = validMethod();
try {
  if(x == 5) throw new IOException();
  else if(x == 6) throw new Exception();
}finally {
 x = 8;
}
catch(Exception e){ x = 9; }
```

Check Answer

53. QID - 2.962

Consider the following method...

```
public int setVar(int a, int b, float c) { ...}
```

Which of the following methods correctly overload the above method?

Select 2 options

A.
```
public int setVar(int a, float b, int c){
  return (int)(a + b + c);
}
```

B.
```
public int setVar(int a, float b, int c){
  return this(a, c, b);
}
```

C.
```
public int setVar(int x, int y, float z){
  return x+y;
}
```

D.
```
public float setVar(int a, int b, float c){
  return c*a;
}
```

E.
```
public float setVar(int a){
  return a;
}
```

Check Answer

54. QID - 2.1210

Consider the following code snippet:

```
void m1() throws Exception{
   try{
      // line1
   }
   catch (IOException e){
       throw new SQLException();
   }
   catch(SQLException e){
       throw new InstantiationException();
   }
   finally{
      throw new CloneNotSupportedException();   // this is not a Runt
   }
}
```

Which of the following statements are true?

Select 2 options

A. If IOException gets thrown at line1, then the whole method will end up throwing SQLException.

B. If IOException gets thrown at line1, then the whole method will end up throwing CloneNotSupportedException.

C. If IOException gets thrown at line1, then the whole method will end up throwing InstantiationException()

D. If no exception is thrown at line1, then the whole method will end up throwing

CloneNotSupportedException.

E. If SQLException gets thrown at line1, then the whole method will end up throwing InstantiationException()

Check Answer

55. QID - 2.1187

Which of the following methods can be called on a String object?

Select 3 options

A. `substring(int i)`

B. `substring(int i, int j)`

C. `substring(int i, int j, int k)`

D. `equals(Object o)`

Check Answer

56. QID - 2.1285

Which of the following operators can be used in conjunction with a String object?

Select 3 options

A. +

B. ++

C. +=

D. .

E. *

Check Answer

57. QID - 2.1180

The following code snippet will not compile:

```
int i = 10;
System.out.println( i<20 ? out1() : out2() );
```

Assume that out1 and out2 have method signature: public void out1(); and public void out2();

Select 1 option

A. True

B. False

Check Answer

58. QID - 2.1335

Consider the code shown below:

```
public class TestClass{
  public static int switchTest(int k){
     int j = 1;
     switch(k){
        case 1: j++;
        case 2: j++;
        case 3: j++;
        case 4: j++;
        case 5: j++;
        default : j++;
     }
     return j + k;
  }
  public static void main(String[] args){
     System.out.println( switchTest(4) );
  }
}
```

What will it print when compiled and run?

Select 1 option

A. 5

B. 6

C. 7

D. 8

E. 9

Check Answer

59. QID - 2.938

What will the following program print?

```
class Test{
   public static void main(String[] args){
      int i = 4;
      int ia[][][] = new int[i][i = 3][i];
      System.out.println( ia.length + ", " + ia[0].length+", "+ ia[0
   }
}
```

Select 1 option

A. It will not compile.

B. 3, 4, 3

C. 3, 3, 3

D. 4, 3, 4

E. 4, 3, 3

Check Answer

60. QID - 2.1223

Consider the following hierarchy of Exception classes :

```
java.lang.RuntimeException
  +-------- IndexOutOfBoundsException
                  +---------ArrayIndexOutOfBoundsException,
StringIndexOutOfBoundsException
```

Which of the following statements are correct for a method that can throw `ArrayIndexOutOfBounds` as well as `StringIndexOutOfBounds` Exceptions but does not have try catch blocks to catch the same?

Select 3 options

A. The method calling this method will either have to catch these 2 exceptions or declare them in its `throws` clause.

B. It is ok if it declares just `throws ArrayIndexOutOfBoundsException`

C. It must declare `throws ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException`

D. It is ok if it declares just `throws IndexOutOfBoundsException`

E. It does not need to declare any throws clause.

Check Answer

61. QID - 2.927

What will the following code print ?

```
class Test{
   public static void main(String[] args){
      int k = 1;
      int[] a = { 1 };
      k += (k = 4) * (k + 2);
      a[0] += (a[0] = 4) * (a[0] + 2);
      System.out.println( k + " , " + a[0]);
   }
}
```

Select 1 option

A. It will not compile.

B. 4 , 4

C. 25 , 25

D. 13 , 13

E. None of the above.

Check Answer

62. QID - 2.905

Given the following line of code:

```
List students = new ArrayList();
```

Identify the correct statement:

Select 1 option

A. The reference type is List and the object type is ArrayList.

B. The reference type is ArrayList and the object type is ArrayList.

C. The reference type is ArrayList and the object type is List.

D. The reference type is List and the object type is List.

Check Answer

63. QID - 2.1258

Consider the following code:

```
class Test{
  public static void main(String[] args){
    for (int i = 0; i < args.length; i++)   System.out.print(i == 0
? args[i] : " " + args[i]);
  }
}
```

What will be the output when it is run using the following command:

java Test good bye friend!

Select 1 option

A. good bye friend!

B. good good good

C. goodgoodgood

D. good bye

E. None of the above.

Check Answer

64. QID - 2.1054

What will be written to the standard output when the following program is run?

```
public class TrimTest{
   public static void main(String args[]){
      String blank  = " ";  // one space
      String line = blank + "hello" + blank + blank;
      line.concat("world");
      String newLine  =  line.trim();
      System.out.println((int)(line.length() + newLine.length()));
   }
}
```

Select 1 option

A. 25

B. 24

C. 23

D. 22

E. None of the above.

Check Answer

65. QID - 2.898

Given:

```
public class Employee{
    String name;
    public Employee(){
    }
}
```

Which of the following lines creates an Employee instance?

Select 1 option

A. `Employee e;`

B. `Employee e = new Employee();`

C. `Employee e = Employee.new();`

D. `Employee e = Employee();`

Check Answer

66. QID - 2.1148

What will the following code print?

```
public class Test{
   public int luckyNumber(int seed){
      if(seed > 10) return seed%10;
         int x = 0;
            try{
               if(seed%2 == 0) throw new Exception("No Even no.");
               else return x;
            }
            catch(Exception e){
               return 3;
            }
            finally{
               return 7;
            }
         }

         public static void main(String args[]){
            int amount = 100, seed = 6;
            switch( new Test().luckyNumber(6) ){
                case 3: amount = amount * 2;
                case 7: amount = amount * 2;
                case 6: amount = amount + amount;
                default :
            }
           System.out.println(amount);
        }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. 800

D. 200

E. 400

Check Answer

67. QID - 2.944

Which of the following are valid declarations:

Select 3 options

A. int a = b = c = 100;

B. int a, b, c; a = b = c = 100;

C. int a, b, c=100;

D. int a=100, b, c;

E. int a= 100 = b = c;

Check Answer

68. QID - 2.1022

What will be the output of the following program (excluding the quotes)?

```
public class SubstringTest{
   public static void main(String args[]){
      String String = "string isa string";
      System.out.println(String.substring(3, 6));
   }
}
```

Select 1 option

A. It will not compile.

B. "ing is"

C. "ing isa"

D. "ing " (There is a space after g)

E. None of the above.

Check Answer

69. QID - 2.1114

What will the following code print?

```
public class TestClass{
        int x = 5;
        int getX(){ return x; }

        public static void main(String args[]) throws Exception{
            TestClass tc = new TestClass();
            tc.looper();
            System.out.println(tc.x);
        }

        public void looper(){
            int x = 0;
            while( (x = getX()) != 0 ){
                for(int m = 10; m>=0; m--){
                    x = m;
                }
            }

       }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. It will print 0.

D. It will print 5.

E. None of these.

Check Answer

70. QID - 2.1037

What will the following code print when compiled and run?

```
class Base{
   void methodA(){
      System.out.println("base - MethodA");
   }
}

class Sub extends Base{
   public void methodA(){
      System.out.println("sub - MethodA");
   }
   public void methodB(){
      System.out.println("sub - MethodB");
   }
   public static void main(String args[]){
      Base b=new Sub(); //1
      b.methodA(); //2
      b.methodB(); //3
   }
}
```

Select 1 option

A. `sub - MethodA` and `sub - MethodB`

B. `base - MethodA` and `sub - MethodB`

C. Compile time error at `//1`

D. Compile time error at `//2`

E. Compile time error at `//3`

Check Answer

71. QID - 2.1338

What is the correct parameter specification for the standard main method?

Select 2 options

A. void

B. String[ ] args

C. Strings args[ ]

D. String args

E. String args[ ]

Check Answer

72. QID - 2.903

A java source file contains the following code:

```
interface I {
  int getI(int a, int b);
}

interface J{
    int getJ(int a, int b, int c);
}

abstract class MyIJ implements J , I { }

class MyI{
    int getI(int x, int y){ return x+y; }
}

interface K extends J{
    int getJ(int a, int b, int c, int d);
}
```

Identify the correct statements:

Select 1 option

A. It will fail to compile because of `MyIJ`

B. It will fail to compile because of `MyIJ` and `K`

C. It will fail to compile because of `K`

D. It will fail to compile because of `MyI` and `K`

E. It will fail to compile because of `MyIJ, K,` and `MyI`

F. It will compile without any error.

Check Answer

73. QID - 2.1318

Which of the following statements is correct?

Select 1 option

A. new, delete and goto are keywords in the Java language

B. try, catch and thrown are keywords in the Java language

C. static, unsigned and long are keywords in the Java language

D. exit, class and while are keywords in the Java language

E. return, goto and default are keywords in the Java language

Check Answer

74. QID - 2.1340

Given the following code, which method declarations can be inserted at line 1 without any problems?

```
public class OverloadTest{
    public int sum(int i1, int i2) { return i1 + i2; }
    // 1
}
```

Select 3 options

A. `public int sum(int a, int b) { return a + b; }`

B. `public int sum(long i1, long i2) { return (int) i1; }`

C. `public int sum(int i1, long i2) { return (int) i2; }`

D. `public long sum(long i1, int i2) { return i1 + i2; }`

E. `public long sum(int i1, int i2) { return i1 + i2; }`

Check Answer

75. QID - 2.1299

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int var = 20, i=0;
      do{
         while(true){
         if( i++ > var) break;
         }
      }while(i<var--);
      System.out.println(var);
   }
}
```

Select 1 option

A. 19

B. 20

C. 21

D. 22

E. It will enter an infinite loop.

Check Answer

76. QID - 2.1183

Given the following LOCs:

```
int rate = 10;
XXX amount = 1 - rate/100*1 - rate/100;
```

What can XXX be?

Select 1 option

A. only int or long

B. only long or double

C. only double

D. double or float

E. long or double but not int or float.

F. int, long, float or double

Check Answer

77. QID - 2.1170

Consider the classes shown below:

```
class A{
   public A() { }
   public A(int i) {   System.out.println(i );    }
}
class B{
   static A s1 = new A(1);
   A a = new A(2);
   public static void main(String[] args){
      B b = new B();
      A a = new A(3);
   }
   static A s2 = new A(4);
}
```

Which is the correct sequence of the digits that will be printed when B is run?

Select 1 option

A. 1 ,2 ,3 4.

B. 1 ,4, 2 ,3

C. 3, 1, 2, 4

D. 2, 1, 4, 3

E. 2, 3, 1, 4

Check Answer

78. QID - 2.955

An instance member ...

Select 2 options

A. can be a variable, a constant or a method.

B. is a variable or a constant.

C. belongs to the class.

D. belongs to an instance of the class.

E. is same as a local variable.

Check Answer

79. QID - 2.917

Given:

```
public class Square {
    private double side = 0;  // LINE 2

    public static void main(String[] args) {   // LINE 4
        Square sq = new Square();  // LINE 5
        side = 10;  // LINE 6
   }
}
```

What can be done to make this code compile and run?

Select 1 option

A. replace // LINE 2 with:

```
private int side = 0;
```

B. replace // LINE 2 with:

```
public int side = 0;
```

C. replace // LINE 5 with:

```
double sq = new Square();
```

D. replace // LINE 6 with:

```
sq.side = 10;
```

Check Answer

80. QID - 2.1133

Which of the following can be thrown using a throw statement?

Select 3 options

A. Event

B. Object

C. Throwable

D. Exception

E. RuntimeException

Check Answer

81. QID - 2.1262

Consider the following code snippet ...

```
boolean[] b1 = new boolean[2];
boolean[] b2 = {true , false};
System.out.println( "" + (b1[0] == b2[0]) + ", "+ (b1[1] ==
b2[1])  );
```

What will it print ?

Select 1 option

A. It will not compile.

B. It will throw ArrayIndexOutOfBoundsError at Runtime.

C. false, true

D. true, false

E. It will print false, false.

Check Answer

82. QID - 2.1014

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     Object obj1 = new Object();
     Object obj2 = obj1;
     if( obj1.equals(obj2) ) System.out.println("true");
     else  System.out.println("false");
  }
}
```

Select 1 option

A. true

B. false

C. It will not compile.

D. It will compile but throw an exception at run time.

E. None of the above.

Check Answer

83. QID - 2.877

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){        this.side = length;    }

    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

    double getArea() {   return area;   }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Select 4 options

A. Add a calculateArea method:

```
private void calculateArea(){
  this.area = this.side*this.side;
}
```

B. Make `side` and `area` fields private.

C. Change setSide method to:

```
public void setSide(double d){
this.side = d;
calculateArea();
}
```

D. Make the `getArea` method public.

E. Add a `setArea() method:`

```
public void setArea(double d){ area = d; }
```

Check Answer

84. QID - 2.1251

What will be the result of attempting to compile and run the following code?

```
public class InitClass{
   public static void main(String args[ ] ){
      InitClass obj = new InitClass(5);
   }
   int m;
   static int i1 = 5;
   static int i2 ;
   int  j = 100;
   int x;
   public InitClass(int m){
      System.out.println(i1 + "  " + i2 + "   " + x + "  " + j + "  '
   }
  { j = 30; i2 = 40; }  // Instance Initializer
   static { i1++; }      // Static Initializer
}
```

Select 1 option

A. The code will fail to compile, since the instance initializer tries to assign a value to a static member.

B. The code will fail to compile, since the member variable x will be uninitialized when it is used.

C. The code will compile without error and will print 6, 40, 0, 30, 5 when run.

D. The code will compile without error and will print 5, 0, 0, 100, 5 when run.

E. The code will compile without error and will print 5, 40, 0, 30, 0 when run.

Check Answer

85. QID - 2.1068

Which of the following statements are valid ?

Select 2 options

A. `String[ ] sa = new String[3]{ "a", "b", "c"};`

B. `String sa[ ] = { "a ", " b", "c"};`

C. `String sa = new String[ ]{"a", "b", "c"};`

D. `String sa[ ] = new String[ ]{"a", "b", "c"};`

E. `String sa[ ] = new String[ ] {"a" "b" "c"};`

Check Answer

86. QID - 2.1087

Which of the given lines can be inserted at //1 of the following program ?

```
public class TestClass{
   public static void main(String[] args){
     short s = 9;
     //1
   }
}
```

Select 2 options

A. `Short k = new Short(9); System.out.println(k instanceof Short);`

B. `System.out.println(s instanceof Short);`

C. `Short k = 9; System.out.println( k instanceof s);`

D. `int i = 9; System.out.println(s == i);`

E. `Boolean b = s instanceof Number;`

F. `Short k = 9; Integer i = 9; System.out.println(k == i);`

G. `Integer i = 9; System.out.println( s == i );`

Check Answer

87. QID - 2.1332

What happens when you try to compile and run the following program?

```
public class CastTest{
   public static void main(String args[ ] ){
      byte b = -128 ;
      int i = b ;
      b = (byte) i;
      System.out.println(i+" "+b);
   }
}
```

Select 1 option

A. The compiler will refuse to compile it because i and b are of different types cannot be assigned to each other.

B. The program will compile and will print -128 and -128 when run .

C. The compiler will refuse to compile it because -128 is outside the legal range of values for a byte.

D. The program will compile and will print 128 and -128 when run .

E. The program will compile and will print 255 and -128 when run .

Check Answer

88. QID - 2.1191

Which of the following code fragments will successfully initialize a two-dimensional array of chars named cA with a size such that cA[2][3] refers to a valid element?

```
1.
  char[][] cA = {  { 'a', 'b', 'c' },  { 'a', 'b', 'c' }   };
2.
  char cA[][] = new char[3][];
  for (int i=0; i<cA.length; i++) cA[i] = new char[4];
3.
  char cA[][] = { new char[ ]{ 'a', 'b', 'c' }  ,   new char[ ]{
'a', 'b', 'c' }  };
4
  char cA[3][2] = new char[][] {  { 'a', 'b', 'c' },   { 'a', 'b',
'c' }   };
5.
  char[][] cA = { "1234", "1234",  "1234"  };
```

Select 1 option

A. 1, 3

B. 4, 5

C. 2, 3

D. 1, 2, 3

E. 2

Check Answer

89. QID - 2.952

Which of the following lines of code that, when inserted at line 1, will make the overriding method in SubClass invoke the overridden method in BaseClass on the current object with the same parameter.

```
class BaseClass{
   public void print(String s) {  System.out.println("BaseClass :"+s)
}
class SubClass extends BaseClass{
   public void print(String s){
      System.out.println("SubClass :"+s);
      // Line 1
   }
   public static void main(String args[]){
      SubClass sc = new SubClass();
      sc.print("location");
   }
}
```

Select 1 option

A. `this.print(s);`

B. `super.print(s);`

C. `print(s);`

D. `BaseClass.print(s);`

Check Answer

90. QID - 2.1282

What, if anything, is wrong with the following code?

```
void test(int x){
   switch(x){
      case 1:
      case 2:
      case 0:
      default :
      case 4:
   }
}
```

Select 1 option

A. Data Type of 'x' is not valid to be used as an expression for the switch clause.

B. The case label 0 must precede case label 1.

C. Each case section must end with a break keyword.

D. The default label must be the last label in the switch statement.

E. There is nothing wrong with the code.

Check Answer

Test 2 (Answered)

01. QID - 2.1045 : Using Operators and Decision Constructs

Which line, if any, will give a compile time error ?

```
void test(byte x){
   switch(x){
      case 'a':    // 1
      case 256:    // 2
      case 0:      // 3
      default :    // 4
      case 80:     // 5
   }
}
```

Correct Option is : B

~~**A.**~~ Line 1 as 'a' is not compatible with byte.

> int value of 'a' can easily fit into a byte.

B. Line 2 as 256 cannot fit into a byte.

~~**C.**~~ No compile time error but a run time error at line 2.

~~**D.**~~ Line 4 as the default label must be the last label in the switch statement.

> Any order of case statements is valid.

~~**E.**~~ There is nothing wrong with the code.

Explanation:

Every case constant expression in a switch block must be assignable to the type of switch expression. Meaning :

```
byte by = 10;
switch(by){
     300 :  //some code;
     56 :   //some code;
}
```

This will not compile as 300 is not assignable to 'by ' which can only hold values from -128 to 127. This gives compile time error as the compiler detects it while compiling. The use of break keyword is not mandatory, and without it the control will simply fall through the labels of the switch statement.

Back to Question without Answer

02. QID - 2.1276 : Handling Exceptions

What is wrong with the following code?

```
class MyException extends Exception {}
public class TestClass{
   public static void main(String[] args){
      TestClass tc = new TestClass();
      try{
         tc.m1();
      }
      catch (MyException e){
         tc.m1();
      }
      finally{
         tc.m2();
      }
   }
   public void m1() throws MyException{
      throw new MyException();
   }
   public void m2() throws RuntimeException{
      throw new NullPointerException();
   }
}
```

Correct Option is : D

~~**A.**~~ It will not compile because you cannot throw an exception in finally block.

> You can, but then you have to declare it in the method's throws clause.

~~**B.**~~ It will not compile because you cannot throw an exception in catch block.

> You can, but then you have to declare it in the method's throws clause.

~~**C.**~~ It will not compile because NullPointerException cannot be created this way.

> It does have a no args constructor.

D. It will not compile because of unhandled exception.

~~**E.**~~ It will compile but will throw an exception when run.

Explanation:

The catch block is throwing a checked exception (i.e. non-RuntimeException) which must be handled by either a try catch block or declared in the throws clause of the enclosing method.
Note that finally is also throwing an exception here, but it is a RuntimeException so there is no need to handle it or declare it in the throws clause.

Back to Question without Answer

03. QID - 2.1274 : Working with Java Data Types - String, StringBuilder

What will be the result of attempting to compile and run the following code?

```
class TestClass{
   public static void main(String args[] ){
      String str1 = "str1";
      String str2 = "str2";
      System.out.println( str1.concat(str2) );
      System.out.println(str1);
   }
}
```

Correct Option is : D

~~**A.**~~ The code will fail to compile.

~~**B.**~~ The program will print `str1` and `str1`.

~~**C.**~~ The program will print `str1` and `str1str2`

D. The program will print `str1str2` and `str1`

> `str1.concat(str2)` actually creates a new object that contains `"str1str2"`. So it does not affect the object referenced by `str1`.

~~**E.**~~ The program will print `str1str2` and `str1str2`.

Explanation:

Note that String objects are immutable. No matter what operation you do, the original object will remain the same and a new object will be returned. Here, the statement

str1.concat(str2) creates a new String object which is printed but its reference is lost after the printing.

Back to Question without Answer

04. QID - 2.949 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2 = false;
if (b2 != b1 = !b2){
   System.out.println("true");
}
else{
   System.out.println("false");
}
```

Correct Option is : A

A. Compile time error.

~~**B.**~~ It will print `true`.

~~**C.**~~ It will print `false`.

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

Note that boolean operators have more precedence than =. (In fact, = has least precedence of all operators.)
so, in (b2 != b1 = !b2) first b2 != b1 is evaluated which returns a value 'false'. So the expression becomes false = !b2. And this is illegal because false is a value and not a

variable!

Had it been something like (b2 = b1 != b2) then it is valid because it will boil down to : b2 = false.
Because all an if() needs is a boolean, now b1 != b2 returns false which is a boolean and as b2 = false is an expression and every expression has a return value (which is actually the Left Hand Side of the expression). Here, it returns false, which is again a boolean.

Note that return value of expression : i = 10 , where i is an int, is 10 (an int).

Back to Question without Answer

05. QID - 2.873 : Creating and Using Arrays

Which of the following are benefits of ArrayList over an array?

Correct Option is : A

A. You do not have to worry about the size of the ArrayList while inserting elements.

> An ArrayList resized dynamically at run time as per the situation. An array cannot be resized once created. This reduces the amount of boiler plate code that is required to do the same task using an array.

~~**B.**~~ It consumes less memory space.

> Because of additional internal data structure and pointers, it actually consumes a little more memory than an array.

~~**C.**~~ You do not have to worry about thread safety.

> An ArrayList, just like an array is not thread safe. If you have multiple threads trying to add and remove elements from an ArrayList, you have to write additional code to ensure thread safety.

~~**D.**~~ It allows you to write type safe code.

> Since ArrayList is a generics enabled class, it helps you write type safe code. For example, if you have:
> `ArrayList<String> al = new ArrayList<>();`
> `al.add(new Integer(10));` will not compile because the compiler knows that al can only contain Strings.
>
> However, this is not an advantage over an array because arrays are also type

safe. For example, if you have:

```
String[] sa = new String[10];
```

you cannot do `sa[0] = new Integer(10);` either.

But you can do `Object[] oa = sa;` and `oa[0]  = new Integer(10);` This will compile fine but will fail at runtime. This is a hole in the type safety provided by arrays.

Back to Question without Answer

06. QID - 2.923 : Java Basics - Garbage Collection

Given the following code:

```
class M { }
class N{
   private M m = new M();
   public void makeItNull(M pM){
      pM = null;
   }
   public void makeThisNull(){
      makeItNull(m);
   }
   public static void main(String[] args){
      N n = new N();
      n.makeThisNull();
   }
}
```

Which of the following statements are correct?

Correct Option is : A

A. There are no instances of M to be garbage collected till the program ends.

~~**B.**~~ A call to `n.makeThisNull()` marks the private instance of M for garbage collection.

~~**C.**~~ Setting `pM = null;` in `makeItNull()`, marks the private instance of M for garbage collection.

> `pM` is just a method parameter (a copy of the original reference) that is passed to `makeItNull()`. So setting it to null will not affect the original variable.

~~**D.**~~ `private` members of a class become eligible for garbage collection only when the instance of the class itself becomes eligible for garbage collection.

This is not true. Any instance can be made eligible by setting all its references to `null`. For example, in the following code, the Object instance referred to by 'o', can be made eligible for garbage collection by calling `setNull()`, even if the instance of X itself is not eligible for garbage collection.

```
class X{
  Object o = new Object();
  public void setNull(){ o = null; }
}
```

On the other hand, if the container object becomes eligible for GC and if there are no references to the contained objects outside of the container, the contained objects also become eligible for GC. For example, in the following code, both - the instance of X and the object instance contained inside X, will become eligible for garbage collection:

```
...
X x = new X();
x = null;
...
```

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

Back to Question without Answer

07. QID - 2.1263 : Working with Methods - Access Modifiers

Consider the following code in TestClass.java file:

```
package p;
private class TC extends java.util.HashMap{
   public TC(){
      super(100);
      System.out.println("TC created");
   }
}
public class TestClass extends TC{
   public TestClass(){
      System.out.println("TestClass created");
   }
   public static void main(String[] args){ new TestClass(); }
}
```

What will be the output when TestClass is run?

Correct Option is : E

~~**A.**~~ `"TestClass created"` as well as `"TC created"`.

~~**B.**~~ It will not compile because `HashMap` is a final class.

> `HashMap` is not a final class.

~~**C.**~~ Only `"TestClass created"` will be printed.

~~**D.**~~ Only `"TC created"` will be printed.

E. None of the above are correct.

Explanation:

The file will not compile because `TC` is a top level class and `private` is not a valid access modifier for a top level class. `private` can be applied to an inner class.

Back to Question without Answer

08. QID - 2.1227 : Using Loop Constructs

Given the following code, which of these statements are true?

```
class TestClass{
   public static void main(String args[]){
      int k = 0;
      int m = 0;
      for ( int i = 0; i <= 3; i++){
         k++;
         if ( i == 2){
            // line 1
         }
         m++;
      }
      System.out.println( k + ", " + m );
   }
}
```

Correct Options are : A C E

A. It will print 3, 2 when line 1 is replaced by break;

B. It will print 3, 2 when line 1 is replaced by continue.

C. It will print 4, 3 when line 1 is replaced by continue.

D. It will print 4, 4 when line 1 is replaced by i = m++;

It will print 4, 5

E. It will print 3, 3 when line 1 is replaced by i = 4;

Explanation:

This is a simple loop. All you need to do is execute each statement in your head. For example, if line 1 is replaced by break:

```
1. k=0, m=0
2. iteration 1: i=0
    2.1 k = 1
    2.2 i == 2 is false
    2.3 m = 1
3. iteration 2: i = 1
    3.1 k=2
    3.2 i==2 is false
    3.3 m = 2
4. iteration 3: i = 2
    4.1 k=3
    4.2 i==2 is true
    4.3 break
5. print 3, 2
```

Back to Question without Answer

09. QID - 2.866 : Handling Exceptions

What can be the type of a `catch` argument ?

Correct Option is : C

~~**A.**~~ Any class that extends `java.lang.Exception`

~~**B.**~~ Any class that extends `java.lang.Exception` except any class that extends `java.lang.RuntimeException`

C. Any class that is-a `Throwable`.

> The catch argument type declares the type of exception that the handler can handle and must be the name of a class that extends `Throwable` or `Throwable` itself.

~~**D.**~~ Any Object

~~**E.**~~ Any class that extends `Error`

Explanation:

You must remember the hierarchy of exception classes:

The base class of all exceptions is `java.lang.Throwable`. `java.lang.Error` and `java.lang.Exception` are the only two subclasses of `Throwable`.

Error is used by the JVM to throw exception that have nothing to do with the program code as such but occur because of environment. For example, `OutOfMemoryError`. It

indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions. Error and its subclasses are regarded as unchecked exceptions for the purposes of compile-time checking of exceptions.

Exception is used by the programmer as well as the JVM when it encounters exceptional situation in the program. Exception and its subclasses (except RuntimeException) are called Checked Exceptions. Checked exceptions need to be declared in a method or constructor's throws clause if they can be thrown by the execution of the method or constructor and propagate outside the method or constructor boundary. For example, `java.io.IOException`.

`RuntimeException` extends `Exception`, which is used to report exceptional situations that cannot be predetermined at compile time. For example, `IndexOutOfBoundsException` or `NullPointerException`. `RuntimeException` and its subclasses are unchecked exceptions. Unchecked exceptions do not need to be declared in a method or constructor's throws clause.

Back to Question without Answer

10. QID - 2.1052 : Overloading methods

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10);        //1
        test1(10, 20); //2
   }

   public static void main(String[] args){
     new Varargs().test();
   }

   //insert method here.
}
```

Which of the following lines can be added independently to the above class so that it will run without any errors or exceptions?

Correct Options are : B C

A. ~~`public void test1(int i, int j){}`~~

This will work only for //2.

B. `public void test1(int i, int... j){}`

The last parameter is a varargs of type int, which means, it can take any number of integers. Thus, it satisfies both //1 and //2.

C. `public void test1(int... i){}`

Since the only parameter is a varargs of type int, it can take any number of

integers. Thus, it satisfies both //1 and //2.

D. `public void test1(int i...){}`

This is not a correct syntax for a var-arg parameter.

E. `public void test1(int[] i){}`

Even though a var-arg parameter of type int is very similar to int[], they are not interchangeable. Therefore, int[] cannot be substituted for int... and it will not satisfy either of //1 or //2.

Explanation:

An interesting observation is that if you do javap on the following class, you will see the same signature for method m1 and m2. This shows that a var-arg parameter of type T is same as an array of type T.

```
class TestClass {
    String m1(int[] i) { return ""+i.length; }
    String m2(int... i) { return ""+i.length; }
}


javap TestClass produces this:

    java.lang.String m1(int[]);
    java.lang.String m2(int[]);


Conversely, the following code will not compile:

class TestClass {
    String m1(int[] i) { return ""+i.length; }
    String m1(int... i) { return ""+i.length; }  // Compiler determi
}
```

Back to Question without Answer

11. QID - 2.1325 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following class?

```
public class IfTest{
   public static void main(String args[]){
      if (true)
      if (false)
      System.out.println("True False");
      else
      System.out.println("True True");
   }
}
```

Correct Option is : D

~~**A.**~~ The code will fail to compile because the syntax of the `if` statement is not correct.

> It is perfectly valid.

~~**B.**~~ The code will fail to compile because the values in the condition bracket are invalid.

> Any expression that returns a boolean is valid. false and true are valid expressions that return boolean.

~~**C.**~~ The code will compile correctly and will not display anything.

D. The code will compile correctly and will display `True True`.

~~**E.**~~ The code will compile correctly but will display `True False`

Explanation:

This code can be rewritten as follows:

```
public class IfTest{
    public static void main(String args[]) {
        if (true) {
            if (false) {
                System.out.println("True False");
            } else {
                System.out.println("True True");
            }
        }
    }
}
```

Notice how the last "else" is associated with the last "if" and not the first "if". Now, the first if condition returns true so the next 'if' will be executed. In the second 'if' the condition returns false so the else part will be evaluated which prints 'True True'.

Back to Question without Answer

12. QID - 2.958 : Using Operators and Decision Constructs

Consider the following code:

```
public class Conversion{
   public static void main(String[] args){
     int i = 1234567890;
     float f = i;
     System.out.println(i - (int)f);
   }
}
```

What will it print when run?

Correct Option is : B

~~**A.**~~ It will print 0.

B. It will not print 0.

~~**C.**~~ It will not compile.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

Actually it prints `-46`. This is because the information was lost during the conversion from type int to type float as values of type float are not precise to nine significant digits.

Note: You are not required to know the number of significant digits that can be stored by a float for the exam. However, it is good to know about loss of precision while using float and double.

Back to Question without Answer

13. QID - 2.1361 : Using Operators and Decision Constructs

The following program will print

```
java.lang.ArithmeticException: / by zero
```

```
class Test{
   public static void main(String[] args){
      int d = 0;
      try{
         int i = 1 / (d* doIt());
      } catch (Exception e){
         System.out.println(e);
      }
   }
   public static int doIt() throws Exception{
      throw new Exception("Forget It");
   }
}
```

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

It will print `Forget It` because before the division can take place `doIt()` will throw an exception.

Back to Question without Answer

14. QID - 2.1029 : Creating and Using Arrays

The following class will print 'index = 2' when compiled and run.

```
class Test{
   public static int[ ] getArray() {  return null;  }
   public static void main(String[] args){
      int index = 1;
      try{
         getArray()[index=2]++;
      }
      catch (Exception e){  }  //empty catch
      System.out.println("index = " + index);
   }
}
```

Correct Option is : A

A. True

B. False

Explanation:

If the array reference expression produces null instead of a reference to an array, then a NullPointerException is thrown at runtime, but only after all parts of the array reference expression have been evaluated and only if these evaluations completed normally.

This means, first index = 2 will be executed, which assigns 2 to index. After that null[2] is executed, which throws a NullPointerException. But this exception is caught by the catch block, which prints nothing. So it seems like NullPointerException is not thrown but it actually is.

In other words, the embedded assignment of 2 to index occurs before the check for array reference produced by getArray().

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated. Note that if evaluation of the expression to the left of the brackets completes abruptly, no part of the expression within the brackets will appear to have been evaluated.

Back to Question without Answer

15. QID - 2.1300 : Working with Inheritance

An abstract method cannot be overridden.

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

Abstract methods are meant to be overridden in the subclass. Abstract methods describe a behavior but do not implement it. So the subclasses have to override it to actually implement the behavior. A subclass may chose not to override it, in which case, the subclass will have to be abstract too.

Back to Question without Answer

16. QID - 2.1071 : Using Operators and Decision Constructs

What will be the output of the following code snippet?

```
int a = 1;
int[] ia = new int[10];
int b = ia[a];
int c = b + a;
System.out.println(b = c);
```

Correct Option is : B

~~**A.** 0~~

B. 1

~~**C.** 2~~

~~**D.** true~~

~~**E.** false~~

Explanation:

1. All the elements of an array of primitives are automatically initialized by default values, which is 0 for numeric types and false for boolean.
Therefore, ia[1] is 0.
2. = is not same as ==. The statement b = c assigns c (whose value is 1) to b. which is then printed.

Back to Question without Answer

17. QID - 2.1221 : Working with Java Data Types - Variables and Objects

Which of the following is illegal ?

Correct Option is : D

~~**A.**~~ char c = 320;

> This is valid because 320 is below the maximum value that a char can take, which is 2^16 -1. Remember that char can take only positive values.

~~**B.**~~ float f = 320;

> 320 is an int and so can be assigned to an int variable. f = 320.0 is not valid as 320.0 would be a double.

~~**C.**~~ double d = 320;

> This is valid because an int can be assigned to a double without any cast.

D. byte b = 320;

> 320 cannot fit into a byte so you must cast it.: byte b = (byte) 320;

~~**E.**~~ None of the above is illegal.

Back to Question without Answer

18. QID - 2.982 : Working with Java Data Types - Variables and Objects

Given the following class, which of the given blocks can be inserted at line 1 without errors?

```
public class InitClass{
       private static int loop = 15 ;
       static final int INTERVAL = 10 ;
       boolean flag ;
       //line 1
}
```

Correct Options are : A B C E

A. `static {System.out.println("Static"); }`

B. `static { loop = 1; }`

C. `static { loop += INTERVAL; }`

D. `static { INTERVAL = 10; }`

> `INTERVAL` is final and so it can never be changed after it is given a value.

E. `{ flag = true; loop = 0; }`

> `flag` is not static and so it can be accessed only from a non-static block. `loop` is static so can be accessed from any block.

Back to Question without Answer

19. QID - 2.836 : Working with Inheritance

Which of the following statements are correct?

Correct Options are : A B C

A. An abstract class can be extended by an abstract or a concrete class.

B. A concrete class can be extended by an abstract or a concrete class.

C. An interface can be extended by another interface.

~~**D.**~~ An interface can be extended by an abstract class.

A class "implements" an interface. It does not "extend" an interface.

~~**E.**~~ An interface can be extended by a concrete class.

~~**F.**~~ An abstract class cannot implement an interface.

Any class, whether abstract or concrete, can implement any interface.

Back to Question without Answer

20. QID - 2.1130 : Working with Inheritance

You want to invoke the overridden method (the method in the base class) from the overriding method (the method in the derived class) named `m()`.
Which of the following constructs which will let you do that?

Correct Option is : A

A. `super.m();`

~~**B.**~~ `super.this();`

~~**C.**~~ `base.m();`

~~**D.**~~ `parent.m();`

~~**E.**~~ `super();`

Explanation:

Note that calling `super();` means you are trying to call the super class's constructor. But you can't call the super class's constructor (or its own constructor) from a method (because by the time a method gets to run, the object has already been constructed), therefore calling `super()` from a method is not valid.
`super()` can only be the first statement of a constructor.

Back to Question without Answer

21. QID - 2.1209 : Working with Inheritance

Which statements concerning the following code are true?

```
class A{
  public A() {} // A1
  public A(String s) {  this();  System.out.println("A :"+s);  }  //
}

class B extends A{
  public int B(String s) {  System.out.println("B :"+s);  return 0;
}
class C extends B{
    private C(){ super(); } // C1
    public C(String s){  this();  System.out.println("C :"+s);  } //
    public C(int i){} // C3
}
```

Correct Options are : A B C D

A. At least one of the constructors of each class is called as a result of constructing an object of class C.

> To create any object one and only one constructor of that class and each of the super classes is called. (A constructor may delegate the construction to another constructor of the same class by calling this(...) as the first statement.)

B. Constructor at //A2 will never be called in creation of an object of class C

> Because B has no defined constructor and so a default no-argument constructor will be called, which will call the no-argument constructor of A

C. Class C can be instantiated only in two ways by users of this class.

As one constr. is private, users of this class (i.e. classes other than class C) can use only other 2 public constr.

D. //B1 will never be called in creation of objects if class A, B, or C

Because //B1 is not a constructor. Note that it is returning an int. A constructor does not have any return type, not even void.

~~**E.**~~ The code will not compile.

Back to Question without Answer

22. QID - 2.1127 : Working with Inheritance

Consider the following class and interface definitions (in separate files):

```
public class Sample implements IInt{
 public static void main(String[] args){
  Sample s = new Sample();  //1
  int j = s.thevalue;       //2
  int k = IInt.thevalue;    //3
  int l = thevalue;         //4
 }
}
public interface IInt{
 int thevalue = 0;
}
```

What will happen when the above code is compiled and run?

Correct Option is : E

~~**A.**~~ It will give an error at compile time at line //1.

~~**B.**~~ It will give an error at compile time at line //2.

~~**C.**~~ It will give an error at compile time at line //3

~~**D.**~~ It will give an error at compile time at line //4.

E. It will compile and run without any problem.

Explanation:

As a rule, fields defined in an interface are public, static, and final. (The methods are public and abstract.)
Here, the interface IInt defines 'thevalue' and thus any class that implements this interface inherits this field. Therefore, it can be accessed using s.thevalue or just 'thevalue' inside the class. Also, since it is static, it can also be accessed using IInt.thevalue or Sample.thevalue.

Back to Question without Answer

23. QID - 2.1104 : Using Operators and Decision Constructs

Consider the following lines of code:

```
Integer i = new Integer(42);
Long ln = new Long(42);
Double d = new Double(42.0);
```

Which of the following options are valid?

Correct Options are : C D E

~~**A.**~~ `i == ln;`

> This will fail at compile time

~~**B.**~~ `ln == d;`

> This will fail at compile time

C. `i.equals(d);`

D. `d.equals(ln);`

E. `ln.equals(42);`

> Due to auto-boxing int 42 is converted into an Integer object containing 42. So this is valid. It will return false though because ln is a Long and 42 is boxed into an Integer.

Explanation:

The concept to understand here is as follows -
If the compiler can figure out that something can NEVER happen, then it flags an error. In this question, the compiler knows that ln, i or d can never point to the same object in any case because they are references to different classes of objects that have no relation (superclass/subclass) between themselves.

Back to Question without Answer

24. QID - 2.1141 : Java Basics

Which line(s) in the following code will cause a compilation error?

```
static import java.lang.System.*; //1
class $$ //2
{
    static public void main(String... _$_) //3
   {
       String _ = ""; //4
       for(int $=0; ++$ < _$_.length; )  //5
           _ += _$_[$];  //6
       out.println(_);  //7
    }
}
```

Correct Option is : A

A. 1

> Read the question carefully. The order of static and import is invalid. It should have been "import static". Everything else is legal in the code!

~~B.~~ 2

~~C.~~ 3

~~D.~~ 4

~~E.~~ 5

~~F.~~ 6

~~G.~~ 7

~~H.~~ None of the lines is invalid.

Back to Question without Answer

25. QID - 2.1041 : Working with Java Data Types - Variables and Objects

Which of the following is not a primitive data value in Java?

Correct Options are : A D

A. `"x"`

> This is a string containing x. String is not a primitive data type.

~~B.~~ `'x'`

> This is a char.

~~C.~~ `10.2F`

D. `Object`

~~E.~~ `false`

Explanation:

Java has only the following primitive data types:
boolean, byte, short, char, int, long, float and double.

Back to Question without Answer

26. QID - 2.1032 : Using Loop Constructs

What will the following code print?

```
public class BreakTest{
  public static void main(String[] args){
    int i = 0, j = 5;
    lab1 : for( ; ; i++){
      for( ; ; --j)  if( i >j ) break lab1;
    }
    System.out.println(" i = "+i+", j = "+j);
  }
}
```

Correct Option is : D

~~**A.**~~ `i = 1, j = -1`

~~**B.**~~ `i = 1, j = 4`

~~**C.**~~ `i = 0, j = 4`

D. `i = 0, j = -1`

~~**E.**~~ It will not compile.

Explanation:

The values of i and j in the inner most for loop change as follows:

```
i = 0 j = 5
i = 0 j = 4
```

```
i = 0 j = 3
i = 0 j = 2
i = 0 j = 1
i = 0 j = 0
i = 0 j = -1
```

Therefore, the final println prints `i = 0, j = -1`

Back to Question without Answer

27. QID - 2.1049 : Working with Inheritance

Consider that you are writing a set of classes related to a new Data Transmission Protocol and have created your own exception hierarchy derived from java.lang.Exception as follows:

```
enthu.trans.ChannelException
              +-- enthu.trans.DataFloodingException,
                    enthu.trans.FrameCollisionException
```

You have a TransSocket class that has the following method:

```
long connect(String ipAddr) throws ChannelException
```

Now, you also want to write another "AdvancedTransSocket" class, derived from "TransSocket" which overrides the above mentioned method. Which of the following are valid declaration of the overriding method?

Correct Options are : C E

~~**A.**~~ `int connect(String ipAddr) throws DataFloodingException`

> The return type must match. Otherwise the method is OK.

~~**B.**~~ `int connect(String ipAddr) throws ChannelException`

> The return type must match. Otherwise the method is OK.

C. `long connect(String ipAddr) throws FrameCollisionException`

~~**D.**~~ `long connect(String ipAddr) throws Exception`

This option is invalid because `Exception` is a super class of `ChannelException` so it cannot be thrown by the overriding method.

E. `long connect(String str)`

Explanation:

There are 2 important concepts involved here:
1. The overriding method must have same return type in case of primitives (a subclass is allowed in case of classes) (Therefore, the choices returning int are not valid.) and the parameter list must be the same (The name of the parameter does not matter, just the Type is important).

2. The overriding method can throw a subset of the exception or subclass of the exceptions thrown by the overridden class. Having no throws clause is also valid since an empty set is a valid subset.

Back to Question without Answer

28. QID - 2.1053 : Working with Methods - Access Modifiers

Compared to public, protected and private accessibility, default accessibility is....

Correct Option is : C

~~**A.**~~ Less restrictive than public

> public is least restrictive.

~~**B.**~~ More restrictive than public, but less restrictive than protected.

C. More restrictive than protected, but less restrictive than private.

> The default accessibility is more restrictive than `protected`, but less restrictive than `private`. Members with default accessibility are only accessible within the class itself and from other classes in the same package. `protected` members are in addition accessible from subclasses in any other package as well. Members with `private` accessibility are only accessible within the class itself.

~~**D.**~~ More restrictive than private.

> private is most restrictive.

~~**E.**~~ Less restrictive than protected from within a package, and more restrictive than protected from outside a package.

Explanation:

The correct order :
public < protected < package (or default) < private

(here, public is least restrictive and private is most restrictive.)

Back to Question without Answer

29. QID - 2.1118 : Java Basics

Which of the changes given in options can be done (independent of each other) to let the following code compile and run without errors?

```
class SomeClass{
   String s1 = "green mile";   // 0
   public void generateReport( int n ){
      String local;   // 1
      if( n > 0 ) local = "good";   //2
      System.out.println( s1+" = " + local );   //3
   }
}
```

Correct Options are : A C

A. Insert after line 2 : `else local = "bad";`

~~**B.**~~ Insert after line 2 : `if(n <= 0) local = "bad";`

C. Move line 1 and place it after line 0.

~~**D.**~~ change line 1 to : `final String local = "rocky";`

> Making it final will not let //2 compile as it would then try to modify a final variable.

~~**E.**~~ The program already is without any errors.

Explanation:

The problem is that `local` is declared inside a method is therefore local to that

method. It is called a local variable (also known as automatic variable) and it cannot be used before initialized. Further, it will not be initialized unless you initialize it explicitly because local variables are not initialized by the JVM on its own. The compiler spots the usage of such uninitialized variables and ends with an error message.

1. Making it a member variable (choice "Move line 1 and place it after line 0.") will initialize it to `null`.
2. Putting an else part (choice "Insert after line 2 : `else local = "bad";`") will ensure that it is initialized to either '`good`' or '`bad`'. So this also works.
Choice "Insert after line 2 : `if(n <= 0) local = "bad";`" doesn't work because the second `'if'` will actually be another statement and is not considered as a part of first `'if'`. So, compiler doesn't realize that '`local`' will be initialized even though it does get initialized.

Back to Question without Answer

30. QID - 2.1293 : Working with Methods

Which of the following correctly defines a method named stringProcessor that can be called by other programmers as follows: `stringProcessor(str1)` or `stringProcessor(str1, str2)` or `stringProcessor(str1, str2, str3)`, where `str1`, `str2`, and `str3` are references to Strings.

Correct Option is : B

~~**A.**~~
```
public void stringProcessor(...String){
}
```

B.
```
public void stringProcessor(String... strs){
}
```

~~**C.**~~
```
public void stringProcessor(String[] strs){
}
```

~~**D.**~~
```
public void stringProcessor(String a, String b, String c){
}
```

~~**E.**~~ Three separate methods need to be written.

Explanation:

To allow a method to take variable arguments of a type, you must use the . . . syntax:
`methodName( <type>... variableName);`
Remember that there can be only one vararg argument in a method. Further, the vararg argument must be the last argument.
So this is invalid: `stringProcessor( String... variableName, int age);`

but this is valid: `stringProcessor(int age, String... variableName);`

Though not important for this exam, it is good to know that within the method, the vararg argument is treated like an array:

```
public void stringProcessor(String... names){
   for (String n : names) {
       System.out.println("Hello " + n);
   }
}
```

Back to Question without Answer

31. QID - 2.994 : Using Operators and Decision Constructs

Assume that a, b, and c refer to instances of primitive wrapper classes. Which of the following statements are correct?

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Correct Options are : A E

A. a.equals(a) will always return true.

~~**B.**~~ b.equals(c) may return false even if c.equals(b) returns true.

~~**C.**~~ a.equals(b) returns same as a == b.

> The wrapper classes's equals() method overrides Object's equals() method to compare the actual value instead of the reference.

~~**D.**~~ a.equals(b) throws an exception if they refer to instances of different classes.

> It returns false in such a case.

E. a.equals(b) returns false if they refer to instances of different classes.

Explanation:

Equals method of a primitive wrapper class (e.g. java.lang.Integer, Double, Float etc) are
1. symmetric => a.equals(b) returns same as b.equals(a)

2. transitive => if a.equals(b) and b.equals(c) return true, then a.equals(c) returns true.
3. reflexive => a.equals(a) return true.

For example, the following code for the equals method on Integer explains how it works:

```
public boolean equals(Object obj) {
   if (obj instanceof Integer) {
       return value == ((Integer)obj).intValue();
   }
   return false;
}
```

Back to Question without Answer

32. QID - 2.858 : Constructors

Which of the following are true about the "default" constructor?

Correct Option is : B

~~A.~~ It is provided by the compiler only if the class and any of its super classes does not define any constructor.

> It is provided by the compiler if the class does not define any constructor. It is immaterial if the super class provides a constructor or not.

B. It takes no arguments.

~~C.~~ A default constructor is used to return a default value.

> A constructor does not return any value at all. It is meant to initialize the state of an object.

~~D.~~ To define a default constructor, you must use the default keyword.

~~E.~~ It is always public.

> The access type of a default constructor is same as the access type of the class. Thus, if a class is public, the default constructor will be public.

Explanation:

The default constructor is provided by the compiler only when a class does not define ANY constructor explicitly. For example,

```
public class A{
```

```
  public A()  //This constructor is automatically inserted by the cor
  {
    super();  //Note that it calls the super class's default no-args
  }
}
public class A{
  //Compiler will not generate any constructor because the programme
  public A(int i){
     //do something
  }
}
```

Back to Question without Answer

33. QID - 2.972 : Working with Java Data Types - Variables and Objects

Consider the following program :

```
class Test{
  public static void main(String[] args){
    short s = 10;    // 1
    char c = s;      // 2
    s = c;           // 3
  }
}
```

Identify the correct statements.

Correct Options are : A B

A. Line 3 is not valid.

B. Line 2 is not valid.

~~**C.**~~ It will compile because both short and char can hold 10.

~~**D.**~~ None of the lines 1, 2 and 3 is valid.

Line 1 is valid because 10 is a constant and can fit into a short.

Explanation:

Not all short values are valid char values, and neither are all char values valid short values, therefore compiler complains for both the lines 2 and 3. They will require an explicit cast.

Back to Question without Answer

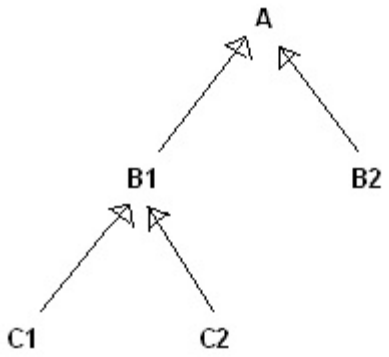
34. QID - 2.1287 : Working with Inheritance

Consider the following class hierarchy shown in the image. (B1 and B2 are subclasses of A and C1, C2 are subclasses of B1)

Assume that method `public void m1(){ ... }` is defined in all of these classes EXCEPT B1 and C1.

Assume that `"objectOfXX"` means a variable that points to an object of class XX. So, `objectOfC1` means a reference variable that is pointing to an object of class C1.

Which of the following statements are correct?



Correct Option is : C

~~**A.**~~ `objectOfC1.m1();` will cause a compilation error.

> C1 will inherit B1's `m1()` which in turn inherits `m1()` from A.

~~**B.**~~ `objectOfC2.m1();` will cause A's `m1()` to be called.

> C2 has `m1()`, so its `m1()` will override A's `m1()`.

C. `objectOfC1.m1();` will cause A's `m1()` to be called.

> C1 will inherit B1's `m1()` which in turn inherits `m1()` from A.

~~**D.**~~ `objectOfB1.m1();` will cause an exception at runtime.

> B1 will inherit `m1()` from A. So this is valid.

~~**E.**~~ `objectOfB2.m1();` will cause an exception at runtime.

> B2 overrides `m1()` of A. So there will be no exception.

Back to Question without Answer

35. QID - 2.999 : Working with Java Data Types - String, StringBuilder

Which of the following method calls can be applied to a String object?

Correct Options are : A B E

A. `equals(Object)`

B. `equalsIgnoreCase(String)`

~~C.~~ `prune()`

> There is no such method.

~~D.~~ `append()`

> This method is in StringBuffer and StringBuilder but not in String.

E. `intern()`

Explanation:

public String intern()
Returns a canonical representation for the string object.
A pool of strings, initially empty, is maintained privately by the class String.
When the intern method is invoked, if the pool already contains a string equal to this String object as determined by the equals(Object) method, then the string from the pool is returned. Otherwise, this String object is added to the pool and a reference to this String object is returned.
It follows that for any two strings s and t, s.intern() == t.intern() is true if and only if s.equals(t) is true.

All literal strings and string-valued constant expressions are interned. String literals are defined in 3.10.5 of the Java Language Specification

Returns:

a string that has the same contents as this string, but is guaranteed to be from a pool of unique strings.

Back to Question without Answer

36. QID - 2.1179 : Using Operators and Decision Constructs

The following code snippet will print 'true'.

```
short s = Short.MAX_VALUE;
char c = s;
System.out.println( c == Short.MAX_VALUE);
```

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

This will not compile because a short VARIABLE can NEVER be assigned to a char without explicit casting. A short CONSTANT can be assigned to a char only if the value fits into a char.

short s = 1; byte b = s; => this will also not compile because although value is small enough to be held by a byte but the Right Hand Side i.e. s is a variable and not a constant.
final short s = 1; byte b = s; => This is fine because s is a constant and the value fits into a byte.
final short s = 200; byte b = s; => This is invalid because although s is a constant but the value does not fit into a byte.

Implicit narrowing occurs only for byte, char, short, and int. Remember that it does not occur for long, float, or double. So, this will not compile: int i = 129L;

Back to Question without Answer

37. QID - 2.1239 : Using Operators and Decision Constructs

What letters will be printed by this program?

```
public class ForSwitch{
    public static void main(String args[]){
        char i;
        LOOP: for (i=0;i<5;i++){
            switch(i++){
                case '0': System.out.println("A");
                case 1: System.out.println("B"); break LOOP;
                case 2: System.out.println("C"); break;
                case 3: System.out.println("D"); break;
                case 4: System.out.println("E");
                case 'E' : System.out.println("F");
            }
        }
    }
}
```

Correct Options are : C E

~~**A.**~~ A

~~**B.**~~ B

C. C

~~**D.**~~ D

E. F

Explanation:

1. Defining i as char doesn't mean that it can only hold characters (a, b, c etc). It is an integral data type which can take any +ive integer value from 0 to 2^16 -1.
2. Integer 0 or 1, 2 etc. is not same as char '0', '1' or '2' etc.
so when i is equal to 0, nothing gets printed and i is incremented to 1 (due to i++ in the switch).
i is then incremented again by the for loop for next iteration. so i becomes 2.
when i = 2, "C" is printed and i is incremented to 3 (due to i++ in the switch) and then i is incremented to 4 by the for loop so i becomes 4.
when i = 4, "E" is printed and since there is no break, it falls through to case '5' and "F" is printed.
i is incremented to 5 and the for loop ends.

Back to Question without Answer

38. QID - 2.1358 : Handling Exceptions

What will the following program print when run using the command line: `java TestClass`

```
public class TestClass {

   public static void methodX() throws Exception {
      throw new AssertionError();
   }

   public static void main(String[] args) {
      try{
         methodX();
      }
      catch(Exception e) {
        System.out.println("EXCEPTION");
      }
   }
}
```

Correct Option is : A

A. It will throw `AssertionError` out of the main method.

> A subclass of `Error` cannot be caught using a catch block for `Exception` because `java.lang.Error` does not extend `java.lang.Exception`.

~~**B.**~~ It will print `EXCEPTION`.

> The catch block will not be able to catch the Error thrown by `methodX()`.

~~**C.**~~ It will not compile because of the throws clause in `methodX()`.

> The `throws` clause is valid even though unnecessary in this case.

~~**D.**~~ It will end without printing anything because assertions are disabled by default.

> It is true that assertions are disabled by default however, `methodX` is throwing an `AssertionError` explicitly like any other Throwable. Here, the assertion mechanism is not even used.

Back to Question without Answer

39. QID - 2.1351 : Working with Java Data Types - String, StringBuilder

Which of the following methods modify the object on which they are called?

Correct Option is : D

~~**A.**~~ `setValue(int)` of java.lang.Integer class.

> There is no such method in Integer class. Note that Integer, Float and other such wrapper objects are immutable.

~~**B.**~~ The `substring(int)` method of the String class

> String is an immutable object. calling substring(...) returns a new different String object. It cannot change the original object.

~~**C.**~~ The `replace()` method of the String class.

> String objects can never be modified once created.

D. The `reverse()` method of the StringBuffer class.

~~**E.**~~ None of these.

Back to Question without Answer

40. QID - 2.1196 : Using Operators and Decision Constructs

Consider the following code snippet:

```
XXXX m ;
  switch( m ){
     case 32  : System.out.println("32");   break;
     case 64  : System.out.println("64");   break;
     case 128 : System.out.println("128");  break;
  }
```

What type can 'm' be of so that the above code compiles and runs as expected ?

Correct Options are : A C E

A. `int m;`

> m can hold all the case values.

~~**B.**~~ `long m;`

> long, float, double, and boolean can never be used as a switch variable.

C. `char m;`

> m can hold all the case values.

~~**D.**~~ `byte m;`

> m will not be able to hold 128. a byte's range is -128 to 127.

E. `short m;`

m can hold all the case values.

Explanation:

Three rules must be remembered about switch statement:
1. Only byte, char, short, int, and enum values can be used as types of a switch variable. (Java 7 allows String as well.)
2. The switch variable must be big enough to hold all the case constants.
So, if switch variable is of type char, then none of the case constants can be greater than 65535 because char's range is from 0 to 65535.
3. All case labels should be COMPILE TIME CONSTANTS.

Back to Question without Answer

41. QID - 2.922 : Java Basics - Garbage Collection

In the following code, after which statement (earliest), the object originally held in s, may be garbage collected ?

```
1. public class TestClass{
2.   public static void main (String args[]){
3.      Student s = new Student("Vaishali", "930012");
4.      s.grade();
5.      System.out.println(s.getName());
6.      s = null;
7.      s = new Student("Vaishali", "930012");
8.      s.grade();
9.      System.out.println(s.getName());
10      s = null;
     }
   }

public class Student{
   private String name, rollNumber;

   public Student(String name, String rollNumber) {
      this.name = name;
      this.rollNumber = rollNumber;
   }

   //valid setter and getter for name and rollNumber follow

   public void grade() {
   }

}
```

Correct Option is : C

~~A.~~ It will not be Garbage Collected till the end of the program.

~~**B.** Line 5~~

C. Line 6

~~**D.** Line 7~~

~~**E.** Line 10~~

Explanation:

In this case, since there is only one reference to Student object, as soon as it is set to null, the object held by the reference is eligible for GC, here it is done at line 6.
Note that although an object is created at line 7 with same parameters, it is a different object and it will be eligible for GC after line 10.

Back to Question without Answer

42. QID - 2.988 : Working with Java Data Types - String, StringBuilder

What will the following code print?

```
public class Test{
    public static void stringTest(String s){
        s.replace('h', 's');
    }
    public static void stringBuilderTest(StringBuilder s){
        s.append("o");
    }
    public static void main(String[] args){
        String s = "hell";
        StringBuilder sb = new StringBuilder("well");
        stringTest(s);
        stringBuilderTest(sb);
        System.out.println(s + sb);
    }
}
```

Correct Option is : B

~~**A.**~~ sellwello

B. hellwello

~~**C.**~~ hellwell

~~**D.**~~ sellwell

~~**E.**~~ None of these.

Explanation:

A String is immutable while a StringBuilder is not. So in `stringTest()`, `"hell".replace('h', 's')` will produce a new String `"sell"` but will not affect the original String that was passed to the method.
However, the `append()` method of `StringBuilder` appends to the original String object. So, `"well"` becomes `"wello"`.

Back to Question without Answer

43. QID - 2.833 : Working with Methods

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showTwo(200);
        System.out.println(ct.myValue);
        ct.showOne(100);
        System.out.println(ct.myValue);
    }
}
```

Correct Option is : E

~~**A.**~~ 0 followed by 100.

~~**B.**~~ 100 followed by 100.

~~**C.**~~ 0 followed by 200.

~~**D.**~~ 100 followed by 200.

E. 200 followed by 200.

~~**F.**~~ 200 followed by 100

Explanation:

There are a couple of important concepts in this question:

1. Within an instance method, you can access the current object of the same class using 'this'. Therefore, when you access this.myValue, you are accessing the instance member myValue of the ChangeTest instance.

2. If you declare a local variable (or a method parameter) with the same name as the instance field name, the local variable "shadows" the member field. Ideally, you should be able to access the member field in the method directly by using the name of the member (in this example, myValue). However, because of shadowing, when you use myValue, it refers to the local variable instead of the instance field.

In showTwo method, there are two variables accessible with the same name myValue. One is the method parameter and another is the member field of ChangeTest object. Ideally, you should be able to access the member field in the method directly by using the name myValue but in this case, the method parameter shadows the member field because it has the same name. So by doing this.myValue, you are changing the instance variable myValue by assigning it the value contained in local variable myValue, which is 200. So in the next line when you print ct.myValue, it prints 200.

Now, in the showOne() method also, there are two variables accessible with the same name myValue. One is the method parameter and another is the member field of ChangeTest object. So when you use myValue, you are actually using the method parameter instead of the member field.

Therefore, when you do : myValue = myValue; you are actually assigning the value contained in method parameter myValue to itself. You are not changing the member field myValue. Hence, when you do System.out.println(ct.myValue); in the next line, it still prints 200.

Back to Question without Answer

44. QID - 2.1015 : Working with Inheritance

Which statement regarding the following code is correct?

```
class A{
   public int i = 10;
   private int j = 20;

}

class B extends A{
   private int i = 30; //1
   public int k = 40;

}

class C extends B{
}

public class TestClass{
   public static void main(String args[]){
      C c = new C();
      System.out.println(c.i); //2
      System.out.println(c.j); //3
      System.out.println(c.k);
   }
}
```

Correct Option is : B

~~**A.**~~ It will print `10` and `40` if //3 is commented.

B. It will print `40` if //2 and //3 are commented.

~~**C.**~~ It will not compile because of `//1`.

~~**D.**~~ It will compile if `//2` is commented.

~~**E.**~~ None of these.

Explanation:

You cannot access c.i because i is private in B. But you can access `( (A)c).i` because `i` is public in A. Remember that member variables are shadowed and not overridden. So, B's i shadows A's i and since B's i is private, you can't access A's i unless you cast the reference to A.
You cannot access `c.j` because `j` is private in A.

Back to Question without Answer

45. QID - 2.997 : Encapsulation

Consider the following class written by a novice programmer.

```
class Elliptical{
    public int radiusA, radiusB;
    public int sum = 100;

    public void setRadius(int r){
        if(r>99) throw new IllegalArgumentException();
        radiusA = r;
        radiusB = sum - radiusA;

    }
}
```

After some time, the requirements changed and the programmer now wants to make sure that radiusB is always (200 - radiusA) instead of (100 - radiusA) without breaking existing code that other people have written. Which of the following will accomplish his goal?

Correct Option is : E

~~**A.**~~ `Make sum = 200;`

~~**B.**~~ `Make sum = 200` and make it private.

~~**C.**~~ `Make sum = 200` and make all fields (radiusA, radiusB, and sum) private.

> This should have been done when the class was first written.

~~**D.**~~ Write another method `setRadius2(int r)` and `set radiusB` accordingly in this method.

E. His goal cannot be accomplished.

~~**F.**~~ This class will not compile.

> There is no problem with the code. Remember, IllegalArgumentException extends from RuntimeException and is a super class of NumberFormatException

Explanation:

setRadius method makes sure that radiusB is set to sum - radiusA. So changing sum to 200 should do it. However, note that radiusA, radiusB, and sum are public which means that any other class can access these fields directly without going through the setRadius method. So there is no way to make sure that the value of radiusB is correctly set at all times. If you make them private now, other classes that are accessing the fields directly will break.

The class should have been coded with proper encapsulation of the fields in the first place.

Back to Question without Answer

46. QID - 2.1324 : Using Operators and Decision Constructs

What will happen when the following program is compiled and run?

```
public class SM{
   public String checkIt(String s){
      if(s.length() == 0 || s == null){
         return "EMPTY";
      }
      else return "NOT EMPTY";
    }

    public static void main(String[] args){
      SM a = new SM();
      a.checkIt(null);
    }
}
```

Correct Option is : C

~~**A.**~~ It will print `EMPTY`.

~~**B.**~~ It will print `NOT EMPTY`.

C. It will throw `NullPointerException`.

> Because the first part of the expression `(s.length() == 0)` is trying to call a method on s, which is `null`. The check `s == null` should be done before calling a method on the reference.

~~**D.**~~ It will print `EMPTY` if || is replaced with |.

> In this case, replacing || with | will not make any difference because `s.length()`

will anyway be called before checking whether `s` is `null` or not. The right expression would be:

```
if( s == null || s.length() == 0) { ... }
```

In this case, || being a short circuit expression, `s.length() == 0` will not be called if `s == null` returns `true`. Hence, no `NullPointerExpression` will be thrown.

Back to Question without Answer

47. QID - 2.1009 : Working with Inheritance

Consider the following code:

```
class Super { static String ID = "QBANK"; }

class Sub extends Super{
   static { System.out.print("In Sub"); }
}
public class Test{
   public static void main(String[] args){
      System.out.println(Sub.ID);
   }
}
```

What will be the output when class Test is run?

Correct Option is : B

~~**A.**~~ It will print `In Sub` and `QBANK`.

B. It will print `QBANK`.

~~**C.**~~ Depends on the implementation of JVM.

~~**D.**~~ It will not even compile.

~~**E.**~~ None of the above.

Explanation:

A class or interface type T will be initialized at its first active use, which occurs if:

T is a class and a method actually declared in T (rather than inherited from a superclass) is invoked.

T is a class and a constructor for class T is invoked, or U is an array with element type T, and an array of type U is created.

A non-constant field declared in T (rather than inherited from a superclass or superinterface) is used or assigned. A constant field is one that is (explicitly or implicitly) both final and static, and that is initialized with the value of a compile-time constant expression . Java specifies that a reference to a constant field must be resolved at compile time to a copy of the compile-time constant value, so uses of such a field are never active uses.

All other uses of a type are passive

A reference to a field is an active use of only the class or interface that actually declares it, even though it might be referred to through the name of a subclass, a subinterface, or a class that implements an interface.

Back to Question without Answer

48. QID - 2.1042 : Using Loop Constructs

Consider the following method which is called with an argument of 7:

```
public void method1(int i){
   int j = (i*30 - 2)/100;

   POINT1 : for(;j<10; j++){
       boolean flag  = false;
       while(!flag){
 if(Math.random()>0.5) break POINT1;
       }
   }
  while(j>0){
     System.out.println(j--);
     if(j == 4) break POINT1;
   }
}
```

What will it print?
(Assume that Math.random() return a double between 0.0 and 1.0, not including 1.0)

Correct Option is : C

~~**A.**~~ It will print `1` and `2`

~~**B.**~~ It will print `1` to N where N is a random number.

C. It will not compile.

Remember that a labeled break or continue statement must always exist inside the loop where the label is declared. Here, `if(j == 4) break POINT1;` is a labelled break that is occurring in the second loop while the label POINT1 is

declared for the first loop.

D. It will throw an exception at runtime.

Back to Question without Answer

49. QID - 2.838 : Working with Inheritance

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Flyer) System.out.println("f is a Flyer");
        if(e instanceof Bird) System.out.println("e is a Bird");
        if(b instanceof Bird) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Correct Option is : A

A. It will not compile.

b points to an object of class Bat, which does not extend from Bird. Now, it is possible for b to point to an object of any subclass of Bat. However, it is not possible for that sub class to extend Bird (because a class can at most extend from only one class). Therefore, it is not possible for b to point to an object of a class that extends Bird. The compiler figures out this fact at compile time itself and so the code fails to compile.

~~**B.**~~ It will throw an exception when run.

~~**C.**~~ f is a Flyer
e is a Bird

> At run time, f points to an object of class Eagle. Now, Eagle extends Bird, which implements Flyer so f instanceof Flyer returns true.
>
> e points to an object of class Eagle. Eagle extends Bird. Therefore, e instanceof Bird will also return true.

~~**D.**~~ f is a Flyer

~~**E.**~~ e is a Bird

Explanation:

Note that there is no compilation issue with b instanceof Flyer because Flyer is an interface and it is possible for b to point to an object of a class that is a sub class of Bat and also implements Flyer. So the compiler doesn't complain. If you make Bat class as final, b instanceof Flyer will not compile because the compiler knows that it is not possible for b to point to an object of a class that implements Flyer.

Back to Question without Answer

50. QID - 2.1341 : Java Basics

What will be result of attempting to compile this class?

```
import java.util.*;
package test;
public class TestClass{
    public OtherClass oc = new OtherClass();
}
class OtherClass{
    int value;
}
```

Correct Option is : D

~~**A.**~~ The class will fail to compile, since the class OtherClass is used before it is defined.

~~**B.**~~ There is no problem with the code.

~~**C.**~~ The class will fail to compile, since the class OtherClass must be defined in a file called OtherClass.java

> This is not needed because OtherClass is not public. The class & file name must match only if the class is public.

D. The class will fail to compile .

> The package declaration can never occur after an import statement.

~~**E.**~~ None of the above.

Explanation:

The order is:
package statement.
import statements
class/ interface definitions.
Important point to note here is YOU MUST READ THE QUESTIONS VERY CAREFULLY.

Back to Question without Answer

51. QID - 2.1149 : Working with Inheritance

Given the following class definitions :

```
interface MyIface{};
class A {};
class B extends A implements MyIface{};
class C implements MyIface{};
```

and the following object instantiations:

```
A a = new A();
B b = new B();
C c = new C();
```

Which of the following assignments are legal at compile time?

Correct Option is : C

~~**A.**~~ `b = c;`

There is no relation between b and c.

~~**B.**~~ `c = b;`

There is no relation between b and c.

C. `MyIface i = c;`

Because C implements I.

~~**D.**~~ `c = (C) b;`

> Compiler can see that in no case can an object referred to by b can be of class c. So it is a compile time error.

~~**E.**~~ `b = a;`

> It will fail at compile time because a is of class A and can potentially refer to an object of class A, which cannot be assigned to b, which is a variable of class B. To make it compile, you have to put an explicit cast, which assures the compiler that a will point to an object of class B (or a subclass of B) at run time. Note that, in this case, an explicit cast can take it through the compiler but it will then fail at run time because a does not actually refer to an object of class B (or a subclass of B), so the JVM will throw a ClassCastException.

Explanation:

The statements `c = b` and `b = c` are illegal, since neither of the classes C and B is a subclass of the other. Even though a cast is provided, the statement `c = (C) b` is illegal because the object referred to by b cannot ever be of type C.

Back to Question without Answer

52. QID - 2.1025 : Handling Exceptions

Identify the correct constructs.

Correct Option is : A

A.

```
try {
  for( ;; );
}finally { }
```

A try block must be accompanied by either a catch block or a finally block or both.

~~B.~~

```
try {
  File f = new File("c:\a.txt");
} catch {  f = null; }
```

Invalid syntax for catch. A catch must have a exception: catch(SomeException se){ }

~~C.~~

```
int k = 0;
try {
  k = callValidMethod();
}
System.out.println(k);
catch {  k = -1; }
```

There cannot be any thing between a catch or a finally block and the closing brace of the previous try or catch block.

~~D.~~

```
try {
  try {
     Socket s = new ServerSocket(3030);
  }catch(Exception e) {
    s = new ServerSocket(4040);
  }
}
```

The first try doesn't have any catch or finally block.

~~E.~~

```
try {
       s = new ServerSocket(3030);
}
catch(Exception t){ t.printStackTrace(); }
catch(IOException e) {
    s = new ServerSocket(4040);
}
catch(Throwable t){ t.printStackTrace();  }
```

You can have any number of catch blocks in any order but each catch must be of a different type. Also, a catch for a subclass exception should occur before a catch block for the superclass exception. Here, IOException is placed before Throwable, which is good but Exception is placed before IOException, which is invalid and will not compile.

~~F.~~

```
int x = validMethod();
try {
  if(x == 5) throw new IOException();
  else if(x == 6) throw new Exception();
}finally {
 x = 8;
}
```

```
catch(Exception e){ x = 9; }
```

finally cannot occur before any catch block.

Explanation:

Note that a try with resources block may or may not to have any catch or finally block at all. However, try with resources is not in scope for this exam.

Back to Question without Answer

53. QID - 2.962 : Overloading methods

Consider the following method...

```
public int setVar(int a, int b, float c) { ...}
```

Which of the following methods correctly overload the above method?

Correct Options are : A E

A.
```
public int setVar(int a, float b, int c){
  return (int)(a + b + c);
}
```

~~**B.**~~
```
public int setVar(int a, float b, int c){
  return this(a, c, b);
}
```

this(...) can only be called in a constructor and that too as a first statement.

~~**C.**~~
```
public int setVar(int x, int y, float z){
  return x+y;
}
```

It will not compile because it is same as the original method. The name of parameters do not matter.

~~**D.**~~
```
public float setVar(int a, int b, float c){
  return c*a;
}
```

It will not compile as it is same as the original method. The return type does not matter.

E.
```
public float setVar(int a){
  return a;
}
```

Explanation:

A method is said to be overloaded when the other method's name is same and parameters (either the number or their order) are different.
Option 2 is not valid Because of the line: `return this(a, c, b);` This is the syntax of calling a constructor and not a method. It should have been: `return this.setVar(a, c, b);`

Back to Question without Answer

54. QID - 2.1210 : Handling Exceptions

Consider the following code snippet:

```
void m1() throws Exception{
   try{
      // line1
   }
   catch (IOException e){
       throw new SQLException();
   }
   catch(SQLException e){
       throw new InstantiationException();
   }
   finally{
      throw new CloneNotSupportedException();   // this is not a Runt
   }
}
```

Which of the following statements are true?

Correct Options are : B D

~~A.~~ If IOException gets thrown at line1, then the whole method will end up throwing SQLException.

B. If IOException gets thrown at line1, then the whole method will end up throwing CloneNotSupportedException.

~~C.~~ If IOException gets thrown at line1, then the whole method will end up throwing InstantiationException()

D. If no exception is thrown at line1, then the whole method will end up throwing

CloneNotSupportedException.

~~E.~~ If SQLException gets thrown at line1, then the whole method will end up throwing InstantiationException()

Explanation:

The fundamental concepts at play here are:

1. The Exception that is thrown in the last, is the Exception that is thrown by the method to the caller.
So, when no exception or any exception is thrown at line 1, the control goes to finally or some catch block. Now, even if the catch blocks throws some exception, the control goes to finally. The finally block throws CloneNotSupportedException, so the method ends up throwing CloneNotSupportedException. Other exceptions thrown by the code prior to this point are lost.

2. Exception thrown by a catch cannot be caught by the following catch blocks at the same level. So, if IOException is thrown at line 1, the control goes to first catch which throws SQLException. Now, although there is a catch for SQLException, it won't catch the exception because it is at the same level. So, the control goes to the finally and same process as explained above continues. Any exceptions thrown before this exception are lost.

Back to Question without Answer

55. QID - 2.1187 : Working with Java Data Types - String, StringBuilder

Which of the following methods can be called on a String object?

Correct Options are : A B D

A. `substring(int i)`

> returns substring starting from i to end.

B. `substring(int i, int j)`

> returns substring starting from i to j-1.

C. `substring(int i, int j, int k)`

D. `equals(Object o)`

> Since Object class has this method, every java class inherits it.

Back to Question without Answer

56. QID - 2.1285 : Working with Java Data Types - String, StringBuilder

Which of the following operators can be used in conjunction with a String object?

Correct Options are : A C D

A. +

~~**B.** ++~~

C. +=

D. .

~~**E.** *~~

Explanation:

Only + is overloaded for String. a+=x is actually converted to a = a + x. so it is valid for Strings. dot (.) operator accesses members of the String object. There is only one member variable though: CASE_INSENSITIVE_ORDER. It is of type Comparator (which is an interface).

Back to Question without Answer

57. QID - 2.1180 : Using Operators and Decision Constructs

The following code snippet will not compile:

```
int i = 10;
System.out.println( i<20 ? out1() : out2() );
```

Assume that out1 and out2 have method signature: public void out1(); and public void out2();

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

Note that it is not permitted for either the second or the third operand expression of the ? operator to be an invocation of a void method.

If one of the operands is of type byte and the other is of type short, then the type of the conditional expression is short.

If one of the operands is of type T where T is byte, short, or char, and the other operand is a constant expression of type int whose value is representable in type T, then the type of the conditional expression is T.

Otherwise, binary numeric promotion (5.6.2) is applied to the operand types, and the type of the conditional expression is the promoted type of the second and third operands.

If one of the second and third operands is of the null type and the type of the other is a reference type, then the type of the conditional expression is that reference type.

If the second and third operands are of different reference types, then it must be possible to convert one of the types to the other type (call this latter type T) by assignment conversion (5.2); the type of the conditional expression is T. It is a compile-time error if neither type is assignment compatible with the other type.

Back to Question without Answer

58. QID - 2.1335 : Using Operators and Decision Constructs

Consider the code shown below:

```
public class TestClass{
  public static int switchTest(int k){
     int j = 1;
     switch(k){
        case 1: j++;
        case 2: j++;
        case 3: j++;
        case 4: j++;
        case 5: j++;
        default : j++;
     }
     return j + k;
  }
  public static void main(String[] args){
     System.out.println( switchTest(4) );
  }
}
```

What will it print when compiled and run?

Correct Option is : D

~~**A.**~~ 5

~~**B.**~~ 6

~~**C.**~~ 7

D. 8

~~E.~~ 9

Explanation:

The control in the case falls through till reaches the break statement.
Here, switch(4) will take the control to case 4:.
Now since there is no break statement, all the statements till the end will be executed.
So j will be incremented 3 time making it 4. finally 4 + 4 i.e. 8 will be returned.

Back to Question without Answer

59. QID - 2.938 : Creating and Using Arrays

What will the following program print?

```
class Test{
   public static void main(String[] args){
      int i = 4;
      int ia[][][] = new int[i][i = 3][i];
      System.out.println( ia.length + ", " + ia[0].length+", "+ ia[0
   }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

~~**B.**~~ 3, 4, 3

~~**C.**~~ 3, 3, 3

~~**D.**~~ 4, 3, 4

E. 4, 3, 3

Explanation:

In an array creation expression, there may be one or more dimension expressions, each within brackets. Each dimension expression is fully evaluated before any part of any dimension expression to its right. The first dimension is calculated as 4 before the second dimension expression sets 'i' to 3.

Note that if evaluation of a dimension expression completes abruptly, no part of any dimension expression to its right will appear to have been evaluated.

Back to Question without Answer

60. QID - 2.1223 : Handling Exceptions

Consider the following hierarchy of Exception classes :

```
java.lang.RuntimeException
  +-------- IndexOutOfBoundsException
                  +---------ArrayIndexOutOfBoundsException,
StringIndexOutOfBoundsException
```

Which of the following statements are correct for a method that can throw `ArrayIndexOutOfBounds` as well as `StringIndexOutOfBounds` Exceptions but does not have try catch blocks to catch the same?

Correct Options are : B D E

~~**A.**~~ The method calling this method will either have to catch these 2 exceptions or declare them in its `throws` clause.

B. It is ok if it declares just `throws ArrayIndexOutOfBoundsException`

~~**C.**~~ It must declare `throws ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException`

D. It is ok if it declares just `throws IndexOutOfBoundsException`

E. It does not need to declare any throws clause.

Explanation:

Note that both the exceptions are `RuntimeExceptions` so there is no need to catch

these. But it is ok even if the method declares them explicitly.

Back to Question without Answer

61. QID - 2.927 : Using Operators and Decision Constructs

What will the following code print ?

```
class Test{
   public static void main(String[] args){
      int k = 1;
      int[] a = { 1 };
      k += (k = 4) * (k + 2);
      a[0] += (a[0] = 4) * (a[0] + 2);
      System.out.println( k + " , " + a[0]);
   }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ 4 , 4

C. 25 , 25

~~**D.**~~ 13 , 13

~~**E.**~~ None of the above.

Explanation:

The value `1` of `k` is saved by the compound assignment operator `+=` before its right-hand operand `(k = 4) * (k + 2)` is evaluated. Evaluation of this right-hand operand then assigns `4` to `k`, calculates the value `6` for `k + 2`, and then multiplies `4` by

`6` to get `24`. This is added to the saved value 1 to get 25, which is then stored into `k` by the `+=` operator. An identical analysis applies to the case that uses `a[0]`.

```
k += (k = 4) * (k + 2);
a[0] += (a[0] = 4) * (a[0] + 2);
k = k + (k = 4) * (k + 2);
a[0] = a[0] + (a[0] = 4) * (a[0] + 2);
```

Back to Question without Answer

62. QID - 2.905 : Working with Inheritance

Given the following line of code:

```
List students = new ArrayList();
```

Identify the correct statement:

Correct Option is : A

A. The reference type is List and the object type is ArrayList.

> Since you are doing new ArrayList, you are creating an object of class ArrayList. You are assigning this object to variable "students", which is declared of class List. Reference type means the declared type of the variable.

~~**B.**~~ The reference type is ArrayList and the object type is ArrayList.

~~**C.**~~ The reference type is ArrayList and the object type is List.

~~**D.**~~ The reference type is List and the object type is List.

Back to Question without Answer

63. QID - 2.1258 : Java Basics

Consider the following code:

```
class Test{
  public static void main(String[] args){
    for (int i = 0; i < args.length; i++)   System.out.print(i == 0
? args[i] : " " + args[i]);
  }
}
```

What will be the output when it is run using the following command:

java Test good bye friend!

Correct Option is : A

A. good bye friend!

~~**B.**~~ good good good

~~**C.**~~ goodgoodgood

~~**D.**~~ good bye

~~**E.**~~ None of the above.

Explanation:

The arguments passed on the command line can be accessed using the args array. The first argument (i.e. good) is stored in args[0], second argument (i.e. bye) is stored in

args[1] and so on.

Here, we are passing 3 arguments. Therefore, args.length is 3 and the for loop will run 3 times. For the first iteration, i is 0 and so the first operand of the ternary operator (?) will be returned, which is args[i]. For the next two iterations, " "+args[i] will be returned. Hence, the program will print three strings: "good", " bye", and " friend" on the same line.

Notice that unlike in C++, program name is not the first parameter in the argument list. Java does not need to know the program name because the .class file name and the java class name are always same (for a public class). So the java code always knows the program name it is running in. So there is no need to pass the program name as the first parameter of the argument list.
Note that in C/C++, the binary file name may be anything so the code does not know what binary file it is going to end up in. That's why the program name is also sent (automatically) in parameter list.

Back to Question without Answer

64. QID - 2.1054 : Working with Java Data Types - String, StringBuilder

What will be written to the standard output when the following program is run?

```
public class TrimTest{
   public static void main(String args[]){
      String blank  = " ";  // one space
      String line = blank + "hello" + blank + blank;
      line.concat("world");
      String newLine  =  line.trim();
      System.out.println((int)(line.length() + newLine.length()));
   }
}
```

Correct Option is : E

~~**A.**~~ 25

~~**B.**~~ 24

~~**C.**~~ 23

~~**D.**~~ 22

E. None of the above.

It will print 13 !!!

Explanation:

Note that line.concat("world") does not change line itself. It creates a new String object containing " hello world " but it is lost because there is no reference to it.

Similarly, calling trim() does not change the object itself.
So the answer is 8 + 5 = 13 !

Back to Question without Answer

65. QID - 2.898 : Working with Java Data Types - Variables and Objects

Given:

```
public class Employee{
    String name;
    public Employee(){
    }
}
```

Which of the following lines creates an Employee instance?

Correct Option is : B

~~A.~~ `Employee e;`

> This declares a variable of class Employee but does not create any object.

B. `Employee e = new Employee();`

> Using the new operator is the right way to create an object.

~~C.~~ `Employee e = Employee.new();`

~~D.~~ `Employee e = Employee();`

Back to Question without Answer

66. QID - 2.1148 : Handling Exceptions

What will the following code print?

```
public class Test{
   public int luckyNumber(int seed){
      if(seed > 10) return seed%10;
         int x = 0;
            try{
               if(seed%2 == 0) throw new Exception("No Even no.");
               else return x;
            }
            catch(Exception e){
               return 3;
            }
            finally{
               return 7;
            }
         }

        public static void main(String args[]){
           int amount = 100, seed = 6;
           switch( new Test().luckyNumber(6) ){
               case 3: amount = amount * 2;
               case 7: amount = amount * 2;
               case 6: amount = amount + amount;
               default :
           }
          System.out.println(amount);
       }
}
```

Correct Option is : E

~~A.~~ It will not compile.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ 800

~~**D.**~~ 200

E. 400

Explanation:

When you pass 6 to `luckyNumber()`, `if(seed%2 == 0) throw new Exception("No Even no.");` is executed and the exception is caught by the catch block where it tries to `return 3;` But as there is a finally associated with the try/catch block, it is executed before anything is returned. Now, as finally has `return 7;`, this value supersedes 3.
In fact, this method will always return 7 if seed <= 10.

Now, in the switch there is no break statement. So both -
`case 7: amount = amount * 2;`
and
`case 6: amount = amount + amount;`
are executed. so the final amount becomes 400.

Back to Question without Answer

67. QID - 2.944 : Java Basics

Which of the following are valid declarations:

Correct Options are : B C D

~~**A.** int a = b = c = 100;~~

B. int a, b, c; a = b = c = 100;

C. int a, b, c=100;

D. int a=100, b, c;

~~**E.** int a= 100 = b = c;~~

Explanation:

Java does not allow chained initialization in declaration so option 1 and 5 are not valid.

Back to Question without Answer

68. QID - 2.1022 : Working with Java Data Types - String, StringBuilder

What will be the output of the following program (excluding the quotes)?

```
public class SubstringTest{
   public static void main(String args[]){
      String String = "string isa string";
      System.out.println(String.substring(3, 6));
   }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

String String = "String"; is a perfectly valid syntax!

~~**B.**~~ "ing is"

~~**C.**~~ "ing isa"

~~**D.**~~ "ing " (There is a space after g)

E. None of the above.

It will print 'ing'. (No space after 'g')

Explanation:

Remember, indexing always starts from 0.
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"

Parameters:
beginIndex - the beginning index, inclusive.
endIndex - the ending index, exclusive.
Returns:
the specified substring.
Throws:
IndexOutOfBoundsException - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

Back to Question without Answer

69. QID - 2.1114 : Using Loop Constructs

What will the following code print?

```
public class TestClass{
        int x = 5;
        int getX(){ return x; }

        public static void main(String args[]) throws Exception{
            TestClass tc = new TestClass();
            tc.looper();
            System.out.println(tc.x);
        }

        public void looper(){
            int x = 0;
            while( (x = getX()) != 0 ){
                for(int m = 10; m>=0; m--){
                    x = m;
                }
            }

       }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ It will print 0.

~~**D.** It will print 5.~~

E. None of these.

This program will compile and run but will never terminate.

Explanation:

Note that looper() declares an automatic variable x, which shadows the instance variable x. So when x = m; is executed, it is the local variable x that is changed not the instance field x. So getX() never returns 0. If you remove int x = 0; from looper(), it will print 0 and end.

Back to Question without Answer

70. QID - 2.1037 : Working with Inheritance

What will the following code print when compiled and run?

```
class Base{
   void methodA(){
      System.out.println("base - MethodA");
   }
}

class Sub extends Base{
   public void methodA(){
      System.out.println("sub - MethodA");
   }
   public void methodB(){
      System.out.println("sub - MethodB");
   }
   public static void main(String args[]){
      Base b=new Sub(); //1
      b.methodA(); //2
      b.methodB(); //3
   }
}
```

Correct Option is : E

~~**A.**~~ `sub - MethodA` and `sub - MethodB`

~~**B.**~~ `base - MethodA` and `sub - MethodB`

~~**C.**~~ Compile time error at `//1`

~~**D.**~~ Compile time error at `//2`

E. Compile time error at `//3`

Explanation:

The point to understand here is, `b` is declared to be a reference of class Base and `methodB()` is not defined in Base. So the compiler cannot accept the statement `b.methodB()` because it only verifies the validity of a call by looking at the declared class of the reference.
For example, the compiler is able to verify that `b.methodA()` is a valid call because class Base has method `methodA`. But it does not "bind" the call. Call binding is done at runtime by the jvm and the jvm looks for the actual class of object referenced by the variable before invoking the method.

Back to Question without Answer

71. QID - 2.1338 : Java Basics

What is the correct parameter specification for the standard main method?

Correct Options are : B E

~~**A.** void~~

B. String[ ] args

~~**C.** Strings args[ ]~~

~~**D.** String args~~

E. String args[ ]

Explanation:

There is a no difference for `args` whether it is defined as `String[] args` or `String args[]`. However, there is an important difference in the way it is defined as illustrated by the following:

1. `String[] sa1, sa2;`
Here, both - `sa1` and `sa2` are String arrays.

2. `String sa1[], sa2;`
Here, only `sa1` is a String array. `sa2` is just a String.

Back to Question without Answer

72. QID - 2.903 : Working with Methods

A java source file contains the following code:

```
interface I {
  int getI(int a, int b);
}

interface J{
    int getJ(int a, int b, int c);
}

abstract class MyIJ implements J , I { }

class MyI{
    int getI(int x, int y){ return x+y; }
}

interface K extends J{
    int getJ(int a, int b, int c, int d);
}
```

Identify the correct statements:

Correct Option is : F

~~**A.**~~ It will fail to compile because of `MyIJ`

> `MyIJ` declares that it implements interfaces I and J, but does not implement the methods declared in these interfaces. However, since `MyIJ` has been declared as `abstract`, it is valid.

~~**B.**~~ It will fail to compile because of `MyIJ` and `K`

~~**C.**~~ It will fail to compile because of `K`

K is a valid interface because an interface is permitted to extend another interface.

D. It will fail to compile because of `MyI` and `K`

Both are valid.

E. It will fail to compile because of `MyIJ, K,` and `MyI`

F. It will compile without any error.

Back to Question without Answer

73. QID - 2.1318 : Java Basics

Which of the following statements is correct?

Correct Option is : E

~~**A.**~~ new, delete and goto are keywords in the Java language

'delete' is not a keyword.

~~**B.**~~ try, catch and thrown are keywords in the Java language

'thrown' is not, though 'throws' is.

~~**C.**~~ static, unsigned and long are keywords in the Java language

There is no 'unsigned'. All primitive types are 'signed' except 'char'.

~~**D.**~~ exit, class and while are keywords in the Java language

'exit' is a method name and is not a keyword.

E. return, goto and default are keywords in the Java language

although not used, 'goto' is a reserved word.

Explanation:

Following is a list of Java keywords :

abstract continue for new switch
assert default goto package synchronized

boolean do if private this
break double implements protected throw
byte else import public throws
case enum instanceof return transient
catch extends int short try
char final interface static void
class finally long strictfp volatile
const float native super while

Technically 'true', 'false' and 'null' are literals but for the purpose of the exam consider them as keywords.

Back to Question without Answer

74. QID - 2.1340 : Overloading methods

Given the following code, which method declarations can be inserted at line 1 without any problems?

```
public class OverloadTest{
    public int sum(int i1, int i2) { return i1 + i2; }
    // 1
}
```

Correct Options are : B C D

~~**A.**~~ `public int sum(int a, int b) { return a + b; }`

> Will cause duplicate method. Variable names don't matter. Only their types.

B. `public int sum(long i1, long i2) { return (int) i1; }`

C. `public int sum(int i1, long i2) { return (int) i2; }`

D. `public long sum(long i1, int i2) { return i1 + i2; }`

~~**E.**~~ `public long sum(int i1, int i2) { return i1 + i2; }`

> Only the return type is different so the compiler will complain about having duplicate method sum.

Explanation:

The rule is that you cannot have methods that create ambiguity for the compiler in a

class. It is illegal for a class to have two methods having same name and having same type of input parameters in the same order.
Name of the input variables and return type of the method are not looked into.
1. Option 1 is wrong because, then both the methods will be same (as their method name and the class/type and order of the input parameters will be same). So this amounts to duplicate method which is not allowed.
As mentioned, name of the input parameters does not matter. Only the type of parameters and their order matters.
2. 2 is valid because the type of input parameters are different. So this is a different method and does not amount to duplication.
3 and 4 are valid for the same reason
5 is not valid because it leads to duplicate method(as their method name and the class/type and order of the input parameters will be same). Note that as mentioned in the comments, return type does not matter.

Back to Question without Answer

75. QID - 2.1299 : Using Loop Constructs

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int var = 20, i=0;
      do{
         while(true){
         if( i++ > var) break;
         }
      }while(i<var--);
      System.out.println(var);
   }
}
```

Correct Option is : A

A. 19

~~**B.**~~ 20

~~**C.**~~ 21

~~**D.**~~ 22

~~**E.**~~ It will enter an infinite loop.

Explanation:

When the first iteration of outer do-while loop starts, var is 20. Now, the inner loop

executes till i becomes 21.
Now, the condition for outer do-while is checked, while(22 < 20), [i is 22 because of the last i++>var check], thereby making var 19. And as the condition is false, the outer loop also ends.
So, 19 is printed.

Back to Question without Answer

76. QID - 2.1183 : Using Operators and Decision Constructs

Given the following LOCs:

```
int rate = 10;
XXX amount = 1 - rate/100*1 - rate/100;
```

What can XXX be?

Correct Option is : F

~~**A.**~~ only int or long

~~**B.**~~ only long or double

~~**C.**~~ only double

~~**D.**~~ double or float

~~**E.**~~ long or double but not int or float.

F. int, long, float or double

Explanation:

Note that none of the terms in the expression `1 - rate/100*1 - rate/100;` is double or float. They are all ints. So the result of the expression will be an int.
Since an int can be assigned to a variable of type int, long, float or double, amount can be int, long, float or double.

Back to Question without Answer

77. QID - 2.1170 : Java Basics

Consider the classes shown below:

```
class A{
   public A() { }
   public A(int i) {   System.out.println(i );    }
}
class B{
   static A s1 = new A(1);
   A a = new A(2);
   public static void main(String[] args){
      B b = new B();
      A a = new A(3);
   }
   static A s2 = new A(4);
}
```

Which is the correct sequence of the digits that will be printed when B is run?

Correct Option is : B

~~**A.**~~ 1 ,2 ,3 4.

B. 1 ,4, 2 ,3

~~**C.**~~ 3, 1, 2, 4

~~**D.**~~ 2, 1, 4, 3

~~**E.**~~ 2, 3, 1, 4

Explanation:

The order of initialization of a class is:
1. All static constants, variables and blocks.(Among themselves the order is the order in which they appear in the code.)
2. All non static constants, variables and blocks.(Among themselves the order is the order in which they appear in the code.)
3. Constructor.

Back to Question without Answer

78. QID - 2.955 : Java Basics

An instance member ...

Correct Options are : A D

A. can be a variable, a constant or a method.

~~**B.**~~ is a variable or a constant.

~~**C.**~~ belongs to the class.

D. belongs to an instance of the class.

~~**E.**~~ is same as a local variable.

> variables defined in methods are called local variables (also known as automatic variables) where as instance members are defined in the class scope.

Explanation:

An instance member belongs to a single instance, not the class as a whole. An instance member is a member variable or a member method that belongs to a specific object instance. All non-static members are instance members.

Back to Question without Answer

79. QID - 2.917 : Working with Java Data Types - Variables and Objects

Given:

```
public class Square {
    private double side = 0;  // LINE 2

    public static void main(String[] args) {   // LINE 4
        Square sq = new Square();  // LINE 5
        side = 10;  // LINE 6
   }
}
```

What can be done to make this code compile and run?

Correct Option is : D

~~**A.**~~ replace // LINE 2 with:

```
private int side = 0;
```

~~**B.**~~ replace // LINE 2 with:

```
public int side = 0;
```

~~**C.**~~ replace // LINE 5 with:

```
double sq = new Square();
```

D. replace // LINE 6 with:

```
sq.side = 10;
```

`side` is not a global variable that you can access directly (Note that Java doesn't have the concept of a global variable). `side` is a field in `Square` class. So you need to specify which Square object's `side` you are trying to access.

An integer can be assigned to a double but not vice versa.

Back to Question without Answer

80. QID - 2.1133 : Handling Exceptions

Which of the following can be thrown using a throw statement?

Correct Options are : C D E

~~**A.**~~ Event

~~**B.**~~ Object

C. Throwable

D. Exception

E. RuntimeException

Explanation:

You can only throw a Throwable using a throws clause. Exception and Error are two main subclasses of Throwable.

Back to Question without Answer

81. QID - 2.1262 : Creating and Using Arrays

Consider the following code snippet ...

```
boolean[] b1 = new boolean[2];
boolean[] b2 = {true , false};
System.out.println( "" + (b1[0] == b2[0]) + ", "+ (b1[1] ==
b2[1])  );
```

What will it print ?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw ArrayIndexOutOfBoundsError at Runtime.

C. false, true

~~**D.**~~ true, false

~~**E.**~~ It will print false, false.

Explanation:

Note that whenever you create an array all of its elements are automatically given defaults values. Numeric types are initialized to 0, objects are initialized to null, and booleans to false.

So if you have, float[ ] f = new float[3]; f[0], f[1] and f[2] will all be 0.0.

if you have Object[ ] o = new String[3]; o[0], o[1] and o[2] will all be null.
In this case, b1[0] and b1[1] are false.
whereas b2[0] and b2[1] are true and false.
So the answer is false and true.

Back to Question without Answer

82. QID - 2.1014 : Using Operators and Decision Constructs

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     Object obj1 = new Object();
     Object obj2 = obj1;
     if( obj1.equals(obj2) ) System.out.println("true");
     else  System.out.println("false");
  }
}
```

Correct Option is : A

A. true

~~**B.**~~ false

~~**C.**~~ It will not compile.

~~**D.**~~ It will compile but throw an exception at run time.

~~**E.**~~ None of the above.

Explanation:

Object class's `equals()` method just checks whether the two references are pointing to the same location or not. In this case they really are pointing to the same location because of `obj2 = obj1;` so it returns `true`.

Back to Question without Answer

83. QID - 2.877 : Encapsulation

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){        this.side = length;    }

    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

    double getArea() {   return area;   }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Correct Options are : A B C D

A. Add a calculateArea method:

```
private void calculateArea(){
  this.area = this.side*this.side;
}
```

B. Make `side` and `area` fields private.

C. Change setSide method to:

```
public void setSide(double d){
this.side = d;
calculateArea();
}
```

D. Make the `getArea` method public.

~~**E.**~~ Add a `setArea() method:`

```
public void setArea(double d){ area = d; }
```

> This is not required because area is calculated using the side. So if you allow other classes to set the area, it could make side and area inconsistent with each other.

Explanation:

There can be multiple ways to accomplish this. The exam asks you questions on the similar pattern.
The key is that your data variable should be private and the functionality that is to be exposed outside should be public. Further, your setter methods should be coded such that they don't leave the data members inconsistent with each other.

Back to Question without Answer

84. QID - 2.1251 : Working with Java Data Types - Variables and Objects

What will be the result of attempting to compile and run the following code?

```
public class InitClass{
   public static void main(String args[ ] ){
      InitClass obj = new InitClass(5);
   }
   int m;
   static int i1 = 5;
   static int i2 ;
   int  j = 100;
   int x;
   public InitClass(int m){
      System.out.println(i1 + "  " + i2 + "   " + x + "  " + j + "  '
   }
  { j = 30; i2 = 40; }  // Instance Initializer
   static { i1++; }      // Static Initializer
}
```

Correct Option is : C

~~**A.**~~ The code will fail to compile, since the instance initializer tries to assign a value to a static member.

~~**B.**~~ The code will fail to compile, since the member variable x will be uninitialized when it is used.

C. The code will compile without error and will print 6, 40, 0, 30, 5 when run.

~~**D.**~~ The code will compile without error and will print 5, 0, 0, 100, 5 when run.

~~E.~~ The code will compile without error and will print 5, 40, 0, 30, 0 when run.

Explanation:

The value 5 is passed to the constructor to the local (automatic) variable m. So the instance variable m is shadowed. Before the body of the constructor is executed, the instance initializer is executed and assigns values 30 and 40 to variables j and i2, respectively. A class is loaded when it is first used. For example,

```
class A1{
  static int i = 10;
  static { System.out.println("A1 Loaded "); }
}
public class A{
  static { System.out.println("A Loaded "); }
  public static void main(String[] args){
    System.out.println(" A should have been loaded");
    A1 a1 = null;
    System.out.println(" A1 should not have been loaded");
    System.out.println(a1.i);
  }
}
```

When you run it you get the output:

```
A Loaded
 A should have been loaded
 A1 should not have been loaded
A1 Loaded
10
```

Now, A should be loaded first as you are using its main method. Even though you are doing `A1 a1 = null;` A1 will not be loaded as it is not yet used (so the JVM figures out that it does not need to load it yet.) When you do a1.i, you are using A1, so before you use it, it must be loaded. That's when A1 is loaded. Finally 10 is printed.

Back to Question without Answer

85. QID - 2.1068 : Creating and Using Arrays

Which of the following statements are valid ?

Correct Options are : B D

~~**A.**~~ `String[ ] sa = new String[3]{ "a", "b", "c"};`

> You cannot specify the length of the array (i.e. 3, here) if you are using the initializer block while declaring the array.

B. `String sa[ ] = { "a ", " b", "c"};`

~~**C.**~~ `String sa = new String[ ]{"a", "b", "c"};`

> here `sa` is not declared as array of strings but just as a String.

D. `String sa[ ] = new String[ ]{"a", "b", "c"};`

~~**E.**~~ `String sa[ ] = new String[ ] {"a" "b" "c"};`

> There are no commas separating the strings.

Back to Question without Answer

86. QID - 2.1087 : Using Operators and Decision Constructs

Which of the given lines can be inserted at //1 of the following program ?

```
public class TestClass{
   public static void main(String[] args){
     short s = 9;
     //1
   }
}
```

Correct Options are : D G

~~**A.**~~ `Short k = new Short(9); System.out.println(k instanceof Short);`

> 9 is considered an int. This should be: `Short s = new Short( (short) 9 );`

~~**B.**~~ `System.out.println(s instanceof Short);`

> The left operand of instanceof MUST be an object and not a primitive.

~~**C.**~~ `Short k = 9; System.out.println( k instanceof s);`

> Right operand of instanceof MUST be a class name.

D. `int i = 9; System.out.println(s == i);`

> Any two integral primitives can be compared using == operator.

~~**E.**~~ `Boolean b = s instanceof Number;`

> Left operand of instanceof MUST be an object and not a primitive.

F. ~~`Short k = 9; Integer i = 9; System.out.println(k == i);`~~

This will not compile because k and i are referring to objects that have no IS-A relationship among themselves.

G. `Integer i = 9; System.out.println( s == i );`

Back to Question without Answer

87. QID - 2.1332 : Working with Java Data Types - Variables and Objects

What happens when you try to compile and run the following program?

```
public class CastTest{
   public static void main(String args[ ] ){
      byte b = -128 ;
      int i = b ;
      b = (byte) i;
      System.out.println(i+" "+b);
   }
}
```

Correct Option is : B

~~**A.**~~ The compiler will refuse to compile it because i and b are of different types cannot be assigned to each other.

B. The program will compile and will print -128 and -128 when run .

> A byte can ALWAYS be assigned to an int.

~~**C.**~~ The compiler will refuse to compile it because -128 is outside the legal range of values for a byte.

> Range of byte is -128 to 127

~~**D.**~~ The program will compile and will print 128 and -128 when run .

~~**E.**~~ The program will compile and will print 255 and -128 when run .

Explanation:

`byte` and `int` both hold signed values. So when `b` is assigned to `i`, the sign is preserved.

Back to Question without Answer

88. QID - 2.1191 : Creating and Using Arrays

Which of the following code fragments will successfully initialize a two-dimensional array of chars named cA with a size such that cA[2][3] refers to a valid element?

```
1.
  char[][] cA = {  { 'a', 'b', 'c' },  { 'a', 'b', 'c' }   };
2.
  char cA[][] = new char[3][];
  for (int i=0; i<cA.length; i++) cA[i] = new char[4];
3.
  char cA[][] = { new char[ ]{ 'a', 'b', 'c' }  ,   new char[ ]{
'a', 'b', 'c' }  };
4
  char cA[3][2] = new char[][] {  { 'a', 'b', 'c' },   { 'a', 'b',
'c' }   };
5.
  char[][] cA = { "1234", "1234",  "1234"  };
```

Correct Option is : E

~~**A.**~~ 1, 3

~~**B.**~~ 4, 5

~~**C.**~~ 2, 3

~~**D.**~~ 1, 2, 3

E. 2

Explanation:

1 and 3 declare a two dimensional array alright but they create the array of size 2, 3. And cA[2][3] means we need an array of
size 3, 4 because the numbering starts from 0.
4 : You cannot put array size information on LHS.
5 : This is a one dimensional array and that too of strings. Note that a java String is not equivalent to 1 dimensional array of chars.
This leaves us with only one choice 2.

Back to Question without Answer

89. QID - 2.952 : Working with Inheritance

Which of the following lines of code that, when inserted at line 1, will make the overriding method in SubClass invoke the overridden method in BaseClass on the current object with the same parameter.

```
class BaseClass{
   public void print(String s) {  System.out.println("BaseClass :"+s)
}
class SubClass extends BaseClass{
   public void print(String s){
      System.out.println("SubClass :"+s);
      // Line 1
   }
   public static void main(String args[]){
      SubClass sc = new SubClass();
      sc.print("location");
   }
}
```

Correct Option is : B

A. ~~`this.print(s);`~~

B. `super.print(s);`

This is the right syntax to call the base class's overridden method. However, note that there is no way call a method if it has been overriden more than once. For example, if you make BaseClass extend from another base class SubBase, and if SubBase also has the same method, then there is no way to invoke SubBase's print method from SubClass's print method. You cannot have something like super.super.print(s);

~~C.~~ `print(s);`

This will call the same method and will cause a recursion.

~~D.~~ `BaseClass.print(s);`

print is not a static method.

Back to Question without Answer

90. QID - 2.1282 : Using Operators and Decision Constructs

What, if anything, is wrong with the following code?

```
void test(int x){
   switch(x){
      case 1:
      case 2:
      case 0:
      default :
      case 4:
   }
}
```

Correct Option is : E

~~**A.**~~ Data Type of 'x' is not valid to be used as an expression for the switch clause.

> x is an int and int is perfectly valid. long, double, boolean, and float are not valid.

~~**B.**~~ The case label 0 must precede case label 1.

> While ordering may be important for the logic being implemented in the code, technically, any order is valid.

~~**C.**~~ Each case section must end with a break keyword.

> This is not necessary. If there is no break at the end of a case section, the control will fall through to the next case section (even if the case label doesn't match).

~~**D.**~~ The default label must be the last label in the switch statement.

> Any order of case statements is valid.

E. There is nothing wrong with the code.

Back to Question without Answer

Test 3

01. QID - 2.1091

What will the following code print when TestClass is run?

```
class Employee {

  static int i = 10;  {
      i = 15;
      System.out.print(" Employee "+i);
  }
  static { System.out.print(" Employee static "+i); }
}

class Manager extends Employee {
  static {
    i = 45;
    System.out.print(" Manager static ");
  }{
    i = 30;
    System.out.print(" Manager "+i);
  }
}

class Owner extends Manager{
  static { System.out.println("Owner"); }
}

public class TestClass {
  public static void main(String[] args) {
      Manager m = new Manager();
  }
}
```

Select 1 option

A. `Employee static 10 Manager static Employee 10 Manager 30`

B. `Employee static 10 Manager static Owner Manager 30`

C. `Employee static 10 Manager static Employee 15 Manager 30`

D. `Employee static 10 Manager static 15 Manager 20`

E. It will not compile.

Check Answer

02. QID - 2.1354

Which of the following correctly declare a variable which can hold an array of 10 integers?

Select 2 options

A. `int[ ] iA`

B. `int[10] iA`

C. `int iA[ ]`

D. `Object[ ] iA`

E. `Object[10] iA`

Check Answer

03. QID - 2.921

Consider the following code:

```
class MyClass { }
public class TestClass{
   MyClass getMyClassObject(){
      MyClass mc = new MyClass(); //1
         return mc; //2
   }
   public static void main(String[] args){
      TestClass tc = new TestClass(); //3
      MyClass x = tc.getMyClassObject(); //4
      System.out.println("got myclass object"); //5
      x = new MyClass(); //6
      System.out.println("done"); //7
   }
}
```

After what line the MyClass object created at line 1 will be eligible for garbage collection?

Select 1 option

A. 2

B. 5

C. 6

D. 7

E. Never till the program ends.

Check Answer

04. QID - 2.1111

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 == false){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print `true`

C. It will print `false`

D. Runtime error.

E. It will print nothing.

Check Answer

05. QID - 2.1311

The following class will not throw a `NullPointerException` when compiled and run.

```
class Test{
   public static void main(String[] args) throws Exception{
      int[] a = null;
      int i = a [ m1() ];
   }
   public static int m1() throws Exception{
      throw new Exception("Some Exception");
   }
}
```

Select 1 option

A. True

B. False

Check Answer

06. QID - 2.1085

Consider the following two classes defined in two java source files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1 <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       X x = new X();
       x.apply(LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Select 2 options

A. `import static X;`

B. `import static com.foo.*;`

C. `import static com.foo.X.*;`

D. `import com.foo.*;`

E. `import com.foo.X.LOGICID;`

Check Answer

07. QID - 2.1048

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      try{
         RuntimeException re = null;
         throw re;
      }
      catch(Exception e){
         System.out.println(e);
      }
   }
}
```

Select 1 option

A. The code will fail to compile, since `RuntimeException` cannot be caught by catching an Exception.

B. The program will fail to compile, since `re` is `null`.

C. The program will compile without error and will print `java.lang.RuntimeException` when run.

D. The program will compile without error and will print `java.lang.NullPointerException` when run.

E. The program will compile without error and will run and print `null`.

Check Answer

08. QID - 2.1236

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     int x = 1;
     int y = 0;
     if( x/y ) System.out.println("Good");
     else  System.out.println("Bad");
  }
}
```

Select 1 option

A. Good

B. Bad

C. Exception at runtime saying division by Zero.

D. It will not compile.

E. None of the above.

Check Answer

09. QID - 2.954

What class of objects can be declared by the throws clause?

Select 3 options

A. Exception

B. Error

C. Event

D. Object

E. RuntimeException

Check Answer

10. QID - 2.1150

Which of the following are valid declarations inside an interface?

Select 2 options

A. `void compute();`

B. `public void compute();`

C. `public final void compute();`

D. `static void compute();`

E. `protected void compute();`

Check Answer

11. QID - 2.1242

Given:

```
public class TestClass{
     public static void main(String[] args){
     int i = Integer.parseInt(args[1]);
     System.out.println(args[i]);
   }
}
```

What will happen when you compile and run the above program using the following command line:

```
java TestClass 1 2
```

Select 1 option

A. It will print `1`

B. It will print `2`

C. It will print some junk value.

D. It will throw `ArrayIndexOutOfBoundsException`.

E. It will throw NumberFormatException

Check Answer

12. QID - 2.1147

Given the following code, which statements are true?

```
class A{
   int i;
}
class B extends A{
   int j;
}
```

Select 3 options

A. Class B extends class A.

B. Class B is the superclass of class A.

C. Class A inherits from class B.

D. Class B is a subclass of class A.

E. Objects of class B will always have a member variable named i .

Check Answer

13. QID - 2.1329

Which of the following method definitions will prevent overriding of that method?

Select 4 options

A. `public final void m1()`

B. `public static void m1()`

C. `public static final void m1()`

D. `public abstract void m1()`

E. `private void m1()`

Check Answer

14. QID - 2.959

What will the following class print ?

```
class Test{
   public static void main(String[] args){
      int[][] a = { { 00, 01 }, { 10, 11 } };
      int i = 99;
      try {
         a[val()][i = 1]++;
      } catch (Exception e) {
         System.out.println( i+", "+a[1][1]);
      }
   }
   static int val() throws Exception {
     throw new Exception("unimplemented");
   }
}
```

Select 1 option

A. 99 , 11

B. 1 , 11

C. 1 and an unknown value.

D. 99 and an unknown value.

E. It will throw an exception at Run time.

Check Answer

15. QID - 2.1159

What happens when you try to compile and run the following class...

```
public class TestClass{
      public static void main(String[] args) throws Exception{
            int a = Integer.MIN_VALUE;
            int b = -a;
            System.out.println( a+ "   "+b);
      }
}
```

Select 1 option

A. It throws an `OverFlowException`.

B. It will print two same negative numbers.

C. It will print two different negative numbers.

D. It will print one negative and one positive number of same magnitude.

E. It will print one negative and one positive number of different magnitude.

Check Answer

16. QID - 2.1280

In the following code what will be the output if 0 (integer value zero) is passed to loopTest()?

```
public class TestClass{
   public void loopTest(int x){
      loop: for (int i = 1; i < 5; i++){
         for (int j = 1; j < 5; j++){
            System.out.println(i);
            if (x == 0) {  continue loop;  }
            System.out.println(j);
         }
      }
   }
}
```

Select 1 option

A. The program will not compile.

B. It will print 1 2 3 4

C. It will print 1 1 2 3 4

D. It will print 1 1 2 2 3 3 4 4

E. Produces no output

Check Answer

17. QID - 2.970

Consider the following classes...

```
class Car{
   public int gearRatio = 8;
   public String accelerate() {  return "Accelerate : Car";  }
}
class SportsCar extends Car{
   public int gearRatio = 9;
   public String accelerate() {  return  "Accelerate : SportsCar";
   public static void main(String[] args){
      Car c = new SportsCar();
      System.out.println( c.gearRatio+"  "+c.accelerate() );
   }
}
```

What will be printed when SportsCar is run?

Select 1 option

A. 8 Accelerate : Car

B. 9 Accelerate : Car

C. 8 Accelerate : SportsCar

D. 9 Accelerate : SportsCar

E. None of the above.

Check Answer

18. QID - 2.849

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {
        String[] sa = {"a", "b", "c"};
        for(String s :  sa){
            if("b".equals(s)) continue;
            System.out.println(s);
            if("b".equals(s)) break;
            System.out.println(s+" again");
        }
    }
}
```

Select 1 option

A.
```
a
a again
c
c again
```

B.
```
a
a again
b
```

C.
```
a
a again
b
b again
```

D.
```
c
c again
```

Check Answer

19. QID - 2.1305

Which of these statements are true?

Select 2 options

A. If a RuntimeException is not caught, the method will terminate and normal execution of the thread will resume.

B. An overriding method must declare that it throws the same exception classes as the method it overrides.

C. The main method of a program can declare that it throws checked exceptions.

D. A method declaring that it throws a certain exception class may throw instances of any subclass of that exception class.

E. finally blocks are executed if and only if an exception gets thrown while inside the corresponding try block.

Check Answer

20. QID - 2.936

Consider the following code:

```
class Super{
  static{ System.out.print("super "); }
}
class One{
  static {  System.out.print("one "); }
}
class Two extends Super{
  static {  System.out.print("two "); }
}
class Test{
  public static void main(String[] args){
     One o = null;
     Two t = new Two();
  }
}
```

What will be the output when class Test is run ?

Select 1 option

A. It will print `one, super` and `two`.

B. It will print `one, two` and `super`.

C. It will print `super` and `two`.

D. It will print `two` and `super`

E. None of the above.

Check Answer

21. QID - 2.979

What will be the output when the following code is compiled and run?

```
//in file Test.java
class E1 extends Exception{ }
class E2 extends E1 { }
class Test{
   public static void main(String[] args){
      try{
         throw new E2();
      }
      catch(E1 e){
         System.out.println("E1");
      }
      catch(Exception e){
         System.out.println("E");
      }
      finally{
         System.out.println("Finally");
      }
   }
}
```

Select 1 option

A. It will not compile.

B. It will print E1 and Finally.

C. It will print E1, E and Finally.

D. It will print E and Finally.

E. It will print Finally.

Check Answer

22. QID - 2.1157

What will be the result of attempting to compile and run the following code?

```
class SwitchTest{
   public static void main(String args[]){
      for ( int i = 0 ; i < 3 ; i++){
         boolean flag  = false;
         switch (i){
            flag  = true;
         }
         if ( flag )  System.out.println( i );
      }
   }
}
```

Select 1 option

A. It will print 0, 1 and 2.

B. It will not print anything.

C. Compilation error.

D. Runtime error.

E. None of the above.

Check Answer

23. QID - 2.1213

When a class whose members should be accessible only to members of that class is coded such a way that its members are accessible to other classes as well, this is called ...

Select 1 option

A. strong coupling

B. weak coupling

C. strong typing

D. weak encapsulation

E. weak polymorphism

F. high cohesion

G. low cohesion

Check Answer

24. QID - 2.1197

Which of the following are keywords in Java?

Select 4 options

A. default

B. NULL

C. String

D. throws

E. long

F. strictfp

Check Answer

25. QID - 2.870

What will the following code print when compiled and run?

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<String> al = new ArrayList<String>();
        al.add("111");
        al.add("222");
        System.out.println(al.get(al.size()));
     }
}
```

Select 1 option

A. It will not compile.

B. It will throw a `NullPointerException` at run time.

C. It will throw an `IndexOutOfBoundsException` at run time.

D. `222`

E. `null`

Check Answer

26. QID - 2.1237

Given:

```
enum Season { SUMMER, WINTER, SPRING, FALL }
```

What will the following code print?

```
Season s = Season.SPRING;
    switch(s){
        case SUMMER : System.out.println("SUMMER");
        case default : System.out.println("SEASON");
        case WINTER :  System.out.println("WINTER");
    }
```

Select 1 option

A. `SEASON`

B. `SEASON`
`WINTER`

C. It will not compile.

D. It will not print anything.

Check Answer

27. QID - 2.1369

Given the following declarations, identify which statements will return `true`:

```
Integer i1 = 1;
Integer i2 = new Integer(1);
int i3 = 1;
Byte b1 = 1;
Long g1 = 1L;
```

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Select 2 options

A. `i1 == i2`

B. `i1 == i3`

C. `i1 == b1`

D. `i1.equals(i2)`

E. `i1.equals(g1)`

F. `i1.equals(b1)`

Check Answer

28. QID - 2.969

Which of these array declarations and initializations are NOT legal?

Select 2 options

A.
```
int[ ] i[ ] = { { 1, 2 }, { 1 }, { }, { 1, 2, 3 } } ;
```

B.
```
int i[ ] = new int[2] {1, 2} ;
```

C.
```
int i[ ][ ] = new int[ ][ ] { {1, 2, 3}, {4, 5, 6} } ;
```

D.
```
int i[ ][ ] = { { 1, 2 }, new int[ 2 ] } ;
```

E.
```
int i[4] = { 1, 2, 3, 4 } ;
```

Check Answer

29. QID - 2.1007

How can you declare a method someMethod() such that an instance of the class is not needed to access it and all the members of the same package have access to it.

Select 3 options

A. `public static void someMethod()`

B. `static void someMethod()`

C. `protected static void someMethod()`

D. `void someMethod()`

E. `protected void someMethod()`

F. `public abstract static void someMethod()`

Check Answer

30. QID - 2.1307

Given:

```
double daaa[][][] = new double[3][][];
double d = 100.0;
double[][] daa = new double[1][1];
```

Which of the following will not cause any problem at compile time or runtime?

Select 2 options

A. `daaa[0] = d;`

B. `daaa[0] = daa;`

C. `daaa[0] = daa[0];`

D. `daa[1][1] = d;`

E. `daa = daaa[0]`

Check Answer

31. QID - 2.1146

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    unsigned byte b = 0;
    b--;
    System.out.println(b);
  }
}
```

Select 1 option

A. 0

B. -1

C. 255

D. -128

E. It will not compile.

Check Answer

32. QID - 2.1281

Which of the following statements regarding 'break' and 'continue' are true?

Select 1 option

A. break without a label, can occur only in a switch, while, do, or for statement.

B. continue without a label, can occur only in a switch, while, do, or for statement.

C. break can never occur without a label.

D. continue can never occur WITH a label.

E. None of the above.

Check Answer

33. QID - 2.895

What two changes can you do, independent of each other, to make the following code compile:

```
//assume appropriate imports
class PortConnector {

    public PortConnector(int port) {
        if (Math.random() > 0.5) {
            throw new IOException();
        }

        throw new RuntimeException();
    }
}


public class TestClass {

    public static void main(String[] args) {
        try {
            PortConnector pc = new PortConnector(10);
        } catch (RuntimeException re) {
            re.printStackTrace();
        }
    }
}
```

Select 2 options

A. add `throws IOException` to the `main` method.

B. add `throws IOException` to `PortConnector` constructor.

C. add `throws IOException` to the `main` method as well as to `PortConnector` constructor.

D. Change `RuntimeException` to `java.io.IOException`.

E. add `throws Exception` to `PortConnector` constructor and change `catch(RuntimeException re)` to `catch(Exception re)` in the main method.

Check Answer

34. QID - 2.867

Which of the following keywords may occur multiple times in a Java source file?

Select 4 options

A. import

B. class

C. private

D. package

E. public

Check Answer

35. QID - 2.1154

Consider the following code:

```
class A{
 public XXX m1(int a){
   return a*10/4-30;
 }
}
class A2 extends A{
 public YYY m1(int a){
   return a*10/4.0;
 }
}
```

What can be substituted for XXX and YYY so that it can compile without any problems?

Select 1 option

A. int, int

B. int, double

C. double, double

D. double, int

E. Nothing, they are simply not compatible.

Check Answer

36. QID - 2.1255

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      loop :          // 1
      {
         System.out.println("Loop Lable line");
         try{
            for (  ;  true ;  i++ ){
               if( i >5) break loop;        // 2
            }
         }
         catch(Exception e){
            System.out.println("Exception in loop.");
         }
         finally{
            System.out.println("In Finally");       // 3
         }
      }
   }
}
```

Select 1 option

A. Compilation error at line 1 as this is an invalid syntax for defining a label.

B. Compilation error at line 2 as 'loop' is not visible here.

C. No compilation error and line 3 will be executed.

D. No compilation error and line 3 will NOT be executed.

E. Only the line with the label Loop will be printed.

Check Answer

37. QID - 2.975

In which of these variable declarations, will the variable remain uninitialized unless explicitly initialized?

Select 1 option

A. Declaration of an instance variable of type int.

B. Declaration of a static class variable of type float.

C. Declaration of a local variable of type float.

D. Declaration of a static class variable of class Object

E. Declaration of an instance variable of class Object.

Check Answer

38. QID - 2.1245

Consider the following class :

```
class Test{
   public static void main(String[] args){
      for (int i = 0; i < 10; i++) System.out.print(i + " ");  //1
      for (int i = 10; i > 0; i--) System.out.print(i + " ");  //2
      int i = 20;                  //3
      System.out.print(i + " ");   //4
   }
}
```

Which of the following statements are true?

Select 4 options

A. As such the class will compile and print 20 at the end of its output.

B. It will not compile if line 3 is removed.

C. It will not compile if line 3 is removed and placed before line 1.

D. It will not compile if line 4 is removed and placed before line 3.

E. Only Option 2, 3, and 4 are correct.

Check Answer

39. QID - 2.1370

What will the following code print when run?

```
class Baap {
    public int h = 4;
    public int getH() {
        System.out.println("Baap " + h);
        return h;
    }
}

public class Beta extends Baap {
    public int h = 44;
    public int getH() {
        System.out.println("Beta " + h);
        return h;
    }
    public static void main(String[] args) {
        Baap b = new Beta();
        System.out.println(b.h + " " + b.getH());
        Beta bb = (Beta) b;
        System.out.println(bb.h + " " + bb.getH());
    }
}
```

Select 1 option

A.
```
Beta 44
4 44
Baap 44
44 44
```

B.
```
Baap 44
4 44
Beta 44
44 44
```

C.
```
Beta 44
4 44
Beta 44
4 44
```

D.
```
Beta 44
4 44
Beta 44
44 44
```

Check Answer

40. QID - 2.1152

Which of these expressions will return true?

Select 4 options

A. `"hello world".equals("hello world")`

B. `"HELLO world".equalsIgnoreCase("hello world")`

C. `"hello".concat(" world").trim().equals("hello world")`

D. `"hello world".compareTo("Hello world") < 0`

E. `"Hello world".toLowerCase( ).equals("hello world")`

Check Answer

41. QID - 2.1006

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}
class AnotherException extends Exception {}
public class ExceptionTest{
   public static void main(String [] args) throws Exception{
      try{
         m2();
      }
      finally{ m3(); }
    }
    public static void m2() throws NewException{  throw new NewExcep
    public static void m3() throws AnotherException{  throw new Anot
}
```

Select 1 option

A. It will compile but will throw `AnotherException` when run.

B. It will compile but will throw `NewException` when run.

C. It will compile and run without throwing any exceptions.

D. It will not compile.

E. None of the above.

Check Answer

42. QID - 2.1297

Which of the following are valid identifiers?

Select 2 options

A. class

B. $value$

C. angstrom

D. 2much

E. zer@

Check Answer

43. QID - 2.910

Consider the following code:

```
interface Flyer{ String getName(); }

class Bird implements Flyer{
    public String name;
    public Bird(String name){
        this.name = name;
    }
    public String getName(){ return name; }
}

class Eagle extends Bird {
    public Eagle(String name){
        super(name);
    }
}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Flyer f = new Eagle("American Bald Eagle");
        //PRINT NAME HERE
   }
}
```

Which of the following lines of code will print the name of the Eagle object?

Select 3 options

A. `System.out.println(f.name);`

B. `System.out.println(f.getName());`

C. `System.out.println(((Eagle)f).name);`

D. `System.out.println(((Bird)f).getName());`

E. `System.out.println(Eagle.name);`

F. `System.out.println(Eagle.getName(f));`

Check Answer

44. QID - 2.1216

Which of these statements are true?

Select 2 options

A. All classes must explicitly define a constructor.

B. A constructor can be declared private.

C. A constructor can declare a return value.

D. A constructor must initialize all the member variables of a class.

E. A constructor can access the non-static members of a class.

Check Answer

45. QID - 2.1350

What will the following code snippet print:

```
Float f = null;
try{
    f = Float.valueOf("12.3");
    String s = f.toString();
    int i = Integer.parseInt(s);
    System.out.println("i = "+i);
}
catch(Exception e){
    System.out.println("trouble : "+f);
}
```

Select 1 option

A. 12

B. 13

C. trouble : null

D. trouble : 12.3

E. trouble : 0.0

Check Answer

46. QID - 2.1345

Assume that a method named 'method1' contains code which may raise a non-runtime (checked) Exception.
What is the correct way to declare that method so that it indicates that it expects the caller to handle that exception?

Select 2 options

A. `public void method1() throws Throwable`

B. `public void method1() throw Exception`

C. `public void method1() throw new Exception`

D. `public void method1() throws Exception`

E. `public void method1()`

Check Answer

47. QID - 2.1248

Which of these are valid expressions to create a string of value "hello world" ?

Select 4 options

A. `" hello world".trim()`

B. `("hello" + " world")`

C. `(new String("hello") + " world")`

D. `("hello" + new String("world"))`

E. `"hello".concat(" world")`

Check Answer

48. QID - 2.1115

Consider the following program...

```
class ArrayTest{
   public static void main(String[] args){
      int ia[][] = { {1, 2}, null };
      for (int i = 0; i < 2; i++)
         for (int j = 0; j < 2; j++)
            System.out.println(ia[i][j]);
   }
}
```

Which of the following statements are true?

Select 1 option

A. It will not compile.

B. It will throw an `ArrayIndexOutOfBoundsException` at Runtime.

C. It will throw a `NullPointerException` at Runtime.

D. It will compile and run without throwing any exceptions.

E. None of the above.

Check Answer

49. QID - 2.1010

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     for : for(int i = 0; i< 10; i++){
        for (int j = 0; j< 10; j++){
             if ( i+ j > 10 )  break for;
        }
        System.out.println( "hello");
     }
  }
}
```

Select 1 option

A. It will print `hello` 6 times.

B. It will not compile.

C. It will print `hello` 2 times.

D. It will print `hello` 5 times.

E. It will print `hello` 4 times.

Check Answer

50. QID - 2.1346

Which of the following statements can be inserted at // 1 to make the code compile without errors?

```
public class InitTest{
   static int si = 10;
   int  i;
   final boolean bool;
   // 1
}
```

Select 1 option

A. `instance { bool = true; }`

B. `InitTest() { si += 10; }`

C. `{ si = 5; i = bool ? 1000 : 2000;}`

D. `{ i = 1000; }`

E. `{ bool = (si > 5); i = 1000; }`

Check Answer

51. QID - 2.1186

What will the following code print when compiled and run?

```
public class TestClass {
  public static void main(String[] args) {

    String s = "blooper";
    StringBuilder sb = new StringBuilder(s);
    s.append("whopper");
    sb.append("shopper");

    System.out.println(s);
    System.out.println(sb);
  }
}
```

Select 1 option

A. `blooper` and `bloopershopper`

B. `blooperwhopper` and `bloopershopper`

C. `blooper` and `blooperwhoppershopper`

D. It will not compile.

Check Answer

52. QID - 2.1364

Given the following class, which of these are valid ways of referring to the class from outside of the package com.enthu?

```
package com.enthu;
public class Base{
   // ....
   // lot of code...
}
```

Select 2 options

A. Base

B. By importing the package `com.*` and referring to the class as `enthu.Base`

C. importing `com.*` is illegal.

D. By importing `com.enthu.*` and referring to the class as `Base`.

E. By referring to the class as `com.enthu.Base`.

Check Answer

53. QID - 2.1171

Consider the following variable declaration within the definition of an interface:

```
int i = 10;
```

Which of the following declarations defined in a non-abstract class, is equivalent to the above?

Select 1 option

A. `public static int i = 10;`

B. `public final int i = 10;`

C. `public static final int i = 10;`

D. `public int i = 10;`

E. `final int i = 10;`

Check Answer

54. QID - 2.1301

What is wrong with the following code written in a single file named TestClass.java?

```
class SomeThrowable extends Throwable { }
class MyThrowable extends SomeThrowable { }
public class TestClass{
   public static void main(String args[]) throws SomeThrowable{
      try{
         m1();
      }catch(SomeThrowable e){
         throw e;
      }finally{
         System.out.println("Done");
      }
   }
   public static void m1() throws MyThrowable{
      throw new MyThrowable();
   }
}
```

Select 2 options

A. The main declares that it throws `SomeThrowable` but throws `MyThrowable`.

B. You cannot have more than 2 classes in one file.

C. The catch block in the main method must declare that it catches `MyThrowable` rather than `SomeThrowable`.

D. There is nothing wrong with the code.

E. `Done` will be printed.

Check Answer

55. QID - 2.1309

What can be inserted at //1 and //2 in the code below so that it can compile without errors:

```
class Doll{
    String name;
    Doll(String nm){
        this.name = nm;
    }
}

class Barbie extends Doll{
    Barbie(){
        //1
    }
    Barbie(String nm){
        //2
    }
}
  public class TestClass {
       public static void main(String[] args) {
          Barbie b = new Barbie("mydoll");
       }
   }
```

Select 2 options

A. `this("unknown");` at 1 and `super(nm);` at 2

B. `super("unknown");` at 1 and `super(nm);` at 2

C. `super();` at 1 and `super(nm);` at 2

D. `super();` at 1 and `Doll(nm);` at 2

E. `super("unknown");` at 1 and `this(nm);` at 2

F. `Doll();` at 1 and `Doll(nm);` at 2

Check Answer

56. QID - 2.835

Which of the following can be valid declarations of an integer variable?

Select 2 options

A. `global int x = 10;`

B. `final int x = 10;`

C. `public Int x = 10;`

D. `Int x = 10;`

E. `static int x = 10;`

Check Answer

57. QID - 2.1082

What will be the output when the following class is compiled and run?

```
class ScopeTest{
   static int x = 5;
   public static void main(String[] args){
      int x  = ( x=3 ) * 4;  // 1
      System.out.println(x);
   }
}
```

Select 1 option

A. It will not compile because line //1 cannot be parsed correctly.

B. It will not compile because x is used before initialization.

C. It will not compile because there is an ambiguous reference to x.

D. It will print 12.

E. It will print 3 .

Check Answer

58. QID - 2.1069

Which method declarations will enable a class to be run as a standalone program?

Select 2 options

A. `static void main(String args[ ])`

B. `public void static main(String args[ ])`

C. `public static main(String[ ] argv)`

D. `final public static void main(String [ ] array)`

E. `public static void main(String args[ ])`

Check Answer

59. QID - 2.1124

Given the following declaration, select the correct way to get the number of elements in the array, assuming that the array has been initialized.

```
int[] intArr;
```

Select 1 option

A. `intArr[ ].length( )`

B. `intArr.length( )`

C. `intArr.length`

D. `intArr[ ].size( )`

E. `intArr.size( )`

Check Answer

60. QID - 2.935

Consider the following class...

```
class TestClass{
   int i;
   public TestClass(int i) { this.i = i;  }
   public String toString(){
       if(i == 0) return null;
       else return ""+i;
   }
   public static void main(String[ ] args){
      TestClass t1 = new TestClass(0);
      TestClass t2 = new TestClass(2);
      System.out.println(t2);
      System.out.println(""+t1);
   }
}
```

What will be the output of the following program?

Select 1 option

A. It will throw NullPointerException when run.

B. It will not compile.

C. It will print 2 and then will throw NullPointerException.

D. It will print 2 and null.

E. None of the above.

Check Answer

61. QID - 2.1220

Given the following classes and declarations, which of these statements about //1 and //2 are true?

```
class A{
   private int i = 10;
   public void  f(){}
   public void g(){}
}

class B extends A{
   public int i = 20;
   public void g(){}
}

public class C{
   A a = new A();//1
   A b = new B();//2
}
```

Select 1 option

A. `System.out.println(b.i);` will print 10.

B. The statement `b.f( );` will give compile time error..

C. `System.out.println(b.i);` will print 20

D. All the above are correct.

E. None of the above statements is correct.

Check Answer

62. QID - 2.973

Which operators will always evaluate all the operands?

Select 2 options

A. &&

B. |

C. ||

D. ? :

E. %

Check Answer

63. QID - 2.1264

What sequence of digits will the following program print?

```
import java.util.* ;
public class ListTest{
   public static void main(String args[]){
      List s1 = new ArrayList( );
      s1.add("a");
      s1.add("b");
      s1.add(1, "c");
      List s2 = new ArrayList(  s1.subList(1, 1) );
      s1.addAll(s2);
      System.out.println(s1);
   }
}
```

Select 1 option

A. The sequence a, b, c is printed.

B. The sequence a, b, c, b is printed.

C. The sequence a, c, b, c is printed.

D. The sequence a, c, b is printed.

E. None of the above.

Check Answer

64. QID - 2.1226

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){
      k = j++;
   }
}
```

Select 2 options

A. The class TestClass cannot be declared abstract.

B. The variable j cannot be declared transient.

C. The variable k cannot be declared synchronized.

D. The constructor TestClass( ) cannot be declared final.

E. The method f( ) cannot be declared static.

Check Answer

65. QID - 2.1205

Which of these statements about interfaces are true?

Select 3 options

A. Interfaces permit multiple implementation inheritance.

B. Unlike a class, an interface can extend from multiple interfaces.

C. Members of an interface are never static.

D. Members of an interface may be static.

E. Interfaces cannot be final.

Check Answer

66. QID - 2.1019

You are modeling a class hierarchy for living things. You have a class `LivingThing` which has an abstract method `reproduce()`.
Now, you want to have 2 subclasses of `LivingThing` - `Plant` and `Animal`. Both do reproduce but the mechanisms are different. What would you do?

Select 1 option

A. Overload the `reproduce` method in Plant and Animal classes

B. Overload the `reproduce` method in LivingThing class.

C. Override the `reproduce` method in Plant and Animal classes

D. Either overload or override `reproduce` in `Plant` and `Animal` classes, it depends on the preference of the designer.

Check Answer

67. QID - 2.1272

Which of these are not legal declarations within a class?

Select 1 option

A. `static volatile int sa ;`

B. `final Object[ ] objArr = { null } ;`

C. `abstract int t ;`

D. `native void format( ) ;`

E. `final transient static private double PI = 3.14159265358979323846 ;`

Check Answer

68. QID - 2.1265

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int i=0, j=0;
        X1: for(i = 0; i < 3; i++){
            X2: for(j = 3; j > 0; j--){
                if(i < j) continue X1;
                else break X2;
            }
        }
        System.out.println(i+" "+j);
    }
}
```

Select 1 option

A. 0 3

B. 0 2

C. 3 0

D. 3 3

E. 2 2

Check Answer

69. QID - 2.1360

Given the following code, which statements can be placed at the indicated position without causing compile and run time errors?

```
public class Test{
   int i1;
   static int i2;
   public void method1(){
      int i;
      // ... insert statements here
   }
}
```

Select 3 options

A. `i = this.i1;`

B. `i = this.i2;`

C. `this = new Test( );`

D. `this.i = 4;`

E. `this.i1 = i2;`

Check Answer

70. QID - 2.1092

What will the following class print when compiled and run?

```
class Holder{
   int value = 1;
   Holder link;
   public Holder(int val){ this.value = val; }
   public static void main(String[] args){
 final Holder a = new Holder(5);
 Holder b = new Holder(10);
 a.link = b;
 b.link = setIt(a, b);
 System.out.println(a.link.value+" "+b.link.value);
   }

   public static Holder setIt(final Holder x, final Holder y){
       x.link = y.link;
       return x;
   }

}
```

Select 1 option

A. It will not compile because 'a' is final.

B. It will not compile because method setIt() cannot change x.link.

C. It will print 5, 10.

D. It will print 10, 10.

E. It will throw an exception when run.

Check Answer

71. QID - 2.937

Given the following class, which statements can be inserted at line 1 without causing the code to fail compilation?

```
public class TestClass{
   int a;
   int b = 0;
   static int c;
   public void m(){
      int d;
      int e = 0;
      // Line 1
   }
}
```

Select 4 options

A. `a++;`

B. `b++;`

C. `c++;`

D. `d++;`

E. `e++;`

Check Answer

72. QID - 2.1202

What would be the result of trying to compile and run the following program?

```
public class Test{
   int[] ia = new int[1];
   Object oA[]  = new Object[1];
   boolean bool;
   public static void main(String args[]){
      Test test = new Test();
      System.out.println(test.ia[0] + "  " +
test.oA[0]+"  "+test.bool);
   }
}
```

Select 1 option

A. The program will fail to compile, because of uninitialized variable 'bool'.

B. The program will throw a java.lang.NullPointerException when run.

C. The program will print "0 null false".

D. The program will print "0 null true".

E. The program will print null and false but will print junk value for ia[0].

Check Answer

73. QID - 2.948

Which statements about the output of the following programs are true?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true;
      boolean bool2 = false;
      boolean bool  = false;
      bool = (bool2 &  method1("1"));  //1
      bool = (bool2 && method1("2"));  //2
      bool = (bool1 |  method1("3"));  //3
      bool = (bool1 || method1("4"));  //4
   }
   public static boolean method1(String str){
      System.out.println(str);
      return true;
   }
}
```

Select 2 options

A. 1 will be the part of the output.

B. 2 will be the part of the output.

C. 3 will be the part of the output.

D. 4 will be the part of the output.

E. None of the above

Check Answer

74. QID - 2.901

Given:

```
public class Triangle{
    public int base;
    public int height;
    public double area;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }

    void updateArea(){
        area = base*height/2;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

The above class needs to protect an invariant on the "area" field. Which three members must have the public access modifiers removed to ensure that the invariant is maintained?

Select 3 options

A. the base field

B. the height field

C. the area field

D. the Triangle constructor

E. the setBase method

F. the setHeight method

Check Answer

75. QID - 2.978

Given the following definition of class, which member variables are accessible from OUTSIDE the package com.enthu.qb?

```
package com.enthu.qb;
public class TestClass{
   int i;
   public int j;
   protected int k;
   private int l;
}
```

Select 2 options

A. Member variable i.

B. Member variable j.

C. Member variable k.

D. Member variable k, but only for subclasses.

E. Member variable l.

Check Answer

76. QID - 2.1056

Which one of these is a proper definition of a class TestClass that cannot be sub-classed?

Select 1 option

A. `final class TestClass { }`

B. `abstract class TestClass { }`

C. `native class TestClass { }`

D. `static class TestClass { }`

E. `private class TestClass { }`

Check Answer

77. QID - 2.1286

Consider the following array definitions:

```
int[] array1, array2[];
int[][] array3;
int[] array4[], array5[];
```

Which of the following are valid statements?

Select 3 options

A. `array2 = array3;`

B. `array2 = array4;`

C. `array1 = array2;`

D. `array4 = array1;`

E. `array5 = array3`

Check Answer

78. QID - 2.1067

Which of these group of statements are valid?

Select 2 options

A. `{ { } }`

B. `{ continue ; }`

C. `block : { break block ; }`

D. `block : { continue block ; }`

E. The break keyword can only be used if there exists an enclosing loop construct (i.e. while, do-while or for).

Check Answer

79. QID - 2.977

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 != b2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print true;

C. It will print false;

D. Runtime error.

E. It will print nothing.

Check Answer

80. QID - 2.1046

What will be the output of the following program:

```
public class TestClass{
   public static void main(String args[]){
      try{
         m1();
      }catch(IndexOutOfBoundsException e){
         System.out.println("1");
         throw new NullPointerException();
      }catch(NullPointerException e){
         System.out.println("2");
         return;
      }catch (Exception e) {
         System.out.println("3");
      }finally{
         System.out.println("4");
      }
      System.out.println("END");
   }
   // IndexOutOfBoundsException is a subclass of RuntimeException.
   static void m1(){
      System.out.println("m1 Starts");
      throw new IndexOutOfBoundsException( "Big Bang " );
   }
}
```

Select 3 options

A. The program will print `m1 Starts`.

B. The program will print `m1 Starts, 1` and `4`, in that order.

C. The program will print `m1 Starts, 1` and `2`, in that order.

D. The program will print `m1 Starts, 1, 2` and `4` in that order.

E. `END` will not be printed.

Check Answer

81. QID - 2.1303

Consider following classes:

```
//In File Other.java
package other;
public class Other { public static String hello = "Hello"; }

//In File Test.java
package testPackage;
import other.*;
class Test{
   public static void main(String[] args){
      String hello = "Hello", lo = "lo";
      System.out.print((testPackage.Other.hello == hello) + " ");
      System.out.print((other.Other.hello == hello) + " ");   //line
      System.out.print((hello == ("Hel"+"lo")) + " ");           //l
      System.out.print((hello == ("Hel"+lo)) + " ");              //
      System.out.println(hello == ("Hel"+lo).intern());          //l
   }
}
class Other { static String hello = "Hello"; }
```

What will be the output of running class Test?

Select 1 option

A. false false true false true

B. false true true false true

C. true true true true true

D. true true true false true

E. None of the above.

Check Answer

82. QID - 2.850

Identify the correct statements about `ArrayList`?

Select 3 options

A. ArrayList extends java.util.AbstractList.

B. It allows you to access its elements in random order.

C. You must specify the class of objects you want to store in `ArrayList` when you declare a variable of type ArrayList.

D. `ArrayList` does not implement `RandomAccess`.

E. You can sort its elements using `Collections.sort()` method.

Check Answer

83. QID - 2.1172

Considering the following program, which of the options are true?

```
public class FinallyTest{
   public static void main(String args[]){
      try{
          if (args.length == 0) return;
          else throw new Exception("Some Exception");
      }
      catch(Exception e){
          System.out.println("Exception in Main");
      }
      finally{
          System.out.println("The end");
      }
   }
}
```

Select 2 options

A. If run with no arguments, the program will only print 'The end'.

B. If run with one argument, the program will only print 'The end'.

C. If run with one argument, the program will print 'Exception in Main' and 'The end'.

D. If run with one argument, the program will only print 'Exception in Main'.

E. Only one of the above is correct.

Check Answer

84. QID - 2.1168

Consider the following class hierarchy:

```
A
|
B1,  B2
|
C1, C2
```

(B1 and B2 are subclasses of A and C1, C2 are subclasses of B1) Which of the following statements are correct? Assume that objectOfA, objectOfC1, etc. are objects of classes A and C1 respectively.

Select 1 option

A. `objectOfC2 instanceof B2` will return `true`.

B. `objectOfC1 instanceof B1` will return `true`.

C. `objectOfA instanceof B1` will return `true`.

D. C1 c1 = objectOfA; is a valid statement.

E. `B1 b1 = objectOfB2;` is a valid statement.

Check Answer

85. QID - 2.1002

Given the following definitions and reference declarations:

```
interface I1 { }
interface I2 { }
class C1 implements I1 { }
class C2 implements I2 { }
class C3 extends C1 implements I2 { }
C1 o1;
C2 o2;
C3 o3;
```

Which of these statements are legal?

Select 3 options

A. `class C4 extends C3 implements I1, I2 { }`

B. `o3 = o1;`

C. `o3 = o2;`

D. `I1 i1 = o3; I2 i2 = (I2) i1;`

E. `I1 b = o3;`

Check Answer

86. QID - 2.1337

Consider the following code:

```
class A{
   A() {  print();   }
   void print() { System.out.println("A"); }
}
class B extends A{
   int i =   Math.round(3.5f);
   public static void main(String[] args){
      A a = new B();
      a.print();
   }
   void print() { System.out.println(i); }
}
```

What will be the output when class B is run ?

Select 1 option

A. It will print A, 4.

B. It will print A, A

C. It will print 0, 4

D. It will print 4, 4

E. None of the above.

Check Answer

87. QID - 2.920

Which is the earliest line in the following code after which the object created on line // 1 can be garbage collected, assuming no compiler optimizations are done?

```
public class NewClass{
   private Object o;
   void doSomething(Object s){  o = s;   }

   public static void main(String args[]){
      Object obj = new Object(); // 1
      NewClass tc = new NewClass(); //2
      tc.doSomething(obj); //3
      obj = new Object();    //4
      obj = null;    //5
      tc.doSomething(obj); //6
   }
}
```

Select 1 option

A. Line 1

B. Line 2

C. Line 3

D. Line 4

E. Line 5

F. Line 6

Check Answer

88. QID - 2.1140

Which of the following are valid declarations?

Select 1 option

A. `abstract int absMethod(int param) throws Exception;`

B. `abstract native int absMethod(int param) throws Exception;`

C. `float native getVariance() throws Exception;`

D. `abstract private int absMethod(int param) throws Exception;`

E. `strictfp float f;`

Check Answer

89. QID - 2.1302

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      StringBuilder sb = new StringBuilder("12345678");
      sb.setLength(5);
      sb.setLength(10);
      System.out.println(sb.length());
   }
}
```

Select 1 option

A. It will print 5.

B. It will print 10.

C. It will print 8.

D. Compilation error.

E. None of the above.

Check Answer

90. QID - 2.863

Consider the following code appearing in the same file:

```
class Data {
    private int x = 0, y = 0;
    public Data(int x, int y){
        this.x = x; this.y = y;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d = new Data(1, 1);
        //add code here
    }
}
```

Which of the following options when applied individually will change the Data object currently referred to by the variable d to contain 2, 2 as values for its data fields?

Select 1 option

A. Add the following two statements :

```
d.x = 2;
d.y = 2;
```

B. Add the following statement:

```
d = new Data(2, 2);
```

C. Add the following two statements:

```
d.x += 1;
d.y += 1;
```

D. Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x.setInt(x);   this.y.setInt(y);
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

E. Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x = x;   this.y = y;
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

Check Answer

Test 3 (Answered)

01. QID - 2.1091 : Working with Inheritance

What will the following code print when TestClass is run?

```
class Employee {

  static int i = 10;  {
      i = 15;
      System.out.print(" Employee "+i);
  }
  static { System.out.print(" Employee static "+i); }
}

class Manager extends Employee {
  static {
    i = 45;
    System.out.print(" Manager static ");
  }{
    i = 30;
    System.out.print(" Manager "+i);
  }
}

class Owner extends Manager{
  static { System.out.println("Owner"); }
}

public class TestClass {
  public static void main(String[] args) {
      Manager m = new Manager();
  }
}
```

Correct Option is : C

~~A.~~ Employee static 10 Manager static Employee 10 Manager 30

B. ~~`Employee static 10 Manager static Owner Manager 30`~~

> Since Owner is not at all referred any where, its static block will not be executed.

C. `Employee static 10 Manager static Employee 15 Manager 30`

D. ~~`Employee static 10 Manager static 15 Manager 20`~~

E. ~~It will not compile.~~

Explanation:

Although there is more to it that the following sequence, for the purpose of exam, this is all you need to know:

1. Static blocks of the base class (only once, in the order they appear in the class).

2. Static blocks of the class.

3. Non-static blocks of the base class.

4. Constructor of the base class.

5. Non-static blocks of the class.

6. Constructor of the class.

7. Derived class's static or non-static blocks are not executed if that class is not being used.
(For example, in this question class Owner is not being used.)

Back to Question without Answer

02. QID - 2.1354 : Creating and Using Arrays

Which of the following correctly declare a variable which can hold an array of 10 integers?

Correct Options are : A C

A. `int[ ] iA`

~~B.~~ `int[10] iA`

> Size of the array is NEVER specified on the Left Hand Side.

C. `int iA[ ]`

~~D.~~ `Object[ ] iA`

> Here, iA is an array of Objects. It cannot hold an array of integers.

~~E.~~ `Object[10] iA`

> Size of the array is NEVER specified on the LHS.

Explanation:

Note that an array of integers IS an Object :

```
Object obj = new int[]{ 1, 2, 3 }; // is valid.
```

But it is not an Array of objects.

```
Object[] o = new int[10]; // is not valid.
```

Difference between the placement of square brackets:

```
int[] i, j; //here i and j are both array of integers.
```

```
int i[], j; //here only i is an array of integers. j is just an
integer.
```

Back to Question without Answer

03. QID - 2.921 : Java Basics - Garbage Collection

Consider the following code:

```
class MyClass { }
public class TestClass{
   MyClass getMyClassObject(){
      MyClass mc = new MyClass(); //1
         return mc; //2
   }
   public static void main(String[] args){
      TestClass tc = new TestClass(); //3
      MyClass x = tc.getMyClassObject(); //4
      System.out.println("got myclass object"); //5
      x = new MyClass(); //6
      System.out.println("done"); //7
   }
}
```

After what line the MyClass object created at line 1 will be eligible for garbage collection?

Correct Option is : C

~~**A.**~~ 2

~~**B.**~~ 5

C. 6

> At line 6, x starts pointing to a new MyClassObject and no reference to the original MyClass object is left.

~~**D.**~~ 7

~~**E.**~~ Never till the program ends.

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

Back to Question without Answer

04. QID - 2.1111 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 == false){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Correct Option is : B

~~**A.**~~ Compile time error.

B. It will print `true`

~~**C.**~~ It will print `false`

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

All that `if()` needs is a `boolean`, now `b1 == false` returns `true`, which is a `boolean` and since `b2 = true` is an expression and every expression has a return value (which is the Left Hand Side of the expression), it returns `true`, which is again a `boolean`.

FYI: the return value of expression `i = 10;` is `10` (an int).

Back to Question without Answer

05. QID - 2.1311 : Handling Exceptions

The following class will not throw a `NullPointerException` when compiled and run.

```
class Test{
   public static void main(String[] args) throws Exception{
      int[] a = null;
      int i = a [ m1() ];
   }
   public static int m1() throws Exception{
      throw new Exception("Some Exception");
   }
}
```

Correct Option is : A

A. True

B. False

Explanation:

A NullPointerException never occurs because the index expression must be completely evaluated before any part of the indexing operation occurs, and that includes the check as to whether the value of the left-hand operand is null.
If the array reference expression produces null instead of a reference to an array, then a NullPointerException is thrown at runtime, but only after all parts of the array reference expression have been evaluated and only if these evaluations completed normally.

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated.
Note that if evaluation of the expression to the left of the brackets completes abruptly,

no part of the expression within the brackets will appear to have been evaluated.

Here, m1() is called first, which throws Exception and so 'a' is never accessed and NullPointerException is never thrown.

Back to Question without Answer

06. QID - 2.1085 : Java Basics

Consider the following two classes defined in two java source files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1 <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       X x = new X();
       x.apply(LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Correct Options are : C D

~~**A.**~~ `import static X;`

~~**B.**~~ `import static com.foo.*;`

> Bad syntax. com.foo is a package and you cannot import a package statically. You can only import static members of a class statically.

C. `import static com.foo.X.*;`

This static import is required because of Y is accessing LOGICID directly without its class name (i.e. X.LOGICID).

D. `import com.foo.*;`

This is required because Y also accesses the class X : `X x = new X();` If Y had only one statement, `System.out.println(LOGICID); import static com.foo.X.*` would suffice.

E. `import com.foo.X.LOGICID;`

Syntax for importing static fields is: import static <package>.<classname>.*; or import static <package>.<classname>.<fieldname>;

Back to Question without Answer

07. QID - 2.1048 : Handling Exceptions

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      try{
         RuntimeException re = null;
         throw re;
      }
      catch(Exception e){
         System.out.println(e);
      }
   }
}
```

Correct Option is : D

~~**A.**~~ The code will fail to compile, since `RuntimeException` cannot be caught by catching an Exception.

> `RuntimeException` can be caught by `catch(Exception e)` statement because `RuntimeException` is a subclass of `Exception`.

~~**B.**~~ The program will fail to compile, since `re` is `null`.

~~**C.**~~ The program will compile without error and will print `java.lang.RuntimeException` when run.

D. The program will compile without error and will print `java.lang.NullPointerException` when run.

> A `NullPointerException` will be thrown if the expression given to the throw statement results in a null pointer.

~~E.~~ The program will compile without error and will run and print `null`.

Explanation:

The try block generates `NullPointerException` which will be caught by the catch block.

Back to Question without Answer

08. QID - 2.1236 : Handling Exceptions

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     int x = 1;
     int y = 0;
     if( x/y ) System.out.println("Good");
     else  System.out.println("Bad");
  }
}
```

Correct Option is : D

~~**A.** Good~~

~~**B.** Bad~~

~~**C.** Exception at runtime saying division by Zero.~~

D. It will not compile.

> You need a boolean in the 'if' condition. Here, compiler sees that there is no way x/y can produce a boolean so it generates an error at compile time.

~~**E.** None of the above.~~

Back to Question without Answer

09. QID - 2.954 : Handling Exceptions

What class of objects can be declared by the throws clause?

Correct Options are : A B E

A. Exception

B. Error

~~**C.**~~ Event

~~**D.**~~ Object

E. RuntimeException

Explanation:

You can declare anything that is a Throwable or a subclass of Throwable, in the throws clause.

Back to Question without Answer

10. QID - 2.1150 : Working with Inheritance

Which of the following are valid declarations inside an interface?

Correct Options are : A B

A. `void compute();`

> All interface methods have to be public. No access control keyword in the method declaration also means public in an interface. (Note that the absence of access control keyword in the method declaration in a class means package protected.)

B. `public void compute();`

~~C.~~ `public final void compute();`

> final is not allowed.

~~D.~~ `static void compute();`

> static is not allowed.

~~E.~~ `protected void compute();`

> All interface methods have to be public.

Back to Question without Answer

11. QID - 2.1242 : Java Basics

Given:

```
public class TestClass{
     public static void main(String[] args){
     int i = Integer.parseInt(args[1]);
     System.out.println(args[i]);
   }
}
```

What will happen when you compile and run the above program using the following command line:

```
java TestClass 1 2
```

Correct Option is : D

~~**A.**~~ It will print `1`

~~**B.**~~ It will print `2`

~~**C.**~~ It will print some junk value.

D. It will throw `ArrayIndexOutOfBoundsException`.

~~**E.**~~ It will throw NumberFormatException

Explanation:

1. Arrays are indexed from 0.
2. In java, the name of the class is not the first element of args.

So, when the command line is : `java TestClass 1 2`, `args[0]` is 1 and `args[1]` is

2.

When you try to access `args[2]`, It will throw an `ArrayIndexOutofBoundsException` because the array length is only 2 so `args[2]` is out of bounds.

Back to Question without Answer

12. QID - 2.1147 : Working with Inheritance

Given the following code, which statements are true?

```
class A{
   int i;
}
class B extends A{
   int j;
}
```

Correct Options are : A D E

A. Class B extends class A.

~~B.~~ Class B is the superclass of class A.

> A is the super class of B.

~~C.~~ Class A inherits from class B.

> B inherits from A

D. Class B is a subclass of class A.

> Class B is a subclass of class A. Given the declaration "class B extends A" we can conclude that class B extends class A, class A is the superclass of class B, class B is a subclass of class A, and class B inherits from class A, which means that objects of class B also have all the members that objects of class A have.

E. Objects of class B will always have a member variable named i .

> Note that 'i' is not public or protected. So it will be inherited only if both the

classes are in same package.

Explanation:

Here are a few good words from the Java Language Specification:
Members of a class that are declared private are not inherited by subclasses of that class. Only members of a class that are declared protected or public are inherited by subclasses declared in a package other than the one in which the class is declared. Constructors and static initializers are not members and therefore are not inherited.

Back to Question without Answer

13. QID - 2.1329 : Working with Inheritance

Which of the following method definitions will prevent overriding of that method?

Correct Options are : A B C E

A. `public final void m1()`

> final methods cannot be overridden. That is the purpose of final keyword.

B. `public static void m1()`

C. `public static final void m1()`

> Keep in mind that static methods are not overridden, they are shadowed.

D. ~~`public abstract void m1()`~~

E. `private void m1()`

> private methods are not inherited at all so there is no question of overriding a private method.

Back to Question without Answer

14. QID - 2.959 : Handling Exceptions

What will the following class print ?

```
class Test{
   public static void main(String[] args){
      int[][] a = { { 00, 01 }, { 10, 11 } };
      int i = 99;
      try {
         a[val()][i = 1]++;
      } catch (Exception e) {
         System.out.println( i+", "+a[1][1]);
      }
   }
   static int val() throws Exception {
     throw new Exception("unimplemented");
   }
}
```

Correct Option is : A

A. 99 , 11

~~**B.**~~ 1 , 11

~~**C.**~~ 1 and an unknown value.

~~**D.**~~ 99 and an unknown value.

~~**E.**~~ It will throw an exception at Run time.

Explanation:

If evaluation of a dimension expression completes abruptly, no part of any dimension expression to its right will appear to have been evaluated.
Thus, while evaluating a[val()][i=1]++, when val() throws an exception, i=1 will not be executed. Therefore, i remains 99 and a[1][1] will print 11.

Back to Question without Answer

15. QID - 2.1159 : Working with Java Data Types - Variables and Objects

What happens when you try to compile and run the following class...

```
public class TestClass{
      public static void main(String[] args) throws Exception{
            int a = Integer.MIN_VALUE;
            int b = -a;
            System.out.println( a+ "   "+b);
      }
}
```

Correct Option is : B

~~**A.** It throws an~~ `OverFlowException`.

B. It will print two same negative numbers.

~~**C.** It will print two different negative numbers.~~

~~**D.** It will print one negative and one positive number of same magnitude.~~

~~**E.** It will print one negative and one positive number of different magnitude.~~

Explanation:

It prints: `-2147483648   -2147483648`

This happens because negative integers are stored in 2's complement form (complement the bits and add 1). For example:

Integer 1 in binary is `00000000 00000000 00000000 00000001` (32 bits)

So -1 in binary would be (complement the bits for 1 and add 1) :

Step 1 (complement the bits of 1): `11111111 11111111 11111111 11111110`
Step 2 (add 1 to step 1): `11111111 11111111 11111111 11111111`.

Now, let's see what happens in this question:

`a = Integer.MIN_VALUE = 10000000 00000000 00000000 00000000`

To get `-a`, apply the above two steps:

Step 1 (complement the bits): `011111111 11111111 11111111 11111111`
Step 2 (add 1) : `10000000 00000000 00000000 00000000`

So you got the exact same value that you started with!

(Note that you can see the binary form of an integer using `Integer.toBinaryString(i)` method.)

Back to Question without Answer

16. QID - 2.1280 : Using Loop Constructs

In the following code what will be the output if 0 (integer value zero) is passed to loopTest()?

```
public class TestClass{
   public void loopTest(int x){
      loop: for (int i = 1; i < 5; i++){
         for (int j = 1; j < 5; j++){
            System.out.println(i);
            if (x == 0) {  continue loop;  }
            System.out.println(j);
         }
      }
   }
}
```

Correct Option is : B

~~**A.**~~ The program will not compile.

B. It will print 1 2 3 4

~~**C.**~~ It will print 1 1 2 3 4

~~**D.**~~ It will print 1 1 2 2 3 3 4 4

~~**E.**~~ Produces no output

Explanation:

When x is 0, the statement continue loop; is executed. Note that loop: is for the outer loop. So, only one iteration (that too not full) is performed for the inner loop.
So, the inner loop prints the value of i only once and then next iteration of outer loop starts. 'j' is never printed. So, it prints 1 2 3 4.

Back to Question without Answer

17. QID - 2.970 : Working with Inheritance

Consider the following classes...

```
class Car{
   public int gearRatio = 8;
   public String accelerate() {  return "Accelerate : Car";  }
}
class SportsCar extends Car{
   public int gearRatio = 9;
   public String accelerate() {  return  "Accelerate : SportsCar";
   public static void main(String[] args){
      Car c = new SportsCar();
      System.out.println( c.gearRatio+"  "+c.accelerate() );
   }
}
```

What will be printed when SportsCar is run?

Correct Option is : C

~~**A.**~~ 8 Accelerate : Car

~~**B.**~~ 9 Accelerate : Car

C. 8 Accelerate : SportsCar

~~**D.**~~ 9 Accelerate : SportsCar

~~**E.**~~ None of the above.

Explanation:

The concept is : variables are shadowed and methods are overridden.
Method to be executed depends on the class of the actual object the variable is referencing to. Here, c refers to object of class SportsCar so SportsCar's accelerate() is selected.

Back to Question without Answer

18. QID - 2.849 : Using Loop Constructs

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {
        String[] sa = {"a", "b", "c"};
        for(String s :  sa){
            if("b".equals(s)) continue;
            System.out.println(s);
            if("b".equals(s)) break;
            System.out.println(s+" again");
        }
    }
}
```

Correct Option is : A

A.
```
a
a again
c
c again
```

~~B.~~
```
a
a again
b
```

~~C.~~
```
a
a again
b
b again
```

~~D.~~
```
c
c again
```

Explanation:

To determine the output you have to run through the loop one iteration at a time in your mind:

Iteration 1: s is "a". It is not equal to "b" so, it will print "a", and then "a again".
Iteration 2: s is "b". It is equal to "b", so the first if will execute "continue", which mean the rest of the code in the loop will not be executed (thus b and b again will not be printed), and the next iteration will start. Note that the second if is not executed at all because of the continue in the first if.
Iteration 3: s is "c", both the if conditions are not satisfied. So "c" and "c again" will be printed.

Back to Question without Answer

19. QID - 2.1305 : Handling Exceptions

Which of these statements are true?

Correct Options are : C D

~~**A.**~~ If a RuntimeException is not caught, the method will terminate and normal execution of the thread will resume.

> Any remaining code of the method will not be executed. Further, any uncaught exception will cause the JVM to kill the thread.

~~**B.**~~ An overriding method must declare that it throws the same exception classes as the method it overrides.

> It can throw any subset of the exceptions thrown by overridden class.

C. The main method of a program can declare that it throws checked exceptions.

> Any method can do that !

D. A method declaring that it throws a certain exception class may throw instances of any subclass of that exception class.

> Note that it cannot throw the instances of any superclasses of the exception.

~~**E.**~~ finally blocks are executed if and only if an exception gets thrown while inside the corresponding try block.

> Finally is ALWAYS executed. (Only exception is System.exit())

Explanation:

Normal execution will not resume if an exception is uncaught by a method. The exception will propagate up the method invocation stack until some method handles it. If no one handles it then the exception will be handled by the JVM and the JVM will terminated that thread.

An overriding method only needs to declare that it can throw a subset of the exceptions the overridden method can throw. Having no throws clause in the overriding method is OK.

Back to Question without Answer

20. QID - 2.936 : Working with Inheritance

Consider the following code:

```
class Super{
  static{ System.out.print("super "); }
}
class One{
  static {  System.out.print("one "); }
}
class Two extends Super{
  static {  System.out.print("two "); }
}
class Test{
  public static void main(String[] args){
     One o = null;
     Two t = new Two();
  }
}
```

What will be the output when class Test is run ?

Correct Option is : C

~~**A.**~~ It will print `one, super` and `two`.

> "`one`" will not be printed as class One is not actively used.

~~**B.**~~ It will print `one, two` and `super`.

C. It will print `super` and `two`.

> `Super` will be instantiated before `Two`.

~~**D.**~~ It will print `two` and `super`

~~**E.**~~ None of the above.

Explanation:

As per JLS 12.4.1 - A class or interface type T will be initialized immediately before the first occurrence of any one of the following:

T is a class and an instance of T is created.
T is a class and a static method declared by T is invoked.
A static field declared by T is assigned.
A static field declared by T is used and the field is not a constant variable.
T is a top-level class, and an assert statement lexically nested within T is executed.

The statement `One o = null;` does not fall in either of the cases mentioned above. So class One is not initialized and its static block is not executed. Class Two is initialized only after its superclass Super has been initialized.

Back to Question without Answer

21. QID - 2.979 : Handling Exceptions

What will be the output when the following code is compiled and run?

```
//in file Test.java
class E1 extends Exception{ }
class E2 extends E1 { }
class Test{
   public static void main(String[] args){
      try{
         throw new E2();
      }
      catch(E1 e){
         System.out.println("E1");
      }
      catch(Exception e){
         System.out.println("E");
      }
      finally{
         System.out.println("Finally");
      }
   }
}
```

Correct Option is : B

~~**A.**~~ It will not compile.

B. It will print E1 and Finally.

~~**C.**~~ It will print E1, E and Finally.

~~**D.**~~ It will print E and Finally.

~~**E.**~~ It will print Finally.

Explanation:

Since E2 is a sub class of E1, catch(E1 e) will be able to catch exceptions of class E2. Therefore E1 is printed. Once the exception is caught the rest of the catch blocks at the same level are ignored. So E is not printed. finally is always executed (except in case of System.exit()), so Finally is also printed.

Back to Question without Answer

22. QID - 2.1157 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following code?

```
class SwitchTest{
   public static void main(String args[]){
      for ( int i = 0 ; i < 3 ; i++){
         boolean flag  = false;
         switch (i){
            flag  = true;
         }
         if ( flag )  System.out.println( i );
      }
   }
}
```

Correct Option is : C

~~**A.**~~ It will print 0, 1 and 2.

~~**B.**~~ It will not print anything.

C. Compilation error.

It will say 'case', 'default' or '}' expected at compile time.

~~**D.**~~ Runtime error.

~~**E.**~~ None of the above.

Explanation:

You cannot have unlabeled block of code inside a `switch` block. Any code block must succeed a case label (or default label). Since there is no `case` statement in this switch block, there is no way the line `flag = true;` can be reached! Therefore, it will not compile.

Back to Question without Answer

23. QID - 2.1213 : Encapsulation

When a class whose members should be accessible only to members of that class is coded such a way that its members are accessible to other classes as well, this is called ...

Correct Option is : D

~~**A.**~~ strong coupling

~~**B.**~~ weak coupling

~~**C.**~~ strong typing

D. weak encapsulation

~~**E.**~~ weak polymorphism

~~**F.**~~ high cohesion

~~**G.**~~ low cohesion

Explanation:

When a class is properly encapsulated, only the members that are part of its public API are publicly accessible to other classes. Rest is all private or protected.

Back to Question without Answer

24. QID - 2.1197 : Java Basics

Which of the following are keywords in Java?

Correct Options are : A D E F

A. default

~~**B.**~~ NULL

> null (all lowercase) is a keyword.

~~**C.**~~ String

> It is a Java class.

D. throws

E. long

F. strictfp

Back to Question without Answer

25. QID - 2.870 : Creating and Using Arrays

What will the following code print when compiled and run?

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<String> al = new ArrayList<String>();
        al.add("111");
        al.add("222");
        System.out.println(al.get(al.size()));
     }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw a `NullPointerException` at run time.

C. It will throw an `IndexOutOfBoundsException` at run time.

size() method of ArrayList returns the number of elements. Here, it returns 2. Since numbering in ArrayList starts with 0. al.get(2) will cause an IndexOutOfBoundsException to be thrown because only 0 and 1 are valid indexes for a list of size 2.

~~**D.**~~ `222`

~~**E.**~~ `null`

Back to Question without Answer

26. QID - 2.1237 : Using Operators and Decision Constructs

Given:

```
enum Season { SUMMER, WINTER, SPRING, FALL }
```

What will the following code print?

```
Season s = Season.SPRING;
    switch(s){
        case SUMMER : System.out.println("SUMMER");
        case default : System.out.println("SEASON");
        case WINTER :  System.out.println("WINTER");
    }
```

Correct Option is : C

~~**A.**~~ `SEASON`

~~**B.**~~ `SEASON`
`WINTER`

C. It will not compile.

`case default : System.out.println("SEASON");` is syntactically wrong.
It should just be:
`default : System.out.println("SEASON");`

~~**D.**~~ It will not print anything.

Explanation:

If the code is changed to default : `System.out.println("SEASON");` it will print `SEASON`

```
WINTER.
```

Remember that since there is no `break` after the case statements, the control will go through the rest of the case statements without even checking the value of the cases.

Back to Question without Answer

27. QID - 2.1369 : Using Operators and Decision Constructs

Given the following declarations, identify which statements will return `true`:

```
Integer i1 = 1;
Integer i2 = new Integer(1);
int i3 = 1;
Byte b1 = 1;
Long g1 = 1L;
```

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Correct Options are : B D

~~**A.**~~ `i1 == i2`

> This will return false because both are pointing to different objects.

B. `i1 == i3`

> This will return true because one operand is a primitive int, so the other will be unboxed and then the value will be compared.

~~**C.**~~ `i1 == b1`

> This will not compile because type of i1 and b1 references are classes that are not in the same class hierarchy. So the compiler figures out at compile time itself these two references cannot ever point to the same object.

D. `i1.equals(i2)`

This will return true because both are Integer objects and both have the value 1.

E. `i1.equals(g1)`

This will return false because they are pointing to objects of different types.

Signature of equals method is : boolean equals(Object o);
Thus, it can take any object as a parameter and so there will be no compilation error.

Further, The equals method of all wrapper classes first checks if the two object are of same class or not. If not, they immediately return false.

F. `i1.equals(b1)`

This will return false because they are pointing to objects of different types.

Back to Question without Answer

28. QID - 2.969 : Creating and Using Arrays

Which of these array declarations and initializations are NOT legal?

Correct Options are : B E

~~**A.**~~ `int[ ] i[ ] = { { 1, 2 }, { 1 }, { }, { 1, 2, 3 } } ;`

B. `int i[ ] = new int[2] {1, 2} ;`

> If you give the elements explicitly you can't give the size. So it should be just int[] { 1, 2 } or just { 1, 2 }

~~**C.**~~ `int i[ ][ ] = new int[ ][ ] { {1, 2, 3}, {4, 5, 6} } ;`

~~**D.**~~ `int i[ ][ ] = { { 1, 2 }, new int[ 2 ] } ;`

E. `int i[4] = { 1, 2, 3, 4 } ;`

> You cannot specify the size on left hand side .

Explanation:

If you explicitly specify the members then you can't give the size. So option 2 is wrong.
The size of the array is never given during the declaration of an array reference. So option 5 is wrong.
The size of an array is always associated with the array instance, not the array reference.

Back to Question without Answer

29. QID - 2.1007 : Java Basics

How can you declare a method someMethod() such that an instance of the class is not needed to access it and all the members of the same package have access to it.

Correct Options are : A B C

A. `public static void someMethod()`

B. `static void someMethod()`

C. `protected static void someMethod()`

D. `void someMethod()`

E. `protected void someMethod()`

F. `public abstract static void someMethod()`

static methods can't be abstract.

Explanation:

Since the question says, "...an instance of the class is not needed...", the method has to be static.
Also, as the question does not say that other packages should not have access to the method so public or protected is also correct.

Back to Question without Answer

30. QID - 2.1307 : Creating and Using Arrays

Given:

```
double daaa[][][] = new double[3][][];
double d = 100.0;
double[][] daa = new double[1][1];
```

Which of the following will not cause any problem at compile time or runtime?

Correct Options are : B E

~~**A.**~~ `daaa[0] = d;`

> `daaa[0]` should be a 2 dimensional array because `daaa` is a 3 dimensional array.

B. `daaa[0] = daa;`

~~**C.**~~ `daaa[0] = daa[0];`

> `daaa[0]` should be a 2 dimensional array while `daa[0]` is a one dimensional array.

~~**D.**~~ `daa[1][1] = d;`

> `daa[1][1]` will cause an `ArrayIndexOutofBoundsException` because `daa`'s length is only 1 and the indexing starts from 0. To access the first element, you should use `daa[0][0]`.

E. `daa = daaa[0]`

Back to Question without Answer

31. QID - 2.1146 : Working with Java Data Types - Variables and Objects

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    unsigned byte b = 0;
    b--;
    System.out.println(b);
  }
}
```

Correct Option is : E

~~**A.**~~ 0

~~**B.**~~ -1

~~**C.**~~ 255

~~**D.**~~ -128

E. It will not compile.

Explanation:

There no unsigned keyword in java! A `char` can be used as an unsigned integer.

Back to Question without Answer

32. QID - 2.1281 : Using Loop Constructs

Which of the following statements regarding 'break' and 'continue' are true?

Correct Option is : A

A. break without a label, can occur only in a switch, while, do, or for statement.

~~**B.**~~ continue without a label, can occur only in a switch, while, do, or for statement.

It cannot occur in a switch.

~~**C.**~~ break can never occur without a label.

~~**D.**~~ continue can never occur WITH a label.

~~**E.**~~ None of the above.

Explanation:

A break statement with no label attempts to transfer control to the innermost enclosing switch, while, do, or for statement; this statement, which is called the break target, then immediately completes normally. If no switch, while, do, or for statement encloses the break statement, a compile-time error occurs.

A break statement with label Identifier attempts to transfer control to the enclosing labeled statement that has the same Identifier as its label; this statement, which is called the break target, then immediately completes normally. In this case, the break target need not be a while, do, for, or switch statement.

A continue statement with no label attempts to transfer control to the innermost enclosing while, do, or for statement; this statement, which is called the continue target, then immediately ends the current iteration and begins a new one. If no while, do, or for statement encloses the continue statement, a compile-time error occurs.

A continue statement with label Identifier attempts to transfer control to the enclosing labelled statement that has the same Identifier as its label; that statement, which is called the continue target, then immediately ends the current iteration and begins a new one. The continue target must be a while, do, or for statement or a compile-time error occurs. If no labelled statement with Identifier as its label contains the continue statement, a compile-time error occurs.

Back to Question without Answer

33. QID - 2.895 : Handling Exceptions

What two changes can you do, independent of each other, to make the following code compile:

```
//assume appropriate imports
class PortConnector {

    public PortConnector(int port) {
        if (Math.random() > 0.5) {
            throw new IOException();
        }

        throw new RuntimeException();
    }
}


public class TestClass {

    public static void main(String[] args) {
        try {
            PortConnector pc = new PortConnector(10);
        } catch (RuntimeException re) {
            re.printStackTrace();
        }
    }
}
```

Correct Options are : C E

~~**A.**~~ add `throws IOException` to the `main` method.

~~**B.**~~ add `throws IOException` to `PortConnector` constructor.

C. add `throws IOException` to the `main` method as well as to `PortConnector` constructor.

~~**D.**~~ Change `RuntimeException` to `java.io.IOException`.

E. add `throws Exception` to `PortConnector` constructor and change `catch(RuntimeException re)` to `catch(Exception re)` in the main method.

Explanation:

`IOException` is a checked exception and since the `PortConnector` constructor throws `IOException`, this exception (or its superclass) must be present in the throws clause of the constructor.

Now, the `main` method has two options, either catch `IOException` (or whatever exception PortConnector throws) in its catch block (i.e. option 5) or put that exception in its throws clause (i.e. option 3).

Back to Question without Answer

34. QID - 2.867 : Java Basics

Which of the following keywords may occur multiple times in a Java source file?

Correct Options are : A B C E

A. import

B. class

C. private

~~**D.**~~ package

There can be at most one package statement in a Java source file and it must be the first statement in the file.

E. public

Back to Question without Answer

35. QID - 2.1154 : Working with Inheritance

Consider the following code:

```
class A{
 public XXX m1(int a){
   return a*10/4-30;
 }
}
class A2 extends A{
 public YYY m1(int a){
   return a*10/4.0;
 }
}
```

What can be substituted for XXX and YYY so that it can compile without any problems?

Correct Option is : C

~~**A.**~~ int, int

> `a*10/4.0;` generates a double so, A2's m1() cannot return an int. (It will need a cast otherwise: `return (int) (a*10/4.0);`)

~~**B.**~~ int, double

> The return type should be same for overridden and overriding method.

C. double, double

> `a*10/4-30;` generates an int which can be returned as a double without any cast.

~~**D.**~~ double, int

> The return type should be same for overridden and overriding method.

~~**E.**~~ Nothing, they are simply not compatible.

Explanation:

Note that when a method returns objects (as opposed to primitives, like in this question), the principle of covariant returns applies. Meaning, the overriding method is allowed to return a subclass of the return type defined in the overridden method. Thus, if a base class's method is: `public A m();` then a subclass is free to override it with: `public A1 m();` if A1 extends A.

Back to Question without Answer

36. QID - 2.1255 : Handling Exceptions

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      loop :          // 1
      {
         System.out.println("Loop Lable line");
         try{
            for (  ;  true ;  i++ ){
               if( i >5) break loop;        // 2
            }
         }
         catch(Exception e){
            System.out.println("Exception in loop.");
         }
         finally{
            System.out.println("In Finally");       // 3
         }
      }
   }
}
```

Correct Option is : C

~~**A.**~~ Compilation error at line 1 as this is an invalid syntax for defining a label.

> You can apply a label to any code block or a block level statement (such as a `for` statement) but not to declarations. For example: `loopX : int i = 10;`

~~**B.**~~ Compilation error at line 2 as 'loop' is not visible here.

C. No compilation error and line 3 will be executed.

> Even if the break takes the control out of the block, the finally clause will be executed.

~~**D.**~~ No compilation error and line 3 will NOT be executed.

~~**E.**~~ Only the line with the label Loop will be printed.

Explanation:

A `break` without a label breaks the current loop (i.e. no iterations any more) and a break with a label tries to pass the control to the given label.
'Tries to' means that if the break is in a try block and the try block has a finally clause associated with it then it will be executed.

Back to Question without Answer

37. QID - 2.975 : Working with Java Data Types - Variables and Objects

In which of these variable declarations, will the variable remain uninitialized unless explicitly initialized?

Correct Option is : C

~~**A.**~~ Declaration of an instance variable of type int.

~~**B.**~~ Declaration of a static class variable of type float.

C. Declaration of a local variable of type float.

~~**D.**~~ Declaration of a static class variable of class Object

~~**E.**~~ Declaration of an instance variable of class Object.

Explanation:

We have to explicitly initialize local variables other wise they remain uninitialized and it will be a compile time error if such variables are accessed without getting initialized.
Instance variables and static variables receive a default value if not explicitly initialized. All primitive types get a defaults value equivalent to 0. (eg. int to 0 and float to 0.0f etc) and boolean to false.
The type/class of a variable does not affect whether a variable is initialized or not.

Back to Question without Answer

38. QID - 2.1245 : Using Loop Constructs

Consider the following class :

```
class Test{
   public static void main(String[] args){
      for (int i = 0; i < 10; i++) System.out.print(i + " ");  //1
      for (int i = 10; i > 0; i--) System.out.print(i + " ");  //2
      int i = 20;                  //3
      System.out.print(i + " ");   //4
   }
}
```

Which of the following statements are true?

Correct Options are : A B C D

A. As such the class will compile and print 20 at the end of its output.

B. It will not compile if line 3 is removed.

> If //3 is removed, 'i' will be undefined for //4

C. It will not compile if line 3 is removed and placed before line 1.

D. It will not compile if line 4 is removed and placed before line 3.

~~**E.**~~ Only Option 2, 3, and 4 are correct.

Explanation:

The scope of a local variable declared in 'for' statement is the rest of the 'for' statement, including its own initializer. So, when line 3 is placed before line 1, there is a redeclaration of i in the first for() which is not legal.
As such, the scope of i's declared in for() is just within the 'for' blocks. So placing line 4 before line 3 will not work since 'i' is not in scope there.

Back to Question without Answer

39. QID - 2.1370 : Overloading methods

What will the following code print when run?

```
class Baap {
    public int h = 4;
    public int getH() {
        System.out.println("Baap " + h);
        return h;
    }
}

public class Beta extends Baap {
    public int h = 44;
    public int getH() {
        System.out.println("Beta " + h);
        return h;
    }
    public static void main(String[] args) {
        Baap b = new Beta();
        System.out.println(b.h + " " + b.getH());
        Beta bb = (Beta) b;
        System.out.println(bb.h + " " + bb.getH());
    }
}
```

Correct Option is : D

~~**A.**~~
```
Beta 44
4 44
Baap 44
44 44
```

~~**B.**~~
```
Baap 44
4 44
Beta 44
44 44
```

~~**C.**~~
```
Beta 44
4 44
Beta 44
4 44
```

D.
```
Beta 44
4 44
Beta 44
44 44
```

Explanation:

Always remember: Methods are overridden and variables are shadowed.
Here, b refers to an object of class Beta so `b.getH()` will always call the overridden (subclass's method). However, the type of reference of b is Baap. so b.h will always refer to Baap's h.

Further, inside Beta's `getH()`, Beta's h will be accessed instead of Baap's h because you are accessing this.h ('this' is implicit) and the type of this is Beta.

Back to Question without Answer

40. QID - 2.1152 : Working with Java Data Types - String, StringBuilder

Which of these expressions will return true?

Correct Options are : A B C E

A. `"hello world".equals("hello world")`

B. `"HELLO world".equalsIgnoreCase("hello world")`

> equalsIgnoreCase() method treats both cases (upper and lower) as same.

C. `"hello".concat(" world").trim().equals("hello world")`

> "hello".concat(" world") will return "hello world" and trim() won't do any change because there is no space at the beginning or end of the string.

D. `"hello world".compareTo("Hello world") < 0`

> Notice that the Strings differ at the first position. The value returned by compareTo is (Unicode value of the left hand side - Unicode value of the right hand side).
>
> Although not required for the exam, it is good to know that for English alphabets, the unicode value of any lower case letter is always 32 more than the unicode value of the same letter in upper case. So, 'a' - 'A' or 'h' - 'H' is 32.

E. `"Hello world".toLowerCase( ).equals("hello world")`

> toLowerCase() converts all uppercase letters to lower case.

Explanation:

compareTo() does a lexicographical (like a dictionary) comparison. It stops at the first place where the strings have different letters.
If left hand side is bigger, it returns a positive number otherwise it returns a negative number. The value is equal to the difference of their unicode values.
If there is no difference then it returns zero. In this case, it will return ('h' - 'H') which is 32.

Back to Question without Answer

41. QID - 2.1006 : Handling Exceptions

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}
class AnotherException extends Exception {}
public class ExceptionTest{
   public static void main(String [] args) throws Exception{
      try{
         m2();
      }
      finally{ m3(); }
    }
    public static void m2() throws NewException{  throw new NewExcep
    public static void m3() throws AnotherException{  throw new Anot
}
```

Correct Option is : A

A. It will compile but will throw `AnotherException` when run.

~~**B.**~~ It will compile but will throw `NewException` when run.

~~**C.**~~ It will compile and run without throwing any exceptions.

~~**D.**~~ It will not compile.

~~**E.**~~ None of the above.

Explanation:

`m2()` throws `NewException`, which is not caught anywhere. But before exiting out of the main method, finally must be executed. Since finally throw `AnotherException` (due to a call to `m3()`), the `NewException` thrown in the try block (due to call to `m2()`) is ignored and `AnotherException` is thrown from the main method.

Back to Question without Answer

42. QID - 2.1297 : Java Basics

Which of the following are valid identifiers?

Correct Options are : B C

~~**A.**~~ class

> It is a keyword.

B. $value$

> $ and _ are allowed at any place. Numerals (0 - 9) are also allowed but not at the first place.

C. angstrom

~~**D.**~~ 2much

> Cannot start with a number. But much2 is valid.

~~**E.**~~ zer@

> No special chars except $ and _ are allowed.

Explanation:

As per JLS section 3.8: A valid identifier must start with a Java letter followed by a Java letter or a digit.

A "Java letter" is a character for which the method Character.isJavaIdentifierStart(int) returns true.

A "Java letter-or-digit" is a character for which the method Character.isJavaIdentifierPart(int) returns true.

The "Java letters" include uppercase and lowercase ASCII Latin letters A-Z (\u0041-\u005a), and a-z (\u0061-\u007a), and, for historical reasons, the ASCII underscore (_, or \u005f) and dollar sign ($, or \u0024). The $ character should be used only in mechanically generated source code or, rarely, to access pre-existing names on legacy systems.

The "Java digits" include the ASCII digits 0-9 (\u0030-\u0039).

Letters and digits may be drawn from the entire Unicode character set, which supports most writing scripts in use in the world today, including the large sets for Chinese, Japanese, and Korean. This allows programmers to use identifiers in their programs that are written in their native languages.

An identifier cannot have the same spelling (Unicode character sequence) as a keyword (§3.9), boolean literal (§3.10.3), or the null literal (§3.10.7), or a compile-time error occurs.

Back to Question without Answer

43. QID - 2.910 : Working with Inheritance

Consider the following code:

```
interface Flyer{ String getName(); }

class Bird implements Flyer{
    public String name;
    public Bird(String name){
        this.name = name;
    }
    public String getName(){ return name; }
}

class Eagle extends Bird {
    public Eagle(String name){
        super(name);
    }
}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Flyer f = new Eagle("American Bald Eagle");
        //PRINT NAME HERE
   }
}
```

Which of the following lines of code will print the name of the Eagle object?

Correct Options are : B C D

~~**A.**~~ `System.out.println(f.name);`

B. `System.out.println(f.getName());`

C. `System.out.println(((Eagle)f).name);`

D. `System.out.println(((Bird)f).getName());`

~~**E.**~~ `System.out.println(Eagle.name);`

> name is not a static field in class Eagle.

~~**F.**~~ `System.out.println(Eagle.getName(f));`

> This option doesn't make any sense.

Explanation:

While accessing a method or variable, the compiler will only allow you to access a method or variable that is visible through the class of the reference.

When you try to use `f.name`, the class of the reference f is `Flyer` and `Flyer` has no field named "`name`", thus, it will not compile. But when you cast f to `Bird` (or `Eagle`), the compiler sees that the class `Bird` (or `Eagle`, because `Eagle` inherits from `Bird`) does have a field named "`name`" so `((Eagle)f).name or ((Bird)f).name` will work fine.

`f.getName()` will work because `Flyer` does have a `getName()` method.

Back to Question without Answer

44. QID - 2.1216 : Constructors

Which of these statements are true?

Correct Options are : B E

~~**A.**~~ All classes must explicitly define a constructor.

> A default no args one will be provided if not defined any.

B. A constructor can be declared private.

> This feature is used for implementing Singleton Classes.

~~**C.**~~ A constructor can declare a return value.

~~**D.**~~ A constructor must initialize all the member variables of a class.

> All non-final instance variables get default values if not explicitly initialized.

E. A constructor can access the non-static members of a class.

> A constructor is non-static, and so it can access directly both the static and non-static members of the class.

Explanation:

Constructors need not initialize *all* the member variables of the class. A non-final member(i.e. an instance) variable will be assigned a default value if not explicitly initialized.

Back to Question without Answer

45. QID - 2.1350 : Handling Exceptions

What will the following code snippet print:

```
Float f = null;
try{
    f = Float.valueOf("12.3");
    String s = f.toString();
    int i = Integer.parseInt(s);
    System.out.println("i = "+i);
}
catch(Exception e){
    System.out.println("trouble : "+f);
}
```

Correct Option is : D

~~**A.** 12~~

~~**B.** 13~~

~~**C.** trouble : null~~

D. trouble : 12.3

~~**E.** trouble : 0.0~~

Explanation:

`f = Float.valueOf("12.3");` executes without any problem.

`int i = Integer.parseInt(s);` throws a NumberFormatException because 12.3 is not an integer.

Thus, the catch block prints trouble : 12.3

Back to Question without Answer

46. QID - 2.1345 : Handling Exceptions

Assume that a method named 'method1' contains code which may raise a non-runtime (checked) Exception.
What is the correct way to declare that method so that it indicates that it expects the caller to handle that exception?

Correct Options are : A D

A. `public void method1() throws Throwable`

~~B.~~ `public void method1() throw Exception`

> Note that it should be 'throws' and not 'throw'

~~C.~~ `public void method1() throw new Exception`

> This is not the right syntax.

D. `public void method1() throws Exception`

~~E.~~ `public void method1()`

> Non runtime exception must be declared in the throws clause.

Back to Question without Answer

47. QID - 2.1248 : Working with Java Data Types - String, StringBuilder

Which of these are valid expressions to create a string of value "hello world" ?

Correct Options are : A B C E

A. `" hello world".trim()`

> trim() removes starting and ending spaces.

B. `("hello" + " world")`

> operator + is overloaded for Strings.

C. `(new String("hello") + " world")`

> This will create "hello world"

~~D.~~ `("hello" + new String("world"))`

> It will create helloworld. No space between hello and world.

E. `"hello".concat(" world")`

Explanation:

All the expressions are legal. String literals are String objects and can be used just like any other object.

Back to Question without Answer

48. QID - 2.1115 : Creating and Using Arrays

Consider the following program...

```
class ArrayTest{
   public static void main(String[] args){
      int ia[][] = { {1, 2}, null };
      for (int i = 0; i < 2; i++)
         for (int j = 0; j < 2; j++)
            System.out.println(ia[i][j]);
   }
}
```

Which of the following statements are true?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an `ArrayIndexOutOfBoundsException` at Runtime.

C. It will throw a `NullPointerException` at Runtime.

~~**D.**~~ It will compile and run without throwing any exceptions.

~~**E.**~~ None of the above.

Explanation:

It will throw a `NullPointerException` for `ia[1][0]` because `ia[1]` is `null`. Note that `null` is not same as having less number of elements in an array than

expected.

If you try to access `ia[2][0]`, it would have thrown `ArrayIndexOutOfBoundsException` because the length of `ia` is only 2 and so `ia[2]` tries to access an element out of that range. `ia[2]` is not `null`, it simply does not exist.

Back to Question without Answer

49. QID - 2.1010 : Using Loop Constructs

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     for : for(int i = 0; i< 10; i++){
        for (int j = 0; j< 10; j++){
             if ( i+ j > 10 )  break for;
        }
        System.out.println( "hello");
     }
  }
}
```

Correct Option is : B

~~**A.**~~ It will print `hello` 6 times.

B. It will not compile.

~~**C.**~~ It will print `hello` 2 times.

~~**D.**~~ It will print `hello` 5 times.

~~**E.**~~ It will print `hello` 4 times.

Explanation:

Note that for is a keyword and so cannot be used as a label. But you can use any other

identifier as a label.
For example, The following code is valid even though String is a class name and String is also used as an identifier!

```
String String = "";   //This is valid.
String : for(int i = 0; i< 10; i++) //This is valid too!
{
   for (int j = 0; j< 10; j++){
        if ( i+ j > 10 )  break String;
   }
  System.out.println( "hello");
 }
```

It will print hello 2 times.

Back to Question without Answer

50. QID - 2.1346 : Working with Java Data Types - Variables and Objects

Which of the following statements can be inserted at // 1 to make the code compile without errors?

```
public class InitTest{
   static int si = 10;
   int  i;
   final boolean bool;
   // 1
}
```

Correct Option is : E

~~**A.**~~ `instance { bool = true; }`

> you cannot put the word instance here. It is not a keyword.

~~**B.**~~ `InitTest() { si += 10; }`

> It is a valid constructor but does not initialize bool, which is a final variable and must be initialized either in an instance block or in a constructor.

~~**C.**~~ `{ si = 5; i = bool ? 1000 : 2000;}`

> cannot use bool before initializing it !

~~**D.**~~ `{ i = 1000; }`

> bool remains uninitialized.

E. `{ bool = (si > 5); i = 1000; }`

Explanation:

A final variable must be initialized when an instance is constructed, or else the code will not compile. This can be done either in an instance initializer or in EVERY constructor.
The keyword static is used to signify that a block is static initializer. If nothing is there before starting curly brace then it is an instance initializer.

Back to Question without Answer

51. QID - 2.1186 : Working with Java Data Types - String, StringBuilder

What will the following code print when compiled and run?

```
public class TestClass {
  public static void main(String[] args) {

    String s = "blooper";
    StringBuilder sb = new StringBuilder(s);
    s.append("whopper");
    sb.append("shopper");

    System.out.println(s);
    System.out.println(sb);
  }
}
```

Correct Option is : D

~~**A.**~~ `blooper` and `bloopershopper`

~~**B.**~~ `blooperwhopper` and `bloopershopper`

~~**C.**~~ `blooper` and `blooperwhoppershopper`

D. It will not compile.

`append()` method does not exist in `String` class. It exits only in `StringBuffer` and `StringBuilder`. The value of sb will be `bloopershopper` though.

Back to Question without Answer

52. QID - 2.1364 : Java Basics

Given the following class, which of these are valid ways of referring to the class from outside of the package com.enthu?

```
package com.enthu;
public class Base{
   // ....
   // lot of code...
}
```

Correct Options are : D E

~~**A.**~~ Base

> Only if you import the whole package containing the class or import the class first.

~~**B.**~~ By importing the package `com.*` and referring to the class as `enthu.Base`

> package 'com' does not contain Base.

~~**C.**~~ importing `com.*` is illegal.

> It is perfectly legal.

D. By importing `com.enthu.*` and referring to the class as `Base`.

E. By referring to the class as `com.enthu.Base`.

Explanation:

A class or interface can be referred to by using its fully qualified name or its simple name.
Using the fully qualified name will always work, but to use the simple name either the class must be in the same package or it has to be imported.
By importing `com.enthu.*` all the classes from the package will be imported and can be referred to using simple names.
Importing `com.*` will not import the subpackage `enthu`. It will only import the classes in package `com`.

Back to Question without Answer

53. QID - 2.1171 : Working with Inheritance

Consider the following variable declaration within the definition of an interface:

```
int i = 10;
```

Which of the following declarations defined in a non-abstract class, is equivalent to the above?

Correct Option is : C

~~**A.**~~ `public static int i = 10;`

~~**B.**~~ `public final int i = 10;`

C. `public static final int i = 10;`

~~**D.**~~ `public int i = 10;`

~~**E.**~~ `final int i = 10;`

Explanation:

Fields in an interface are implicitly `public`, `static` and `final`. Although you can put these words in the interface definition but it is not a good practice to do so.

Back to Question without Answer

54. QID - 2.1301 : Handling Exceptions

What is wrong with the following code written in a single file named TestClass.java?

```
class SomeThrowable extends Throwable { }
class MyThrowable extends SomeThrowable { }
public class TestClass{
   public static void main(String args[]) throws SomeThrowable{
      try{
         m1();
      }catch(SomeThrowable e){
         throw e;
      }finally{
         System.out.println("Done");
      }
   }
   public static void m1() throws MyThrowable{
      throw new MyThrowable();
   }
}
```

Correct Options are : D E

~~**A.**~~ The main declares that it throws `SomeThrowable` but throws `MyThrowable`.

> That's OK. You can put a Super class in the throws clause and then you can throw any subclass exception.

~~**B.**~~ You cannot have more than 2 classes in one file.

> You sure can. The only limitation is you can have only one top level public class in a file.

~~**C.**~~ The catch block in the main method must declare that it catches `MyThrowable` rather than `SomeThrowable`.

You can catch a subclass exception in the catch clause that catches a super class.

D. There is nothing wrong with the code.

E. `Done` will be printed.

`Done` will be followed by an exception. Finally is always executed (Only exception is `System.exit();`)

Back to Question without Answer

55. QID - 2.1309 : Working with Inheritance

What can be inserted at //1 and //2 in the code below so that it can compile without errors:

```
class Doll{
    String name;
    Doll(String nm){
        this.name = nm;
    }
}

class Barbie extends Doll{
    Barbie(){
        //1
    }
    Barbie(String nm){
        //2
    }
}
  public class TestClass {
       public static void main(String[] args) {
          Barbie b = new Barbie("mydoll");
       }
   }
```

Correct Options are : A B

A. `this("unknown");` at 1 and `super(nm);` at 2

B. `super("unknown");` at 1 and `super(nm);` at 2

C. `super();` at 1 and `super(nm);` at 2

`super();` at 1 will not compile because super class Doll does not have a no args constructor.

~~**D.**~~ `super();` at 1 and `Doll(nm);` at 2

`super();` at 1 will not compile because super class Doll does not have a no args constructor. `Doll(nm);` at 2 is an invalid syntax for calling the super class's construtor.

~~**E.**~~ `super("unknown");` at 1 and `this(nm);` at 2

`this(nm);` at 2 will not compile because it is a recursive call to the same constructor.

~~**F.**~~ `Doll();` at 1 and `Doll(nm);` at 2

Both are using invalid syntax for calling the super class's construtor.

Explanation:

Since the super class `Doll` explicitly defines a constructor, compiler will not provide the default no-args constructor. Therefore, each of `Barbie`'s constructor must directly or indirectly call `Doll`'s string argument constructor, otherwise it will not compile. Although not relevant for this question, it is interesting to know that `super(name);` at //1 or //2, would not be valid because `name` is defined in the superclass and so it cannot be used by a subclass until super class's constructor has executed. For the same reason, `this(name);` cannot be used either.

Back to Question without Answer

56. QID - 2.835 : Working with Java Data Types - Variables and Objects

Which of the following can be valid declarations of an integer variable?

Correct Options are : B E

~~**A.**~~ `global int x = 10;`

> global is an invalid modifier. There is nothing like global in java. The closest you can get is static.

B. `final int x = 10;`

~~**C.**~~ `public Int x = 10;`

> Int with a capital I is invalid.

~~**D.**~~ `Int x = 10;`

> Int with a capital I is invalid.

E. `static int x = 10;`

Back to Question without Answer

57. QID - 2.1082 : Using Operators and Decision Constructs

What will be the output when the following class is compiled and run?

```
class ScopeTest{
   static int x = 5;
   public static void main(String[] args){
      int x  = ( x=3 ) * 4;  // 1
      System.out.println(x);
   }
}
```

Correct Option is : D

~~**A.**~~ It will not compile because line //1 cannot be parsed correctly.

~~**B.**~~ It will not compile because x is used before initialization.

> It is not.

~~**C.**~~ It will not compile because there is an ambiguous reference to x.

> There is no conflict for resolution of x. The local 'x' simply shadows the member variable 'x'.

D. It will print 12.

~~**E.**~~ It will print 3 .

Explanation:

x is first initialized by x = 3, then the value of this expression (i.e. "x = 3"), which is 3,

is multiplied by 4 and is again assigned to x. So it prints 12.

Back to Question without Answer

58. QID - 2.1069 : Java Basics

Which method declarations will enable a class to be run as a standalone program?

Correct Options are : D E

A. ~~`static void main(String args[ ])`~~

> Surprisingly, it does work. Even if the class is defined in a package.

B. ~~`public void static main(String args[ ])`~~

> Remember, return type and method name are NEVER separated.

C. ~~`public static main(String[ ] argv)`~~

> There always has to be return type for a method. Only constructors don't have a return type.

D. `final public static void main(String [ ] array)`

> final only means that subclasses cannot shadow (in case of static methods) or override (in case of instance methods) it.

E. `public static void main(String args[ ])`

Explanation:

If you run the following program by changing the accessibility from `public` to `private` and `protected`, it may work on some versions of Java.

However, for the purpose of Java Certification exam, it should be assumed that for the

JVM to execute a class using the standard main method, the accessibility of the main method must be `public`.

```
package test;
public class TestClass{
    private static void main(String args[]){
         System.out.println("hello");
    }
}
```

Back to Question without Answer

59. QID - 2.1124 : Creating and Using Arrays

Given the following declaration, select the correct way to get the number of elements in the array, assuming that the array has been initialized.

```
int[] intArr;
```

Correct Option is : C

~~**A.**~~ `intArr[ ].length( )`

~~**B.**~~ `intArr.length( )`

C. `intArr.length`

> Each array object has a member variable named public final length of type 'int' that contains the size of the array.

~~**D.**~~ `intArr[ ].size( )`

~~**E.**~~ `intArr.size( )`

Explanation:

FYI, All types of arrays are objects. i.e. `intArr instanceof Object` is `true`.

Back to Question without Answer

60. QID - 2.935 : Working with Java Data Types - String, StringBuilder

Consider the following class...

```
class TestClass{
   int i;
   public TestClass(int i) { this.i = i;  }
   public String toString(){
       if(i == 0) return null;
       else return ""+i;
   }
   public static void main(String[ ] args){
      TestClass t1 = new TestClass(0);
      TestClass t2 = new TestClass(2);
      System.out.println(t2);
      System.out.println(""+t1);
   }
}
```

What will be the output of the following program?

Correct Option is : D

~~**A.**~~ It will throw NullPointerException when run.

~~**B.**~~ It will not compile.

~~**C.**~~ It will print 2 and then will throw NullPointerException.

D. It will print 2 and null.

~~**E.**~~ None of the above.

Explanation:

The method `print()/println()` of `OutputStream` takes an `Object` and prints out a String that is returned by calling `toString()` on that object. Note that as `toString()` is defined in Object class, all objects in java have this method. So it prints 2 first.

The second object's `toString()` returns `null`, so it prints "`null`". There is no `NullPointerException` because no method is called on `null`.

Now, the other feature of `print/println` methods is that if they get `null` as input parameter, they print "`null`". They do not try to call `toString()` on `null`.

So, if you have, `Object o = null; System.out.println(o);` will print `null` and will not throw a `NullPointerException`.

Back to Question without Answer

61. QID - 2.1220 : Working with Inheritance

Given the following classes and declarations, which of these statements about //1 and //2 are true?

```
class A{
   private int i = 10;
   public void  f(){}
   public void g(){}
}

class B extends A{
   public int i = 20;
   public void g(){}
}

public class C{
   A a = new A();//1
   A b = new B();//2
}
```

Correct Option is : E

~~**A.**~~ `System.out.println(b.i);` will print 10.

> Since variable b is declared as of class A, you cannot do b.i even if the actual object is of class B because i in A is private.

~~**B.**~~ The statement `b.f( );` will give compile time error..

> `class A` has `f()` so `b.f()` is legal.

~~**C.**~~ `System.out.println(b.i);` will print 20

Since variable b is declared as of class A, you cannot do b.i even if the actual object is of class B because i in A is private.

D. All the above are correct.

E. None of the above statements is correct.

Back to Question without Answer

62. QID - 2.973 : Using Operators and Decision Constructs

Which operators will always evaluate all the operands?

Correct Options are : B E

~~**A.** &&~~

B. |

~~**C.** ||~~

~~**D.** ? :~~

> If the condition before ? returns true, only the first operand will be evaluated, otherwise only the second operand is evaluated.

E. %

> All mathematical operators evaluate all the operands.

Explanation:

`||` and `&&` are also known as short circuit operators since they do not evaluate the rest of the expression if the value of the expression can be determined by just evaluating part of the expression for example `( true || bool = false)` will not assign `false` to `bool` because the value of the expression can be told just by seeing the first part i.e. `true`. But `( true | bool = false)` will assign `false` to `bool`.

Back to Question without Answer

63. QID - 2.1264 : Creating and Using Arrays

What sequence of digits will the following program print?

```
import java.util.* ;
public class ListTest{
   public static void main(String args[]){
      List s1 = new ArrayList( );
      s1.add("a");
      s1.add("b");
      s1.add(1, "c");
      List s2 = new ArrayList(  s1.subList(1, 1) );
      s1.addAll(s2);
      System.out.println(s1);
   }
}
```

Correct Option is : D

~~**A.**~~ The sequence a, b, c is printed.

~~**B.**~~ The sequence a, b, c, b is printed.

~~**C.**~~ The sequence a, c, b, c is printed.

D. The sequence a, c, b is printed.

add(1, "c") will insert 'c' between 'a' and 'b' . sublist(1 , 1) will return an empty list.

~~**E.**~~ None of the above.

Explanation:

First, "a" and "b" are appended to an empty list. Next, "c" is added between "1" and "2".
Then a new list s2 is created using the sublist view allowing access to elements from index 1 to index 1(exclusive) (i.e. no elements).
Now, s2 is added to s1.
So s1 remains :a, c, b

Back to Question without Answer

64. QID - 2.1226 : Working with Java Data Types - Variables and Objects

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){
      k = j++;
   }
}
```

Correct Options are : C E

~~**A.**~~ The class TestClass cannot be declared abstract.

> Any class can be declared abstract even if it does not have any abstract method.

~~**B.**~~ The variable j cannot be declared transient.

C. The variable k cannot be declared synchronized.

> Variables cannot be declared synchronized. Only methods can be declared synchronized.

~~**D.**~~ The constructor TestClass( ) cannot be declared final.

> It is not a constructor, it is a simple method. Notice void return type.

E. The method f( ) cannot be declared static.

Because it refers to instance variables j and k

Explanation:

The moment you put a return type, it ceases to be a constructor. So the compiler thinks that option 4 is a method.

FYI, constructors are not inherited, and so it doesn't make sense to mark them as final. (It is illegal to mark a constructor as final for the same reason). So there is no question of overriding them.

Static methods cannot refer to non-static/instance members (this includes fields and methods).

Back to Question without Answer

65. QID - 2.1205 : Working with Inheritance

Which of these statements about interfaces are true?

Correct Options are : B D E

~~**A.**~~ Interfaces permit multiple implementation inheritance.

> Interfaces do not contain any implementations and only permit multiple interface inheritance.

B. Unlike a class, an interface can extend from multiple interfaces.

> For example - interface I1 extends I2, I2, I3{ }

~~**C.**~~ Members of an interface are never static.

> Fields are static and methods are non static.

D. Members of an interface may be static.

> methods of an interface are public and non-static. Fields are public, static and final.

E. Interfaces cannot be final.

Explanation:

An interface can extend any number of other interfaces and can be extended by any number of other interfaces. Variables in interfaces are always static and method prototypes in interfaces can never be static.

Back to Question without Answer

66. QID - 2.1019 : Working with Inheritance

You are modeling a class hierarchy for living things. You have a class `LivingThing` which has an abstract method `reproduce()`.
Now, you want to have 2 subclasses of `LivingThing` - `Plant` and `Animal`. Both do reproduce but the mechanisms are different. What would you do?

Correct Option is : C

~~**A.**~~ Overload the `reproduce` method in Plant and Animal classes

~~**B.**~~ Overload the `reproduce` method in LivingThing class.

C. Override the `reproduce` method in Plant and Animal classes

~~**D.**~~ Either overload or override `reproduce` in `Plant` and `Animal` classes, it depends on the preference of the designer.

Explanation:

This kind of scenario where the subclass HAS the behavior of the base class but implements it in a different way is called as overriding. Here, both `Plant` and `Animal` reproduce, so they have the behavior of the base class but they do it differently, so you have to override the base class method in their code. Inheritance is always involved in overriding.
Overloading is quite different, when you want to do similar (not same) things but the inputs are different then you overload a method. For example, you may have two add methods:
`add(int i1, int i2)` and `add(ComplexNo c1, ComplexNo 2)`. Here both are doing similar things (that is why both are named as add) but inputs are different. Both

are two entirely different methods and there is no inheritance involved.

Back to Question without Answer

67. QID - 2.1272 : Java Basics

Which of these are not legal declarations within a class?

Correct Option is : C

~~**A.**~~ `static volatile int sa ;`

~~**B.**~~ `final Object[ ] objArr = { null } ;`

> Declares and defines an array of Objects of length 1.

C. `abstract int t ;`

> Variables can't be declared abstract and native.

~~**D.**~~ `native void format( ) ;`

~~**E.**~~ `final transient static private double PI = 3.14159265358979323846 ;`

Explanation:

`static` and `final` are valid modifiers for both member field and method declarations within a class.
`transient` and `volatile` modifiers are only valid for member field declarations.
`abstract` and `native` are only valid for member methods.

Note: a class declaration can have only have `final`, `abstract` and `public` as modifiers, unless it is a nested class, in which case, it can be declared `private` or `protected` as well.

Within a method, a local variable may be declared as `final`.

Back to Question without Answer

68. QID - 2.1265 : Using Loop Constructs

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int i=0, j=0;
        X1: for(i = 0; i < 3; i++){
            X2: for(j = 3; j > 0; j--){
                if(i < j) continue X1;
                else break X2;
            }
        }
        System.out.println(i+" "+j);
    }
}
```

Correct Option is : D

~~**A.**~~ 0 3

~~**B.**~~ 0 2

~~**C.**~~ 3 0

D. 3 3

~~**E.**~~ 2 2

Explanation:

The statement: `if(i < j) continue X1; else break X2;` only makes sure that the inner loop does not iterate more than once. i.e. for each iteration of i, j only takes the value of 3 and then the j loop terminates, either because of `continue X1;` or because of `break X2;`.
Now, the point to remember here is that when the loop `for(i = 0; i < 3; i++)` ends, the value of i is 3 and not 2.
Similarly, if there were no statement inside inner loop, the value of j after the end of the loop would have been 0 and not 1.

Back to Question without Answer

69. QID - 2.1360 : Java Basics

Given the following code, which statements can be placed at the indicated position without causing compile and run time errors?

```
public class Test{
   int i1;
   static int i2;
   public void method1(){
      int i;
      // ... insert statements here
   }
}
```

Correct Options are : A B E

A. `i = this.i1;`

> As i1 is an instance variable, it is accessible through 'this'.

B. `i = this.i2;`

> Although 'this' is not needed to access `i2`, it is not an error to do so.

~~C.~~ `this = new Test( );`

> Nope, you can't change `this`.

~~D.~~ `this.i = 4;`

> You cannot do `this.i` as `i` is a local variable.

E. `this.i1 = i2;`

You are just assigning a static field's value to non-static field.

Back to Question without Answer

70. QID - 2.1092 : Working with Methods

What will the following class print when compiled and run?

```
class Holder{
   int value = 1;
   Holder link;
   public Holder(int val){ this.value = val; }
   public static void main(String[] args){
 final Holder a = new Holder(5);
 Holder b = new Holder(10);
 a.link = b;
 b.link = setIt(a, b);
 System.out.println(a.link.value+" "+b.link.value);
   }

   public static Holder setIt(final Holder x, final Holder y){
       x.link = y.link;
       return x;
   }

}
```

Correct Option is : E

~~**A.**~~ It will not compile because 'a' is final.

> 'a' is final is true, but that only means that a will keep pointing to the same object for the entire life of the program. The object's internal fields, however, can change.

~~**B.**~~ It will not compile because method setIt() cannot change x.link.

> Since x and y are final, the method cannot change what x and y to point to some other object but it can change the objects' internal fields.

~~**C.**~~ It will print 5, 10.

~~**D.**~~ It will print 10, 10.

E. It will throw an exception when run.

When method setIt() executes, x.link = y.link, x.link becomes null because y.link is null so a.link.value throws NullPointerException.

Back to Question without Answer

71. QID - 2.937 : Java Basics

Given the following class, which statements can be inserted at line 1 without causing the code to fail compilation?

```
public class TestClass{
   int a;
   int b = 0;
   static int c;
   public void m(){
      int d;
      int e = 0;
      // Line 1
   }
}
```

Correct Options are : A B C E

A. `a++;`

> Here, 'a' is an instance variable of type int. Therefore, it will be given a default value of Zero and so it need not be initialized explicitly.

B. `b++;`

C. `c++;`

> Here 'c' is a class variable (also called as static variable) of type int so it will be given a default value of Zero and so it need not be initialized explicitly.

~~D.~~ `d++;`

This will not compile because 'd' is not initialized. Note that automatic variables (also called as method local variables i.e. variables declared within a method) have to be explicitly initialized.

E. `e++;`

Explanation:

All the instance or static variables are given a default values if they are not explicitly initialized. All numeric variable are given a value of zero or equivalent to zero (i.e. 0 for integral types and 0.0 for double/float). Booleans are initialized to false and objects are initialized to null.

Back to Question without Answer

72. QID - 2.1202 : Creating and Using Arrays

What would be the result of trying to compile and run the following program?

```
public class Test{
   int[] ia = new int[1];
   Object oA[]  = new Object[1];
   boolean bool;
   public static void main(String args[]){
      Test test = new Test();
      System.out.println(test.ia[0] + "  " +
test.oA[0]+"  "+test.bool);
   }
}
```

Correct Option is : C

~~**A.**~~ The program will fail to compile, because of uninitialized variable 'bool'.

> No, All the instance variables are initialized by default values.

~~**B.**~~ The program will throw a java.lang.NullPointerException when run.

> No reason for this at all.

C. The program will print "0 null false".

~~**D.**~~ The program will print "0 null true".

> All the variables, including the array elements, will be initialized to their default values.

~~**E.**~~ The program will print null and false but will print junk value for ia[0].

All the elements of the arrays of primitives are initialized to default values.

Explanation:

Following are the default values that instance variables are initialized with if not initialized explicitly:

types (byte, short, char, int, long, float, double) to 0 (or 0.0).

All Object types to null.

boolean to false.

Back to Question without Answer

73. QID - 2.948 : Using Operators and Decision Constructs

Which statements about the output of the following programs are true?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true;
      boolean bool2 = false;
      boolean bool  = false;
      bool = (bool2 &  method1("1"));  //1
      bool = (bool2 && method1("2"));  //2
      bool = (bool1 |  method1("3"));  //3
      bool = (bool1 || method1("4"));  //4
   }
   public static boolean method1(String str){
      System.out.println(str);
      return true;
   }
}
```

Correct Options are : A C

A. 1 will be the part of the output.

> & (unlike &&), when used as a logical operator, does not short circuit the expression, which means it always evaluates both the operands even if the result of the whole expression can be known by just evaluating the left operand.

B. 2 will be the part of the output.

C. 3 will be the part of the output.

> & and | (unlike && and ||), when used as logicals operators, do not short circuit

the expression, which means they always evaluate both the operands even if the result of the whole expression can be known by just evaluating the left operand.

~~**D.**~~ 4 will be the part of the output.

~~**E.**~~ None of the above

Explanation:

`&` and `|` do not short circuit the expression. The value of all the expressions (1 through 4) can be determined just by looking at the first part.
`&&` and `||` do not evaluate the rest of the expression if the result of the whole expression can be known by just evaluating the left operand, so `method1()` is not called for 2 and 4.

Back to Question without Answer

74. QID - 2.901 : Encapsulation

Given:

```
public class Triangle{
    public int base;
    public int height;
    public double area;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }

    void updateArea(){
        area = base*height/2;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

The above class needs to protect an invariant on the "area" field. Which three members must have the public access modifiers removed to ensure that the invariant is maintained?

Correct Options are : A B C

A. the base field

B. the height field

C. the area field

~~**D.**~~ the Triangle constructor

~~**E.** the setBase method~~

~~**F.** the setHeight method~~

Explanation:

An invariant means a certain condition that constrains the state stored in the object. For example, in this case the value of the area field of the Triangle must always be consistent with its base and height fields. Thus, it should never have a value that is different from `base*height/2`.

If you allow other classes to directly change the value of base, height, or area, using direct field access, the area field may not contain the correct area thereby breaking the invariant.

To prevent this inconsistency from happening, you need to prohibit changing the instance fields directly and instead permit the changes only through the setter method because these methods call the updateArea method and keep the area and base and height consistent.

Back to Question without Answer

75. QID - 2.978 : Working with Methods - Access Modifiers

Given the following definition of class, which member variables are accessible from OUTSIDE the package com.enthu.qb?

```
package com.enthu.qb;
public class TestClass{
   int i;
   public int j;
   protected int k;
   private int l;
}
```

Correct Options are : B D

~~**A.**~~ Member variable i.

> No modifier means package(or default access) and is only accessible inside the package.

B. Member variable j.

> public things (classes, methods and fields) are accessible from anywhere.

~~**C.**~~ Member variable k.

> Only if the accessing class is a subclass of TestClass.

D. Member variable k, but only for subclasses.

> protected things (methods and fields) can be accessed from within the package and from subclasses

~~**E.**~~ Member variable l.

private things are accessible only from the class that has it.

Explanation:

`public > protected > package (i.e. no modifier) > private`
where `public` is least restrictive and `private` is most restrictive.

Remember:
`protected` is less restrictive than package access. So a method(or field) declared as protected will be accessible from a subclass even if the subclass is not in the same package.
The same is not true for package access.
A top level class can only have either `public` or no access modifier but a method or field can have all the four. Note that `static`, `final`, `native` and `synchronized` are not considered as access modifiers.

Back to Question without Answer

76. QID - 2.1056 : Java Basics

Which one of these is a proper definition of a class TestClass that cannot be sub-classed?

Correct Option is : A

A. `final class TestClass { }`

~~B.~~ `abstract class TestClass { }`

~~C.~~ `native class TestClass { }`

~~D.~~ `static class TestClass { }`

~~E.~~ `private class TestClass { }`

Explanation:

A final class cannot be subclassed.
Although declaring a method static usually implies that it is also final, this is not true for classes. An inner class can be declared static and still be extended.
Note that for classes, final means it cannot be extended, while for methods, final means it cannot be overridden in a subclass.
The native keyword can only be used on methods, not on classes and or variables.

Back to Question without Answer

77. QID - 2.1286 : Creating and Using Arrays

Consider the following array definitions:

```
int[] array1, array2[];
int[][] array3;
int[] array4[], array5[];
```

Which of the following are valid statements?

Correct Options are : A B E

A. `array2 = array3;`

B. `array2 = array4;`

~~C.~~ `array1 = array2;`

~~D.~~ `array4 = array1;`

E. `array5 = array3`

Explanation:

There is a subtle difference between: `int[] i;` and `int i[];` although in both the cases, i is an array of integers.
The difference is if you declare multiple variables in the same statement such as:
`int[] i, j;` and `int i[], j;`, j is not of the same type in the two cases.
In the first case, j is an array of integers while in the second case, j is just an integer.
Therefore, in this question:
`array1` is an array of `int`
`array2, array3, array4,` and `array5` are arrays of `int` arrays

Therefore, option 1, 2 and 5 are valid.

Back to Question without Answer

78. QID - 2.1067 : Using Loop Constructs

Which of these group of statements are valid?

Correct Options are : A C

A. `{ { } }`

> See explanation.

~~B.~~ `{ continue ; }`

> continue can be used only inside a 'for', 'while' or 'do while' loop.

C. `block : { break block ; }`

> This is a valid example of breaking out of a labelled block.

~~D.~~ `block : { continue block ; }`

> continue can be used only inside a 'for', 'while' or 'do while' loop.

~~E.~~ The break keyword can only be used if there exists an enclosing loop construct (i.e. while, do-while or for).

> It can also be used to break out of a labeled block and in switch construct. For example, option 3.

Explanation:

The construct '{ }' is a compound statement. The compound statement can contain zero or more arbitrary statements.

Thus, { { } }, which is a compound statement containing one statement which is a compound statement containing no statement, is legal.

Back to Question without Answer

79. QID - 2.977 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 != b2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Correct Option is : C

~~**A.**~~ Compile time error.

~~**B.**~~ It will print true;

C. It will print false;

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

All an `if()` needs is a `boolean`. Now, `b1 != b2` returns `false` which is a boolean and so the expression becomes `b2 = false`. It returns `false` which is again a `boolean`. So there is no error and it prints `false`.
Remember that every expression has a return value (which is actually the Left Hand Side of the expression). For example, The value of the expressing `i = 10` , is `10` (an

int).

Back to Question without Answer

80. QID - 2.1046 : Handling Exceptions

What will be the output of the following program:

```
public class TestClass{
   public static void main(String args[]){
      try{
         m1();
      }catch(IndexOutOfBoundsException e){
         System.out.println("1");
         throw new NullPointerException();
      }catch(NullPointerException e){
         System.out.println("2");
         return;
      }catch (Exception e) {
         System.out.println("3");
      }finally{
         System.out.println("4");
      }
      System.out.println("END");
   }
   // IndexOutOfBoundsException is a subclass of RuntimeException.
   static void m1(){
      System.out.println("m1 Starts");
      throw new IndexOutOfBoundsException( "Big Bang " );
   }
}
```

Correct Options are : A B E

A. The program will print `m1 Starts`.

B. The program will print `m1 Starts, 1` and `4`, in that order.

~~**C.**~~ The program will print `m1 Starts, 1` and `2`, in that order.

~~**D.**~~ The program will print `m1 Starts, 1, 2` and `4` in that order.

E. `END` will not be printed.

Explanation:

The `IndexOutOfBoundsException` is handled by the first catch block. Inside this block, a new `NullPointerException` is thrown. As this exception is not thrown inside the try block, it will not be caught by any of the remaining catch blocks. It will actually be sent to the caller of the `main()` method after the `finally` block is executed. (Hence '4' in the output.)

The code that prints `END` is never reached, since the `NullPointerException` remains uncaught after the execution of the finally block.

At the end a stack trace for the `NullPointerException` will be printed.

Back to Question without Answer

81. QID - 2.1303 : Working with Java Data Types - String, StringBuilder

Consider following classes:

```
//In File Other.java
package other;
public class Other { public static String hello = "Hello"; }

//In File Test.java
package testPackage;
import other.*;
class Test{
   public static void main(String[] args){
      String hello = "Hello", lo = "lo";
      System.out.print((testPackage.Other.hello == hello) + " ");
      System.out.print((other.Other.hello == hello) + " ");   //line
      System.out.print((hello == ("Hel"+"lo")) + " ");           //l
      System.out.print((hello == ("Hel"+lo)) + " ");              //
      System.out.println(hello == ("Hel"+lo).intern());          //l
   }
}
class Other { static String hello = "Hello"; }
```

What will be the output of running class Test?

Correct Option is : D

~~**A.**~~ false false true false true

~~**B.**~~ false true true false true

~~**C.**~~ true true true true true

D. true true true false true

~~**E.**~~ None of the above.

Explanation:

These are the six facts on Strings:
1. Literal strings within the same class in the same package represent references to the same String object.
2. Literal strings within different classes in the same package represent references to the same String object.
3. Literal strings within different classes in different packages likewise represent references to the same String object.
4. Strings computed by constant expressions are computed at compile time and then treated as if they were literals.
5. Strings computed at run time are newly created and therefore are distinct. (So line 4 prints false.)
6. The result of explicitly interning a computed string is the same string as any pre-existing literal string with the same contents. (So line 5 prints true.)

We advise you to read section 3.10.5 String Literals in Java Language Specification.

Back to Question without Answer

82. QID - 2.850 : Creating and Using Arrays

Identify the correct statements about `ArrayList`?

Correct Options are : A B E

A. ArrayList extends java.util.AbstractList.

ArrayList is a subclass of AbstractList.

```
java.lang.Object
 -  java.util.AbstractCollection<E>
   -    java.util.AbstractList<E>
     -      java.util.ArrayList<E>
```

All Implemented Interfaces:

```
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, Rand
```

B. It allows you to access its elements in random order.

This is true because you can directly access any element using `get(index)` method. (This is unlike a `LinkedList`, in which you have to go through all the elements occuring before Nth element before you can access the Nth element.)

~~**C.**~~ You must specify the class of objects you want to store in `ArrayList` when you declare a variable of type ArrayList.

This is not true because you can still use non-generic form. For example, instead of using

```
ArrayList<String> listOfStrings;
```

you can use:

```
ArrayList listOfStrings;
```

Of course, if you use non generic version, you will lose the compile time type checking.

~~D.~~ `ArrayList` does not implement `RandomAccess`.

It does.
`RandomAccess` is a marker interface used by List implementations to indicate that they support fast (generally constant time) random access. The primary purpose of this interface is to allow generic algorithms to alter their behavior to provide good performance when applied to either random or sequential access lists.

E. You can sort its elements using `Collections.sort()` method.

An `ArrayList` is a `List` so you can use it where ever a List is required. This include Collections methods such as `sort, reverse, and shuffle`.

Back to Question without Answer

83. QID - 2.1172 : Handling Exceptions

Considering the following program, which of the options are true?

```
public class FinallyTest{
   public static void main(String args[]){
      try{
          if (args.length == 0) return;
          else throw new Exception("Some Exception");
      }
      catch(Exception e){
          System.out.println("Exception in Main");
      }
      finally{
          System.out.println("The end");
      }
   }
}
```

Correct Options are : A C

A. If run with no arguments, the program will only print 'The end'.

~~**B.**~~ If run with one argument, the program will only print 'The end'.

C. If run with one argument, the program will print 'Exception in Main' and 'The end'.

~~**D.**~~ If run with one argument, the program will only print 'Exception in Main'.

~~**E.**~~ Only one of the above is correct.

Explanation:

There are two points to understand here:
1. Even if the program is executed without any arguments, the 'args' is NOT NULL. In such case it will be initialized to an array of Strings containing zero elements.
2. The finally block is always executed, no matter how control leaves the try block. Only if, in a try or catch block, System.exit() is called then finally will not be executed.

Back to Question without Answer

84. QID - 2.1168 : Working with Inheritance

Consider the following class hierarchy:

```
A
|
B1,  B2
|
C1, C2
```

(B1 and B2 are subclasses of A and C1, C2 are subclasses of B1) Which of the following statements are correct? Assume that objectOfA, objectOfC1, etc. are objects of classes A and C1 respectively.

Correct Option is : B

~~**A.**~~ `objectOfC2 instanceof B2` will return `true`.

> `objectOfC2` is an instance of C2 and as C2 extends B1, it cannot be a subclass of B2 and so `objectOfC2` instanceof B2 cannot be `true`.

B. `objectOfC1 instanceof B1` will return `true`.

> This is because C1 extends B1. Therefore, anything that is a C1 is a B1. It is like saying a Dog is a Pet or a Cat is a Pet, if Dog and Cat extend from Pet.

~~**C.**~~ `objectOfA instanceof B1` will return `true`.

~~**D.**~~ C1 c1 = objectOfA; is a valid statement.

> Since c1 is declared of type C1, an object of class A, cannot be assigned to c1 because A is not a C1. A C1 is an A. So `A a = objectOfC1;` would have been valid.

E. ~~`B1 b1 = objectOfB2;`~~ is a valid statement.

B2 does not extend from B1 and so there is no is-a relation between B1 and B2. Therefore, an object of class B2 cannot be assigned to a variable of class B1.

Back to Question without Answer

85. QID - 2.1002 : Working with Inheritance

Given the following definitions and reference declarations:

```
interface I1 { }
interface I2 { }
class C1 implements I1 { }
class C2 implements I2 { }
class C3 extends C1 implements I2 { }
C1 o1;
C2 o2;
C3 o3;
```

Which of these statements are legal?

Correct Options are : A D E

A. `class C4 extends C3 implements I1, I2 { }`

> Although, the `implements I1, I2` is redundant here because C3 already implements I1 and I2, it is not invalid.

~~B.~~ `o3 = o1;`

> superclass reference cannot be assigned to subclass reference without explicit cast.

~~C.~~ `o3 = o2;`

> There is no way a reference of class C2 (which is o2) can point to an object of class C3 because C2 and C3 have no inheritance relationship. So this assignment is rejected at compile time itself.

D. `I1 i1 = o3; I2 i2 = (I2) i1;`

> This is valid because at run time `i1` actually refers to an object that implements I2.

E. `I1 b = o3;`

> Because C3 extends C1 which implements I1.

Back to Question without Answer

86. QID - 2.1337 : Working with Inheritance

Consider the following code:

```
class A{
   A() {  print();   }
   void print() { System.out.println("A"); }
}
class B extends A{
   int i =   Math.round(3.5f);
   public static void main(String[] args){
      A a = new B();
      a.print();
   }
   void print() { System.out.println(i); }
}
```

What will be the output when class B is run ?

Correct Option is : C

~~**A.**~~ It will print A, 4.

~~**B.**~~ It will print A, A

C. It will print 0, 4

~~**D.**~~ It will print 4, 4

~~**E.**~~ None of the above.

Explanation:

Note that method `print()` is overridden in class B. Due to polymorphism, the method to be executed is selected depending on the class of the actual object.
Here, when an object of class B is created, first A's constructor is called, which in turn calls `print()`. Now, since the class of actual object is B, B's `print()` is selected. At this point of time, variable i has not been initialized (because we are still initializing A at this point), so its default value i.e. 0 is printed.
This happens because the method `print()` is non-private, hence polymorphic.

Finally, 4 is printed.

Back to Question without Answer

87. QID - 2.920 : Java Basics - Garbage Collection

Which is the earliest line in the following code after which the object created on line // 1 can be garbage collected, assuming no compiler optimizations are done?

```
public class NewClass{
   private Object o;
   void doSomething(Object s){  o = s;   }

   public static void main(String args[]){
      Object obj = new Object(); // 1
      NewClass tc = new NewClass(); //2
      tc.doSomething(obj); //3
      obj = new Object();    //4
      obj = null;    //5
      tc.doSomething(obj); //6
   }
}
```

Correct Option is : F

~~**A.**~~ Line 1

~~**B.**~~ Line 2

~~**C.**~~ Line 3

~~**D.**~~ Line 4

~~**E.**~~ Line 5

F. Line 6

Before this line the object is being pointed to by at least one variable.

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling `System.gc();`

Back to Question without Answer

88. QID - 2.1140 : Working with Methods

Which of the following are valid declarations?

Correct Option is : A

A. `abstract int absMethod(int param) throws Exception;`

~~B.~~ `abstract native int absMethod(int param) throws Exception;`

> `native` method cannot be `abstract`.

~~C.~~ `float native getVariance() throws Exception;`

> return type should always be on the immediate left of method name.

~~D.~~ `abstract private int absMethod(int param) throws Exception;`

> `private` method cannot be `abstract`. A `private` method is not inherited so how can a subclass implement it?

~~E.~~ `strictfp float f;`

> The keyword `strictfp` can only be applied to class or method declarations.

Back to Question without Answer

89. QID - 2.1302 : Working with Java Data Types - String, StringBuilder

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      StringBuilder sb = new StringBuilder("12345678");
      sb.setLength(5);
      sb.setLength(10);
      System.out.println(sb.length());
   }
}
```

Correct Option is : B

~~**A.** It will print 5.~~

> Although it truncates the string to length 5 but setLength(10) will append 5 spaces (actually null chars i.e. \u0000).

B. It will print 10.

~~**C.** It will print 8.~~

~~**D.** Compilation error.~~

~~**E.** None of the above.~~

> The program will compile without error and will print 10 when run.

Explanation:

If you do `System.out.println(sb);` it will indeed print 12345 but the length will be 10.

From javadocs:

public void setLength(int newLength)
Sets the length of the character sequence. The sequence is changed to a new character sequence whose length is specified by the argument. For every nonnegative index k less than newLength, the character at index k in the new character sequence is the same as the character at index k in the old sequence if k is less than the length of the old character sequence; otherwise, it is the null character '\u0000'. In other words, if the newLength argument is less than the current length, the length is changed to the specified length.
If the newLength argument is greater than or equal to the current length, sufficient null characters ('\u0000') are appended so that length becomes the newLength argument.

The newLength argument must be greater than or equal to 0.

Parameters:
newLength - the new length
Throws:
IndexOutOfBoundsException - if the newLength argument is negative.

Back to Question without Answer

90. QID - 2.863 : Working with Methods

Consider the following code appearing in the same file:

```
class Data {
    private int x = 0, y = 0;
    public Data(int x, int y){
        this.x = x; this.y = y;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d = new Data(1, 1);
        //add code here
    }
}
```

Which of the following options when applied individually will change the Data object currently referred to by the variable d to contain 2, 2 as values for its data fields?

Correct Option is : E

~~**A.**~~ Add the following two statements :

```
d.x = 2;
d.y = 2;
```

> Note that x and y are private in class Data. Therefore, you cannot access these members from any other class.

~~**B.**~~ Add the following statement:

```
d = new Data(2, 2);
```

> This will create a new Data object and will not change the original Data object referred to be d.

~~C.~~ Add the following two statements:

```
d.x += 1;
d.y += 1;
```

> Note that x and y are private in class Data. Therefore, you cannot access these members from any other class.

~~D.~~ Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x.setInt(x);   this.y.setInt(y);
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

> x is primitive int.You cannot call any methods on a primitive. so this.x.setInt(...) or this.y.setInt(...) don't make any sense.

E. Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x = x;   this.y = y;
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

> This is a good example of encapsulation where the data members of Data class are private and there is a method in Data class to manipulate its data. Compare this approach to making x and y as public and letting other classes directly modify the values.

Back to Question without Answer

Test 4

01. QID - 2.960

What will the following code print when compiled and run?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        A: for(int i = 0; i < 2; i++){
            B: for(int j = 0; j < 2; j++){
                C: for(int k = 0; k < 3; k++){
                    c++;
                    if(k>j) break;
                }
            }
        }
        System.out.println(c);
    }
}
```

Select 1 option

A. 7

B. 8

C. 9

D. 10

E. 11

Check Answer

02. QID - 2.1349

Given a class named Test, which of these would be valid definitions for the constructors for the class?

Select 1 option

A. `Test(Test b) { }`

B. `Test Test( ) { }`

C. `private final Test( ) { }`

D. `void Test( ) { }`

E. `public static void Test(String args[ ] ) { }`

Check Answer

03. QID - 2.1327

What will be the output when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0; j < i; ++j, i++){
         System.out.println(i + " " + j);
      }
      System.out.println(i + " " + j);
   }
}
```

Select 1 option

A. 0 0 will be printed twice.

B. 0 0 will be printed once.

C. It will keep on printing 0 0

D. It will not compile.

E. It will print 0 0 and then 0 1.

Check Answer

04. QID - 2.1119

Given:

```
public class Switcher{

   public static void main(String[] args){
       switch(Integer.parseInt(args[1]))  //1
       {
          case 0 :
             boolean b = false;
             break;

          case 1 :
             b = true; //2
             break;
       }

       if(b) System.out.println(args[2]);
   }
}
```

What will the above program print if compiled and run using the following command line:

```
java Switcher 1 2 3
```

Select 1 option

A. It will print 1

B. It will print 2

C. It will print 3

D. It will not print anything.

E. It will not compile because of //1.

F. It will not compile because of //2.

G. It will not compile for some other reason.

Check Answer

05. QID - 2.1055

Identify valid method declarations.

Select 1 option

A. `public void methodX(int... i, String s);`

B. `public void methodX(int... i, String... s);`

C. `public void methodX(int i, int... j);`

D. `public int... methodX(int i);`

Check Answer

06. QID - 2.1151

Given that TestClass is a class, how many objects and reference variables are created by the following code?

```
TestClass t1, t2, t3, t4;
t1 = t2 = new TestClass();
t3 = new TestClass();
```

Select 1 option

A. 2 objects, 3 references.

B. 2 objects, 4 references.

C. 3 objects, 2 references.

D. 2 objects, 2 references.

E. None of the above.

Check Answer

07. QID - 2.1243

Which of the following are correct ways to initialize the static variables MAX and CLASS_GUID ?

```
class Widget{
   static int MAX;      //1
   static final String CLASS_GUID;    // 2
   Widget(){
       //3
   }
   Widget(int k){
       //4
   }
}
```

Select 2 options

A. Modify lines //1 and //2 as : `static int MAX = 111; static final String CLASS_GUID = "XYZ123";`

B. Add the following line just after //2 : `static {  MAX = 111; CLASS_GUID = "XYZ123"; }`

C. Add the following line just before //1 : `{ MAX = 111; CLASS_GUID = "XYZ123"; }`

D. Add the following line at //3 as well as //4 : `MAX = 111; CLASS_GUID = "XYZ123";`

E. Only option 3 is valid.

Check Answer

08. QID - 2.984

Following is a supposedly robust method to parse an input for a float :

```
public float parseFloat(String s){
   float f = 0.0f;
   try{
      f = Float.valueOf(s).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      System.out.println("Invalid input " + s);
      f = Float.NaN ;
      return f;
   }
   finally { System.out.println("finally");  }
   return f ;
}
```

Which of the following statements about the above method are true??

Select 1 option

A. If input is `"0.1"` then it will return `0.1` and print `finally`.

B. If input is `"0x.1"` then it will return `Float.Nan` and print `Invalid Input 0x.1` and `finally`.

C. If input is `"1"` then it will return `1.0` and print `finally`.

D. If input is `"0x1"` then it will return `0.0` and print `Invalid Input 0x1` and `finally`.

E. The code will not compile.

Check Answer

09. QID - 2.1319

Given the following code snippet:

```
int rate = 10;
int t = 5;
XXX amount = 1000.0;
for(int i=0; i<t; t++){
   amount = amount*(1 - rate/100);
}
```

What can XXX be?

Select 1 option

A. int

B. long

C. only double

D. double or float

E. float

Check Answer

10. QID - 2.1178

Which one of the following class definitions is/are a legal definition of a class that cannot be instantiated?

```
class Automobile{
   abstract void honk();  //(1)
}

abstract class Automobile{
   void honk();   //(2)
}

abstract class Automobile{
   void honk(){};   //(3)
}

abstract class Automobile{
   abstract void honk(){}   //(4)
}

abstract class Automobile{
   abstract void honk();   //(5)
}
```

Select 2 options

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

11. QID - 2.1267

Which of the following implementations of a `max()` method will correctly return the largest value?

Select 1 option

A.
```
int max(int x, int y){
    return(  if(x > y){ x; } else{ y; }  );
}
```

B.
```
int max(int x, int y){
    return( if(x > y){ return x; }  else{ return y; } );
}
```

C.
```
int max(int x, int y){
    switch(x < y){
       case true:
              return y;
       default :
              return x;
    };
}
```

D.
```
int max(int x, int y){
     if (x > y)  return x;
     return y;
}
```

E. None of the above.

Check Answer

12. QID - 2.1322

What is meant by "encapsulation" ?

Select 1 option

A. There is no way to access member variable.

B. There are no member variables.

C. Member fields are declared private but public accessor/mutator methods are provided to access and change their values.

D. Data fields are declared public and accessor methods are provided to access and change their values.

E. None of the above.

Check Answer

13. QID - 2.1086

What will be the result of compiling and running the following code?

```
class Base{
   public Object getValue(){ return new Object(); } //1
}

class Base2 extends Base{
   public String getValue(){ return "hello"; } //2
}

public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Select 1 option

A. It will print the hash code of the object.

B. It will print `hello`.

C. Compile time error at `//1`.

D. Compile time error at `//2`.

E. Compile time error at `//3`.

Check Answer

14. QID - 2.1077

What will the following program print?

```
public class TestClass{
  static boolean b;
  static int[] ia = new int[1];
  static char ch;
  static boolean[] ba = new boolean[1];
  public static void main(String args[]) throws Exception{
    boolean x = false;
    if( b ){
      x = ( ch == ia[ch]);
    }
    else x = ( ba[ch] = b );
    System.out.println(x+" "+ba[ch]);
  }
}
```

Select 1 option

A. true true

B. true false

C. false true

D. false false

E. It will not compile.

Check Answer

15. QID - 2.1359

Consider :

```
class A {  public void perform_work(){}  }
class B extends A {  public void perform_work(){}  }
class C extends B {  public void perform_work(){}  }
```

How can you let `perform_work()` method of A to be called from an instance method in C?

Select 1 option

A. `( (A) this ).perform_work( );`

B. `super.perform_work( );`

C. `super.super.perform_work( );`

D. `this.super.perform_work( );`

E. It is not possible.

Check Answer

16. QID - 2.1336

Which of these methods are not a part of the String class?

Select 1 option

A. `trim( )`

B. `length( )`

C. `concat(String)`

D. `hashCode( )`

E. `reverse( )`

Check Answer

17. QID - 2.1277

Which of the following statements are true?

Select 2 options

A. `System.out.println(1 + 2 + "3");` would print 33.

B. `System.out.println("1" + 2 + 3);` would print 15.

C. `System.out.println(4 + 1.0f);` would print 5.0

D. `System.out.println(5/4);` would print 1.25

E. `System.out.println('a' + 1 );` would print b.

Check Answer

18. QID - 2.1306

Which of these statements concerning interfaces are true?

Select 2 options

A. An interface may extend an interface.

B. An interface may extend a class and may implement an interface.

C. A class can implement an interface and extend a class.

D. A class can extend an interface and can implement a class.

E. An interface can only be implemented and cannot be extended.

Check Answer

19. QID - 2.1362

Which of the following statements are true?

Select 2 options

A. private keyword can never be applied to a class.

B. synchronized keyword can never be applied to a class.

C. synchronized keyword may be applied to a non-primitive variable.

D. final keyword can never be applied to a class.

E. A final variable can be shadowed in a subclass.

Check Answer

20. QID - 2.865

What will the following program print when compiled and run?

```
class Data {
    private int x = 0;
    private String y = "Y";
    public Data(int k){
        this.x = k;
    }
    public Data(String k){
        this.y = k;
    }
    public void showMe(){
        System.out.println(x+y);
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        new Data(10).showMe();
        new Data("Z").showMe();
    }
}
```

Select 1 option

A. 0Z
10Y

B. 10Y
0Z

C. It will not compile.

D. It will throws an exception at run time.

Check Answer

21. QID - 2.1234

Consider the following code:

```
public abstract class TestClass{
    public abstract void m1();
    public abstract void m2(){
        System.out.println("hello");
    }
}
```

Which of the following corrections can be applied to the above code (independently) so that it compiles without any error?

Select 2 options

A. Replace the method body of m2() with a ; (semi-colon).

B. Replace the ; at the end of m1() with a method body.

C. Remove abstract from m2().

D. Remove abstract from the class declaration.

Check Answer

22. QID - 2.974

What will be the contents of s1 and s2 at the time of the println statement in the main method of the following program?

```
import java.util.*;
public class TestClass{
   public static void main(String args[]){
      Stack s1 = new Stack ();
      Stack s2 = new Stack ();
      processStacks (s1,s2);
      System.out.println (s1 + "    "+ s2);
   }
   public static void processStacks(Stack x1, Stack x2){
      x1.push (new Integer ("100")); //assume that the method push a
      x2 = x1;
   }
 }
```

Select 1 option

A. [100] [100]

B. [100] []

C. [] [100]

D. [] []

Check Answer

23. QID - 2.1204

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
  public static void main(String args[ ] ){
    Object a, b, c ;
    a = new String("A");
    b = new String("B");
    c = a;
    a = b;
    System.out.println(""+c);
  }
}
```

Select 1 option

A. The program will print `java.lang.String@XXX`, where XXX is the memory location of the object a.

B. The program will print A

C. The program will print B

D. The program will not compile as a,b and c are of type Object.

E. The program will print `java.lang.String@XXX`, where XXX is the hash code of the object a.

Check Answer

24. QID - 2.892

Which of the following are valid classes?

Select 1 option

A.
```
public class ImaginaryNumber extends Number {
}
```

B.
```
public class ThreeWayBoolean extends Boolean {
}
```

C.
```
public class NewSystem extends System {
}
```

D.
```
public class ReverseString extends String {
}
```

Check Answer

25. QID - 2.906

Given:

```
class Triangle{
    public int base;
    public int height;
    private final double ANGLE;

    public  void setAngle(double a){  ANGLE = a;  }

    public static void main(String[] args) {
        Triangle t = new Triangle();
        t.setAngle(90);
    }
}
```

Select 1 option

A. the value of `ANGLE` will not be set to `90` by the `setAngle` method.

B. An exception will be thrown at run time.

C. The code will work as expected setting the value of `ANGLE` to `90`.

D. The code will not compile.

Check Answer

26. QID - 2.1294

What will the code shown below print when run?

```
public class TestClass{
    static class Wrapper{
        int w = 10;
    }

    static Wrapper changeWrapper(Wrapper w){
        w = new Wrapper();
        w.w += 9;
        return w;
    }


    public static void main(String[] args){
        Wrapper w = new Wrapper();
        w.w = 20;
        changeWrapper(w);
        w.w += 30;
        System.out.println(w.w);
        w = changeWrapper(w);
        System.out.println(w.w);
     }
}
```

Select 2 options

A. 9

B. 19

C. 30

D. 20

E. 29

F. 50

Check Answer

27. QID - 2.1020

What does the zeroth element of the string array passed to the standard main method contain?

Select 1 option

A. The name of the class.

B. The string "java".

C. The number of arguments.

D. The first argument of the argument list, if present.

E. None of the above.

Check Answer

28. QID - 2.965

Consider the following program:

```
public class TestClass{
   public static void main(String[] args)  {     calculate(2);     }
   public static void calculate(int x){
      String val;
      switch(x){
         case 2:
         default:
         val = "def";
      }
      System.out.println(val);
   }
}
```

What will happen if you try to compile and run the program?

Select 2 options

A. It will not compile saying that variable `val` may not have been initialized..

B. It will compile and print `def`

C. As such it will not compile but it will compile if `calculate(2);` is replaced by `calculate(3);`

D. It will compile for any int values in `calculate(...);`

Check Answer

29. QID - 2.1110

Which of these statements are true?

Select 2 options

A. A static method can call other non-static methods in the same class by using the 'this' keyword.

B. A class may contain both static and non-static variables and both static and non-static methods.

C. Each object of a class has its own copy of each non-static member variable.

D. Instance methods may access local variables of static methods.

E. All methods in a class are implicitly passed a 'this' parameter when called.

Check Answer

30. QID - 2.1081

Which of the following code snippets will print exactly 10?

```
1.  Object t = new Integer(106);
    int k = ((Integer) t).intValue()/10;
    System.out.println(k);

2.  System.out.println(100/9.9);

3.  System.out.println(100/10.0);

4.  System.out.println(100/10);

5.  System.out.println(3 + 100/10*2-13);
```

Select 3 options

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

31. QID - 2.930

Expression `(s instanceof java.util.Date)` will return false if 's' was declared as a variable of class `java.lang.String`.

Select 1 option

A. True

B. False

Check Answer

32. QID - 2.1129

Consider the following class:

```
public class PortConnector{
   public PortConnector(int port) throws IOException{
   ...lot of valid code.
   }
   ...other valid code.
}
```

You want to write another class `CleanConnector` that extends from `PortConnector`. Which of the following statements should hold true for `CleanConnector` class?

Select 1 option

A. It is not possible to define `CleanConnector` that does not throw `IOException` at instantiation.

B. `PortConnector` class itself is not valid because you cannot throw any exception from a constructor.

C. `CleanConnector`'s constructor cannot throw any exception other than `IOException`.

D. `CleanConnector`'s constructor cannot throw any exception other than subclass of `IOException`.

E. `CleanConnector`'s constructor cannot throw any exception other than superclass of `IOException`.

F. None of these.

Check Answer

33. QID - 2.1188

What should be the return type of the following method?

```
public RETURNTYPE methodX( byte by){
    double d = 10.0;
    return (long) by/d*3;
}
```

Select 1 option

A. int

B. long

C. double

D. float

E. byte

Check Answer

34. QID - 2.1192

What will be the output of the following lines ?

```
System.out.println("" +5 + 6);    //1
System.out.println(5 + "" +6);    // 2
System.out.println(5 + 6 +"");    // 3
System.out.println(5 + 6);        // 4
```

Select 1 option

A. 56, 56, 11, 11

B. 11, 56, 11, 11

C. 56, 56, 56, 11

D. 56, 56, 56, 56

E. 56, 56, 11, 56

Check Answer

35. QID - 2.1047

An overriding method must have a same parameter list and the same return type as that of the overridden method.

Select 1 option

A. True

B. False

Check Answer

36. QID - 2.1355

Given the following code, which of these constructors can be added to class B without causing a compile time error?

```
class A{
   int i;
   public A(int x) { this.i = x; }
}
class B extends A{
   int j;
   public B(int x, int y) { super(x);  this.j = y; }
}
```

Select 2 options

A. `B( ) { }`

B. `B(int y ) { j = y; }`

C. `B(int y ) { super(y*2 ); j = y; }`

D. `B(int y ) { i = y; j = y*2; }`

E. `B(int z ) { this(z, z); }`

Check Answer

37. QID - 2.914

Consider the following code:

```
public static void main(String[] args) {
   int[] values = { 10, 30, 50 };
        for( int val : values ){
            int x = 0;
            while(x<values.length){
             System.out.println(x+" "+val);
              x++;
            }
        }
    }
```

How many times is 2 printed out?

Select 1 option

A. 0

B. 1

C. 2

D. 3

Check Answer

38. QID - 2.989

Consider the following code:

```
public class Logger{
    private StringBuilder sb = new StringBuilder();

    public void logMsg(String location, String message){
    sb.append(location);
    sb.append("-");
    sb.append(message);
    }

    public void dumpLog(){
    System.out.println(sb.toString());
    //Empty the contents of sb here
    }

}
```

Which of the following options will empty the contents of the StringBuilder referred to by variable sb in method dumpLog()?

Select 1 option

A. `sb.delete(0, sb.length());`

B. `sb.clear();`

C. `sb.empty();`

D. `sb.removeAll();`

E. `sb.deleteAll();`

Check Answer

39. QID - 2.1212

Which of the following expressions will evaluate to true if preceded by the following code?

```
String a = "java";
    char[] b = { 'j', 'a', 'v', 'a' };
    String c = new String(b);
    String d = a;
```

Select 3 options

A. `(a == d)`

B. `(b == d)`

C. `(a == "java")`

D. `a.equals(c)`

Check Answer

40. QID - 2.1312

Which of the following code fragments are valid method declarations?

Select 1 option

A. void method1{ }

B. void method2( ) { }

C. void method3(void){ }

D. method4{ }

E. method5(void){ }

Check Answer

41. QID - 2.1076

What would be the result of compiling and running the following program?

```
class SomeClass{
   public static void main(String args[]){
      int size = 10;
      int[] arr = new int[size];
      for (int i = 0 ; i < size ; ++i) System.out.println(arr[i]);
   }
}
```

Select 1 option

A. The code will fail to compile, because the `int[]` array declaration is incorrect.

B. The program will compile, but will throw an `IndexArrayOutOfBoundsException` when run.

C. The program will compile and run without error, and will print nothing.

D. The program will compile and run without error and will print `null` ten times.

E. The program will compile and run without error and will print `0` ten times.

Check Answer

42. QID - 2.1195

Which letters will be printed when the following program is run?

```
public class TestClass{
  public static void main(String args[]){
    B b = new C();
    A a = b;
    if (a instanceof A) System.out.println("A");
    if (a instanceof B) System.out.println("B");
    if (a instanceof C) System.out.println("C");
    if (a instanceof D) System.out.println("D");
  }
}
class A { }
class B extends A { }
class C extends B { }
class D extends C { }
```

Select 3 options

A. A will be printed.

B. B will be printed.

C. C will be printed.

D. D will be printed.

Check Answer

43. QID - 2.1304

Which of these are not part of the StringBuilder class?

Select 1 option

A. `trim( )`

B. `ensureCapacity(int )`

C. `append(boolean)`

D. `reverse( )`

E. `setLength(int)`

Check Answer

44. QID - 2.1142

Which of the given options should be inserted at line 1 so that the following code can compile without any errors?

```
package objective1;
// 1
public class StaticImports{

    public StaticImports(){
    out.println(MAX_VALUE);
    }

}
```

(Assume that java.lang.Integer has a static field named MAX_VALUE)

Select 2 options

A. `import static java.lang.Integer.*;`

B. `static import java.lang.System.out;`

C. `static import Integer.MAX_VALUE;`

D. `import static java.lang.System.*;`

E. `static import java.lang.System.*;`

Check Answer

45. QID - 2.1296

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    String str = "111";
    boolean[] bA = new boolean[1];
    if( bA[0] ) str = "222";
    System.out.println(str);
  }
}
```

Select 1 option

A. 111

B. 222

C. It will not compile as `bA[0]` is uninitialized.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

46. QID - 2.1123

Consider the contents of following two files:

```
//File A.java
package a;
public class A{
   A(){ }
   public void  print(){ System.out.println("A"); }
}

//File B.java
package b;
import a.*;
public class B extends A{
   B(){ }
   public void  print(){ System.out.println("B"); }
   public static void main(String[] args){
      new B();
   }
}
```

What will be printed when you try to compile and run class B?

Select 1 option

A. It will print A.

B. It will print B.

C. It will not compile.

D. It will compile but will not run.

E. None of the above.

Check Answer

47. QID - 2.1145

Assume the following declarations:

```
class A{ }
class B extends A{ }
class C extends B{ }

class X{
   B getB(){ return new B(); }
}

class Y extends X{
  // method declaration here
}
```

Which of the following methods can be inserted in class Y?

Select 2 options

A. `public C getB(){ return new B(); }`

B. `protected B getB(){ return new C(); }`

C. `C getB(){ return new C(); }`

D. `A getB(){ return new A(); }`

Check Answer

48. QID - 2.1005

Consider the following code:

```
class A {
    public void doA(int k) throws Exception {  // 0
        for(int i=0; i< 10; i++) {
            if(i == k) throw new Exception("Index of k is "+i); // 1
        }
    }
    public void doB(boolean f) { // 2
        if(f) {
            doA(15); // 3
        }
        else return;
    }
    public static void main(String[] args) { // 4
        A a = new A();
        a.doB(args.length>0); // 5
    }
}
```

Which of the following statements are correct?

Select 1 option

A. This will compile and run without any errors or exception.

B. This will compile if `throws Exception` is added at line //2

C. This will compile if `throws Exception` is added at line //4

D. This will compile if `throws Exception` is added at line //2 as well as //4

E. This will compile if line marked // 1 is enclosed in a try - catch block.

Check Answer

49. QID - 2.831

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
        System.out.println(this.myValue);
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
        System.out.println(this.myValue);
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showOne(100);
        ct.showTwo(200);
    }
}
```

Select 1 option

A. 0 followed by 100.

B. 100 followed by 100.

C. 0 followed by 200.

D. 100 followed by 200.

Check Answer

50. QID - 2.1113

Which of the following is a legal return type of a method overriding the given method:

```
public Object  myMethod() {...}
```
(Select the best option.)

Select 1 option

A. Object

B. String

C. Return type can be any object since all objects can be cast to Object.

D. void

E. None of the above.

Check Answer

51. QID - 2.1040

What will the following class print ?

```
class InitTest{
   public static void main(String[] args){
      int a = 10;
      int b = 20;
      a += (a = 4);
      b = b + (b = 5);
      System.out.println(a+ ",  "+b);
   }
}
```

Select 1 option

A. It will print 8, 25

B. It will print 4, 5

C. It will print 14, 5

D. It will print 4, 25

E. It will print 14, 25

Check Answer

52. QID - 2.1215

Consider the following classes :

```
class A{
   public void mA(){ };
}

class B extends A {
   public void mA(){ }
   public void mB() { }
}

class C extends B {
   public void mC(){ }
}
```

and the following declarations:

```
A x = new B(); B y = new B(); B z = new C();
```

Which of the following calls are polymorphic calls?

Select 3 options

A. `x.mA();`

B. `x.mB();`

C. `y.mA();`

D. `z.mC();`

E. `z.mB();`

Check Answer

53. QID - 2.968

Which of these expressions will obtain the substring "456" from a string defined by `String str = "01234567"`?

Select 1 option

A. `str.substring(4, 7)`

B. `str.substring(4)`

C. `str.substring(3, 6)`

D. `str.substring(4, 6)`

E. `str.substring(4, 3)`

Check Answer

54. QID - 2.1102

Under what situations does a class get a default constructor?

Select 1 option

A. All classes in Java get a default constructor.

B. You have to define at least one constructor to get the default constructor.

C. If the class does not define any constructors explicitly.

D. All classes get default constructor from Object class.

E. None of the above.

Check Answer

55. QID - 2.829

What will the following program print when compiled and run:

```
public class TestClass {
    public static void main(String[] args) {
            someMethod();
    }

    static void someMethod(Object parameter)  {
          System.out.println("Value is "+parameter);
    }
}
```

Select 1 option

A. It will not compile.

B. `Value is null`

C. `Value is`

D. It will throw a `NullPointerException` at run time.

Check Answer

56. QID - 2.913

Which of the following statements about an array are correct?

Select 1 option

A. An array can dynamically grow in size.

B. Arrays can be created only for primitive types.

C. Every array has a built in property named 'size' which tells you the number of elements in the array.

D. Every array has in implicit method named '`length`' which tells you the number of elements in the array.

E. Element indexing starts at 0.

Check Answer

57. QID - 2.1211

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      Exception e = null;
      throw e;
   }
}
```

Select 1 option

A. The code will fail to compile.

B. The program will fail to compile, since it cannot throw a `null`.

C. The program will compile without error and will throw an `Exception` when run.

D. The program will compile without error and will throw `java.lang.NullPointerException` when run

E. The program will compile without error and will run and terminate without any output.

Check Answer

58. QID - 2.1101

Consider the following code:

```
class Base{
 private float f = 1.0f;
 void setF(float f1){ this.f = f1; }
}
class Base2 extends Base{
 private float f = 2.0f;
 //1
}
```

Which of the following options is a valid example of overriding?

Select 2 options

A. `protected void setF(float f1){ this.f = 2*f1; }`

B. `public void setF(double f1){ this.f = (float) 2*f1; }`

C. `public void setF(float f1){ this.f = 2*f1; }`

D. `private void setF(float f1){ this.f = 2*f1; }`

E. `float setF(float f1){ this.f = 2*f1; return f;}`

Check Answer

59. QID - 2.837

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Bird) System.out.println("f is a Bird");
        if(e instanceof Flyer) System.out.println("e is a Flyer");
        if(b instanceof Flyer) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Select 1 option

A. It will not compile.

B. It will throw an exception when run.

C. f is a Bird
e is a Flyer

D. f is a Bird

E. e is a Flyer

Check Answer

60. QID - 2.845

The options below contain the complete contents of a file (the name of the file is not specified).
Which of these options can be run with the following command line once compiled?

```
java main
```

Select 1 option

A.
```
//in file main.java
class main {
   public void main(String[] args) {
       System.out.println("hello");
   }
}
```

B.
```
//in file main.java
   public static void main4(String[] args) {
       System.out.println("hello");
   }
```

C.
```
//in file main.java
public class anotherone{
}
class main {
   public static void main(String[] args) {
       System.out.println("hello");
   }
}
```

D.
```
//in file main.java
class anothermain{
   public static void main(String[] args) {
       System.out.println("hello2");
```

```
    }
}
class main {
    public final static void main(String[] args) {
        System.out.println("hello");
    }
}
```

Check Answer

61. QID - 2.859

What will the following code print when run?

```
public class TestClass {

    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "c" : System.out.println( "cat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("c");
    }
}
```

Select 1 option

A. apple
cat
none

B. apple
cat

C. cat
none

D. cat

Check Answer

62. QID - 2.1038

What will be the output of the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true ;
      boolean bool2 = false;
      boolean bool  = false;
      bool = ( bool2 &  method1(i++) ); //1
      bool = ( bool2 && method1(i++) ); //2
      bool = ( bool1 |  method1(i++) ); //3
      bool = ( bool1 || method1(i++) ); //4
      System.out.println(i);
   }
   public static boolean method1(int i){
       return i>0 ? true : false;
   }
}
```

Select 1 option

A. It will print 1.

B. It will print 2.

C. It will print 3.

D. It will print 4.

E. It will print 0.

Check Answer

63. QID - 2.1219

What will be the output of compiling and running the following program:

```
class TestClass implements I1, I2{
   public void m1() { System.out.println("Hello"); }
   public static void main(String[] args){
      TestClass tc = new TestClass();
      ( (I1) tc).m1();
   }
}
interface I1{
   int VALUE = 1;
   void m1();
}
interface I2{
   int VALUE = 2;
   void m1();
}
```

Select 1 option

A. It will print `Hello`.

B. There is no way to access any `VALUE` in TestClass.

C. The code will work fine only if `VALUE` is removed from one of the interfaces.

D. It will not compile.

E. None of the above.

Check Answer

64. QID - 2.1326

What will happen when the following code is compiled and run?

```
class AX{
  static int[] x = new int[0];
  static{
   x[0] = 10;
  }
  public static void main(String[] args){
    AX ax = new AX();
  }
}
```

Select 1 option

A. It will throw `NullPointerException` at runtime.

B. It will throw `ArrayIndexOutOfBoundsException` at runtime.

C. It will throw `ExceptionInInitializerError` at runtime.

D. It will not compile.

Check Answer

65. QID - 2.1132

Which of the following is/are illegal Java identifier(s)?

Select 1 option

A. num

B. int123

C. 2Next

D. _interface

E. a$_123

Check Answer

66. QID - 2.1228

What will be the result of attempting to compile and run the following class?

```
public class InitTest{
   static String s1 = sM1("a");{
      s1 = sM1("b");
   }
   static{
      s1 = sM1("c");
   }
   public static void main(String args[]){
      InitTest it = new InitTest();
   }
   private static String sM1(String s){
      System.out.println(s);  return s;
   }
}
```

Select 1 option

A. The program will fail to compile.

B. The program will compile without error and will print `a, c` and `b` in that order when run.

C. The program will compile without error and will print `a, b` and `c` in that order when run.

D. The program will compile without error and will print `c, a` and `b` in that order when run.

E. The program will compile without error and will print `b, c` and `a` in that order when run.

Check Answer

67. QID - 2.992

Given the following program, which statement is true?

```
class SomeClass{
   public static void main( String args[ ] ){
      if (args.length == 0 ){
         System.out.println("no arguments") ;
      }
      else{
         System.out.println( args.length + " arguments") ;
      }
   }
}
```

Select 1 option

A. The program will fail to compile.

B. The program will throw a `NullPointerException` when run with zero arguments.

C. The program will print `no arguments` and `1 arguments` when called with zero and one arguments.

D. The program will print `no arguments` and `2 arguments` when called with zero and one arguments.

E. The program will print `no arguments` and `3 arguments` when called with zero and one arguments.

Check Answer

68. QID - 2.1036

Consider the following class...

```
class Test{
   public static void main(String[ ] args){
      int[] a = { 1, 2, 3, 4 };
      int[] b = { 2, 3, 1, 0 };
      System.out.println( a [ (a = b)[3] ] );
   }
}
```

What will it print when compiled and run ?

Select 1 option

A. It will not compile.

B. It will throw `ArrayIndexOutOfBoundsException` when run.

C. It will print 1.

D. It will print 3.

E. It will print 4

Check Answer

69. QID - 2.904

Given:

```
class Triangle{
    public int base;
    public int height;
    private static double ANGLE;

    public static double getAngle();

    public static void Main(String[] args) {
        System.out.println(getAngle());
    }
}
```

Identify the correct statements:

Select 1 option

A. It will not compile because it does not implement `setAngle` method.

B. It will not compile because `ANGLE` cannot be private.

C. It will not compile because `getAngle()` has no body.

D. It will not compile because `ANGLE` field is not initialized.

E. It will not compile because of the name of the method `Main` instead of `main`.

Check Answer

70. QID - 2.995

Which of these statements about interfaces are true?

Select 3 options

A. Interfaces are abstract by default.

B. An interface can have static methods.

C. All methods in an interface are abstract although you need not declare them to be so.

D. Fields of an interface may be declared as transient or volatile but not synchronized.

E. interfaces cannot be final.

Check Answer

71. QID - 2.1050

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      for (i=1 ;  i<5  ; i++) continue;  //(1)
      for (i=0 ;       ; i++) break;       //(2)
      for (    ; i<5?false:true ;    );     //(3)
   }
}
```

Select 1 option

A. The code will compile without error and will terminate without problems when run.

B. The code will fail to compile, since the `continue` can't be used this way.

C. The code will fail to compile, since the `break` can't be used this way

D. The code will fail to compile, since the `for` statement in the line 2 is invalid.

E. The code will compile without error but will never terminate.

Check Answer

72. QID - 2.1313

What is the result of executing the following code when the value of i is 5:

```
switch (i){
    default:
    case 1:
        System.out.println(1);
    case 0:
        System.out.println(0);
    case 2:
        System.out.println(2);
        break;
    case 3:
        System.out.println(3);
}
```

Select 1 option

A. It will print `1 0 2`

B. It will print `1 0 2 3`

C. It will print `1 0`

D. It will print `1`

E. Nothing will be printed.

Check Answer

73. QID - 2.1162

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){}
}
```

Select 1 option

A. The class `TestClass` cannot be declared `abstract`.

B. The variable `j` cannot be declared `transient`.

C. The variable `k` cannot be declared `synchronized`.

D. The constructor `TestClass( )` cannot be declared final.

E. The method `f( )` cannot be declared static.

F. This code has no errors.

Check Answer

74. QID - 2.1308

What is the result of compiling and running the following code ?

```
public class TestClass{
   static int si = 10;
   public static void main (String args[]){
      new TestClass();
   }
   public TestClass(){
      System.out.println(this);
   }
   public String toString(){
      return "TestClass.si = "+this.si;
   }
}
```

Select 1 option

A. The class will not compile because you cannot override `toString()` method.

B. The class will not compile as `si` being `static`, `this.si` is not a valid statement.

C. It will print `TestClass@nnnnnnnn`, where `nnnnnnn` is the hash code of the TestClass object referred to by 'this'.

D. It will print `TestClass.si = 10;`

E. None of the above.

Check Answer

75. QID - 2.932

Which of the following statements are true?

Select 2 options

A. The modulus operator % can only be used with integer operands.

B. & can have integral as well as boolean operands.

C. The arithmetic operators *, / and % have the same level of precedence.

D. && can have integer as well as boolean operands.

E. ~ can have integer as well as boolean operands.

Check Answer

76. QID - 2.971

What will be the result of trying to compile and execute of the following program?

```
public class TestClass{
   public static void main(String args[] ){
      int i = 0 ;
      int[] iA = {10, 20} ;
      iA[i] = i = 30 ;
      System.out.println(""+ iA[ 0 ] + " " + iA[ 1 ] + "  "+i) ;
    }
}
```

Select 1 option

A. It will throw ArrayIndexOutofBoundsException at Runtime.

B. Compile time Error.

C. It will prints 10 20 30

D. It will prints 30 20 30

E. It will prints 0 20 30

Check Answer

77. QID - 2.1201

Consider the following interface definition:

```
interface Bozo{
         int type = 0;
         public void jump();
}

Now consider the following class:

public class Type1Bozo implements Bozo{
         public Type1Bozo(){
            type = 1;
         }

         public void jump(){
            System.out.println("jumping..."+type);
         }

         public static void main(String[] args){
            Bozo b = new Type1Bozo();
            b.jump();
         }
}
```

What will the program print when compiled and run?

Select 1 option

A. jumping...0

B. jumping...1

C. This program will not compile.

D. It will throw an exception at runtime.

Check Answer

78. QID - 2.868

How can you initialize a StringBuilder to have a capacity of at least 100 characters?

Select 2 options

A.
```
StringBuilder sb = new StringBuilder(100);
```

B.
```
StringBuilder sb = StringBuilder.getInstance(100);
```

C.
```
StringBuilder sb = new StringBuilder();
sb.setCapacity(100);
```

D.
```
StringBuilder sb = new StringBuilder();
sb.ensureCapacity(100);
```

Check Answer

79. QID - 2.854

What will be printed when the following code is compiled and run?

```
class A {
    public int getCode(){ return 2;}
}

class AA extends A {
  public long getCode(){ return 3;}
}

public class TestClass {

    public static void main(String[] args) throws Exception {
         A a = new A();
        A aa = new AA();
        System.out.println(a.getCode()+" "+aa.getCode());
    }

    public int getCode() {
        return 1;
    }
}
```

Select 1 option

A. 2 3

B. 2 2

C. It will throw an exception at run time.

D. The code will not compile.

Check Answer

80. QID - 2.990

Consider the following classes...

```
class Teacher{
      void teach(String student){
          /* lots of code */
      }
}
class Prof extends Teacher{
        //1
}
```

Which of the following methods can be inserted at line //1 ?

Select 4 options

A. `public void teach() throws Exception`

B. `private void teach(int i) throws Exception`

C. `protected void teach(String s)`

D. `public final void teach(String s)`

E. `public abstract void teach(String s)`

Check Answer

81. QID - 2.1153

What will the following program print?

```
public class TestClass{
  static String str;
  public static void main(String[] args){
     System.out.println(str);
  }
}
```

Select 1 option

A. It will not compile.

B. It will compile but throw an exception at runtime.

C. It will print 'null' (without quotes).

D. It will print nothing.

E. None of the above.

Check Answer

82. QID - 2.1317

Which of the following are NOT valid operators in Java?

Select 4 options

A. sizeof

B. <<<

C. instanceof

D. mod

E. equals

Check Answer

83. QID - 2.843

Given the following declaration:

```
int[][] twoD = { { 1, 2, 3} , { 4, 5, 6, 7}, { 8, 9, 10 } };
```

What will the following lines of code print?

```
System.out.println(twoD[1].length);
System.out.println(twoD[2].getClass().isArray());
System.out.println(twoD[1][2]);
```

Select 1 option

A.
```
4
true
6
```

B.
```
3
true
3
```

C.
```
3
false
3
```

D.
```
4
false
6
```

Check Answer

84. QID - 2.1121

What will the following code print?

```
void crazyLoop(){
  int c = 0;
  JACK: while (c < 8){
    JILL: System.out.println(c);
    if (c > 3) break JACK; else c++;
  }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. It will print numbers from 0 to 8

D. It will print numbers from 0 to 3

E. It will print numbers from 0 to 4

Check Answer

85. QID - 2.1199

Given:

```
public class TestClass{
  public static int getSwitch(String str){
      return (int) Math.round( Double.parseDouble(str.substring(1, s
  }
  public static void main(String args []){
    switch(getSwitch(args[0])){
      case 0 : System.out.print("Hello ");
      case 1 : System.out.print("World"); break;
      default : System.out.print(" Good Bye");
    }
  }
}
```

What will be printed by the above code if it is run with command line:
`java TestClass --0.50`
(There are two minuses before 0.)

Select 1 option

A. Hello

B. World

C. Hello World

D. Hello World Good Bye

E. Good Bye

Check Answer

86. QID - 2.1261

Consider the following code:

```
public class SubClass extends SuperClass{
     int i, j, k;
     public SubClass( int m, int n )     {  i = m ;  j = m ;  } //1
     public SubClass( int m )  {   super(m );   } //2
 }
```

Which of the following constructors MUST exist in SuperClass for SubClass to compile correctly?

Select 2 options

A. It is ok even if no explicit constructor is defined in SuperClass

B. `public SuperClass(int a, int b)`

C. `public SuperClass(int a)`

D. `public SuperClass()`

E. only `public SuperClass(int a)` is required.

Check Answer

87. QID - 2.1062

The following code snippet will print 4.

```
int i1 = 1, i2 = 2, i3 = 3;
int i4 = i1 + (i2=i3 );
System.out.println(i4);
```

Select 1 option

A. True

B. False

Check Answer

88. QID - 2.1269

Which of the following will not give any error at compile time and run time?

Select 4 options

A. `if (8 == 81) {}`

B. `if (x = 3) {}` // assume that x is an int

C. `if (true) {}`

D. `if (bool = false) {}` //assume that bool is declared as a boolean

E. `if (x == 10 ? true:false) { }` // assume that x is an int

Check Answer

89. QID - 2.852

What will the following code print?

```
System.out.println("12345".charAt(6));
```

Select 1 option

A. `5`

B. `null`

C. `-1`

D. It will throw an `ArrayIndexOutOfBoundsException`.

E. It will throw a `StringOutOfBoundsException`.

F. It will throw an `IndexOutOfBoundsException`

Check Answer

90. QID - 2.1373

What should be placed in the two blanks so that the following code will compile without errors:

```
class XXX{
    public void m() throws Exception{
        throw new Exception();
    }
}
class YYY extends XXX{
    public void m() {
    }
}
public class TestClass {
    public static void main(String[] args) {
        ______ obj = new ______();
        obj.m();
    }
}
```

Select 1 option

A. XXX and YYY

B. XXX and XXX

C. YYY and YYY

D. YYY and XXX

E. Nothing will make the code compile.

Check Answer

Test 4 (Answered)

01. QID - 2.960 : Using Loop Constructs

What will the following code print when compiled and run?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        A: for(int i = 0; i < 2; i++){
            B: for(int j = 0; j < 2; j++){
                C: for(int k = 0; k < 3; k++){
                    c++;
                    if(k>j) break;
                }
            }
        }
        System.out.println(c);
    }
}
```

Correct Option is : D

~~**A.**~~ 7

~~**B.**~~ 8

~~**C.**~~ 9

D. 10

~~**E.**~~ 11

Explanation:

The point to note here is that a break without any label breaks the innermost outer loop. So in this case, whenever k>j, the C loop breaks.
You should run the program and follow it step by step to understand how it progresses.

Back to Question without Answer

02. QID - 2.1349 : Constructors

Given a class named Test, which of these would be valid definitions for the constructors for the class?

Correct Option is : A

A. `Test(Test b) { }`

> The constructor can take the same type as a parameter.

~~B.~~ `Test Test( ) { }`

> A constructor cannot return anything.

~~C.~~ `private final Test( ) { }`

> A constructor cannot be final, static or abstract.

~~D.~~ `void Test( ) { }`

> A constructor cannot return anything not even void.

~~E.~~ `public static void Test(String args[ ] ) { }`

> A constructor cannot be final, static or abstract.

Back to Question without Answer

03. QID - 2.1327 : Using Loop Constructs

What will be the output when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0; j < i; ++j, i++){
         System.out.println(i + " " + j);
      }
      System.out.println(i + " " + j);
   }
}
```

Correct Option is : B

~~**A.**~~ 0 0 will be printed twice.

B. 0 0 will be printed once.

~~**C.**~~ It will keep on printing 0 0

~~**D.**~~ It will not compile.

~~**E.**~~ It will print 0 0 and then 0 1.

Explanation:

++j and i++ do not matter in this case.

The loop will not execute even once since j is not less than i at the start of the loop so the condition fails and the program will print 0 0 just once.

Back to Question without Answer

04. QID - 2.1119 : Using Operators and Decision Constructs

Given:

```
public class Switcher{

   public static void main(String[] args){
       switch(Integer.parseInt(args[1]))  //1
       {
          case 0 :
             boolean b = false;
             break;

          case 1 :
             b = true; //2
             break;
       }

       if(b) System.out.println(args[2]);
   }
}
```

What will the above program print if compiled and run using the following command line:

```
java Switcher 1 2 3
```

Correct Option is : G

~~**A.**~~ It will print 1

~~**B.**~~ It will print 2

~~**C.**~~ It will print 3

~~D.~~ It will not print anything.

~~E.~~ It will not compile because of //1.

> There is no problem here because Integer.parseInt() returns an int.

~~F.~~ It will not compile because of //2.

> There is no problem here. b is in scope for the rest of the switch block.

G. It will not compile for some other reason.

> It will not compile because of `if(b)` because `b` is declared in the switch block and it is out of scope after the switch block ends. Pay close attention to question text. It may seem to test you on one concept but actually it could be testing something entirely different.

Back to Question without Answer

05. QID - 2.1055 : Working with Methods

Identify valid method declarations.

Correct Option is : C

~~**A.**~~ `public void methodX(int... i, String s);`

> A var-args parameter must always be the last one.

~~**B.**~~ `public void methodX(int... i, String... s);`

> A method can have only one var-args parameter.

C. `public void methodX(int i, int... j);`

~~**D.**~~ `public int... methodX(int i);`

> The ... syntax is not applicable for return type.

Back to Question without Answer

06. QID - 2.1151 : Working with Java Data Types - Variables and Objects

Given that TestClass is a class, how many objects and reference variables are created by the following code?

```
TestClass t1, t2, t3, t4;
t1 = t2 = new TestClass();
t3 = new TestClass();
```

Correct Option is : B

~~**A.**~~ 2 objects, 3 references.

B. 2 objects, 4 references.

two news => two objects. t1, t2, t3, t4 => 4 references.

~~**C.**~~ 3 objects, 2 references.

~~**D.**~~ 2 objects, 2 references.

~~**E.**~~ None of the above.

Explanation:

A declared reference variable exists regardless of whether a reference value (i.e. an object) has been assigned to it or not.

Back to Question without Answer

07. QID - 2.1243 : Working with Java Data Types - Variables and Objects

Which of the following are correct ways to initialize the static variables MAX and CLASS_GUID ?

```
class Widget{
   static int MAX;      //1
   static final String CLASS_GUID;    // 2
   Widget(){
       //3
   }
   Widget(int k){
       //4
   }
}
```

Correct Options are : A B

A. Modify lines //1 and //2 as : `static int MAX = 111; static final String CLASS_GUID = "XYZ123";`

> You can initialize both the variables at declaration itself.

B. Add the following line just after //2 : `static {  MAX = 111; CLASS_GUID = "XYZ123"; }`

> Initializing the static variables in a static block ensures that they are initialized even when no instance of the class is created.

~~**C.**~~ Add the following line just before //1 : `{ MAX = 111; CLASS_GUID = "XYZ123"; }`

> This is not a static initializer and so will not be executed until an instance is created.

~~**D.**~~ Add the following line at //3 as well as //4 : `MAX = 111; CLASS_GUID = "XYZ123";`

> This works for non-static final fields but not for static final fields.

~~**E.**~~ Only option 3 is valid.

Explanation:

The rules are:
1. static variables can be left without initializing. (They will get default values).
2. final variables must be explicitly initialized.
Now, here `CLASS_GUID` is a 'static final' variable and not just final or static. As static fields can be accessed even without creating an instance of the class, it is entirely possible that this field can be accessed before even a single instance is created. In this case, no constructor or non-static initializer had ever been called. And so, the field (as it is final and so must be initialized explicitly) remains uninitialized. This causes the compiler to complain.

Had `CLASS_GUID` been just a final variable, option 4 would work but as it is also static, it cannot wait till the constructor executes to be initialized.

Back to Question without Answer

08. QID - 2.984 : Handling Exceptions

Following is a supposedly robust method to parse an input for a float :

```
public float parseFloat(String s){
   float f = 0.0f;
   try{
      f = Float.valueOf(s).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      System.out.println("Invalid input " + s);
      f = Float.NaN ;
      return f;
   }
   finally { System.out.println("finally");  }
   return f ;
}
```

Which of the following statements about the above method are true??

Correct Option is : E

~~**A.**~~ If input is `"0.1"` then it will return `0.1` and print `finally`.

~~**B.**~~ If input is `"0x.1"` then it will return `Float.Nan` and print `Invalid Input 0x.1` and `finally`.

~~**C.**~~ If input is `"1"` then it will return `1.0` and print `finally`.

~~**D.**~~ If input is `"0x1"` then it will return `0.0` and print `Invalid Input 0x1` and `finally`.

E. The code will not compile.

Note that the return statement after finally block is unreachable. Otherwise, if this line were not there, choices 1, 2, 3 are valid.

Back to Question without Answer

09. QID - 2.1319 : Working with Java Data Types - Variables and Objects

Given the following code snippet:

```
int rate = 10;
int t = 5;
XXX amount = 1000.0;
for(int i=0; i<t; t++){
   amount = amount*(1 - rate/100);
}
```

What can XXX be?

Correct Option is : C

~~**A.** int~~

~~**B.** long~~

C. only double

~~**D.** double or float~~

~~**E.** float~~

Explanation:

There is no need for analyzing the whole code. `XXX amount = 1000.0;` will be valid only if XXX is double.

Note that the options do not include wrapper classes. Otherwise, Double is also valid

because of auto boxing.

Back to Question without Answer

10. QID - 2.1178 : Working with Methods

Which one of the following class definitions is/are a legal definition of a class that cannot be instantiated?

```
class Automobile{
   abstract void honk();  //(1)
}

abstract class Automobile{
   void honk();   //(2)
}

abstract class Automobile{
   void honk(){};   //(3)
}

abstract class Automobile{
   abstract void honk(){}   //(4)
}

abstract class Automobile{
   abstract void honk();   //(5)
}
```

Correct Options are : C E

~~**A.**~~ 1

> It will not compile as one of its method is abstract but the class itself is not abstract.

~~**B.**~~ 2

> It will not compile as the method doesn't have the body and also is not declared abstract.

C. 3

> This is a valid abstract class although it doesn't have any abstract method.

~~**D.** 4~~

> An abstract method cannot have a method body. {} constitutes a valid method body.

E. 5

> This is a valid abstract class

Explanation:

Here are some points to remember:

A class is uninstantiable if the class is declared `abstract`.
If a method has been declared as `abstract`, it cannot provide an implementation i.e. a method body even if empty (and the class containing that method must be declared `abstract`).
If a method is not declared `abstract`, it must provide a method body (the class can be `abstract` but not necessarily so).
If any method in a class is declared `abstract`, then the whole class must be declared `abstract`.

Back to Question without Answer

11. QID - 2.1267 : Using Operators and Decision Constructs

Which of the following implementations of a `max()` method will correctly return the largest value?

Correct Option is : D

~~A.~~
```
int max(int x, int y){
    return(  if(x > y){ x; } else{ y; }  );
}
```

The if statement does not return any value so it can not be used the way it is used in (1).

~~B.~~
```
int max(int x, int y){
    return( if(x > y){ return x; }  else{ return y; } );
}
```

It would work if the first return and the corresponding brackets is removed.

~~C.~~
```
int max(int x, int y){
    switch(x < y){
       case true:
              return y;
       default :
              return x;
    };
}
```

Neither the switch expression nor the case labels can be of type boolean.

D.
```
int max(int x, int y){
     if (x > y)  return x;
     return y;
}
```

E. None of the above.

Back to Question without Answer

12. QID - 2.1322 : Encapsulation

What is meant by "encapsulation" ?

Correct Option is : C

~~**A.**~~ There is no way to access member variable.

~~**B.**~~ There are no member variables.

C. Member fields are declared private but public accessor/mutator methods are provided to access and change their values.

~~**D.**~~ Data fields are declared public and accessor methods are provided to access and change their values.

~~**E.**~~ None of the above.

Explanation:

Encapsulation is one of the 4 fundamentals of OOP (Object Oriented Programming).

Encapsulation means that the internal representation of an object is generally hidden from view outside of the object's definition. Typically, only the object's own methods can directly inspect or manipulate its fields. Some languages like Smalltalk and Ruby only allow access via object methods, but most others (e.g. C++ or Java) offer the programmer a degree of control over what is hidden, typically via keywords like public and private.

Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state. A benefit of encapsulation is that it can reduce system complexity, and thus increases robustness, by allowing the developer to limit the interdependencies between software components.

Back to Question without Answer

13. QID - 2.1086 : Working with Inheritance

What will be the result of compiling and running the following code?

```
class Base{
   public Object getValue(){ return new Object(); } //1
}

class Base2 extends Base{
   public String getValue(){ return "hello"; } //2
}

public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Correct Option is : B

~~**A.**~~ It will print the hash code of the object.

B. It will print `hello`.

> Covariant returns are allowed since Java 1.5, which means that an overriding method can change the return type to a subclass of the return type declared in the overridden method. But remember than covarient returns does not apply to primitives.

~~**C.**~~ Compile time error at `//1`.

~~**D.**~~ Compile time error at `//2`.

~~**E.**~~ Compile time error at `//3`.

Explanation:

Observe that at run time `b` points to an object of class `Base2`. Further, `Base2` overrides `getValue()`. Therefore, `Base2`'s `getValue()` will be invoked and it will return `hello`.

Back to Question without Answer

14. QID - 2.1077 : Working with Java Data Types - Variables and Objects

What will the following program print?

```
public class TestClass{
  static boolean b;
  static int[] ia = new int[1];
  static char ch;
  static boolean[] ba = new boolean[1];
  public static void main(String args[]) throws Exception{
    boolean x = false;
    if( b ){
      x = ( ch == ia[ch]);
    }
    else x = ( ba[ch] = b );
    System.out.println(x+" "+ba[ch]);
  }
}
```

Correct Option is : D

~~**A.**~~ true true

~~**B.**~~ true false

~~**C.**~~ false true

D. false false

~~**E.**~~ It will not compile.

Explanation:

This question tests your knowledge on the default values of uninitialized primitives and object references. `booleans` are initialized to `false`, numeric types to 0 and object references to `null`. Elements of arrays are initialized to the default values of their types. So, elements of a `boolean` array are initialized to `false`. int, char, float to 0 and Objects to null.

In this case, `b` is `false`. So the else part of `if(b)` is executed.
`ch` is a numeric type to its value is `0`. `ba[0] = false` assigns `false` to `ba[0]` and returns `false` which is assigned to `x`.
Finally, `x` and `ba[0]` are printed as `false false`.

Back to Question without Answer

15. QID - 2.1359 : Working with Inheritance

Consider :

```
class A {  public void perform_work(){}  }
class B extends A {  public void perform_work(){}  }
class C extends B {  public void perform_work(){}  }
```

How can you let `perform_work()` method of A to be called from an instance method in C?

Correct Option is : E

~~**A.**~~ `( (A) this ).perform_work( );`

~~**B.**~~ `super.perform_work( );`

~~**C.**~~ `super.super.perform_work( );`

~~**D.**~~ `this.super.perform_work( );`

E. It is not possible.

Explanation:

The method in C needs to call a method in a superclass two levels up. But `super` is a keyword and not an attribute so `super.super.perform_work( )` strategy will not work. There is no way to go more than one level up for methods.
Remember that this problem doesn't occur for instance variables because variable are never overridden. They are shadowed. So to access any of the super class's variable,

you can unshadow it using a cast. For example, `((A) c).data;` This will give you the data variable defined in A even if it is shadowed in B as well as in C.

Back to Question without Answer

16. QID - 2.1336 : Working with Java Data Types - String, StringBuilder

Which of these methods are not a part of the String class?

Correct Option is : E

~~**A.**~~ `trim( )`

~~**B.**~~ `length( )`

~~**C.**~~ `concat(String)`

~~**D.**~~ `hashCode( )`

E. `reverse( )`

The String class has no `reverse( )` method but `StringBuffer` (and `StringBuilder`) do have this method.

Back to Question without Answer

17. QID - 2.1277 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Options are : A C

A. `System.out.println(1 + 2 + "3");` would print 33.

> operator + is left associative so evaluation of (1 + 2 + "3") is as follows: (1 + 2) + "3" -> 3 + "3" -> "33".

~~B.~~ `System.out.println("1" + 2 + 3);` would print 15.

> evaluation of ("1" + 2 + 3) is as follows: ("1" + 2) + 3 -> "12" + 3 -> "123".

C. `System.out.println(4 + 1.0f);` would print 5.0

> (4 + 1.0f) evaluates as 4.0f + 1.0f ->5.0f -> 5.0

~~D.~~ `System.out.println(5/4);` would print 1.25

> (5/4) performs integer division because both 5 and 4 are integers, resulting in the value 1.

~~E.~~ `System.out.println('a' + 1 );` would print b.

> Both operands in the expression ('a' + 1) will be promoted to int => 97 + 1 = 98

Explanation:

All operands of type byte, char or short are promoted AT LEAST to an int before

performing mathematical operations. If one of the operands is larger than an int then the other one is promoted to the same type.
Note that `System.out.println((float)5/4);` will print `1.25`. If you remove the explicit cast (float), it will print 1.

Back to Question without Answer

18. QID - 2.1306 : Working with Inheritance

Which of these statements concerning interfaces are true?

Correct Options are : A C

A. An interface may extend an interface.

> Unlike a class, an interface can extend from multiple interfaces.

~~**B.**~~ An interface may extend a class and may implement an interface.

> interface does not implement anything. It can extend another interface but not class.

C. A class can implement an interface and extend a class.

~~**D.**~~ A class can extend an interface and can implement a class.

~~**E.**~~ An interface can only be implemented and cannot be extended.

> It can be extended by another interface.

Explanation:

The keyword implements is used when a class inherits method prototypes from an interface. The keyword extends is used when an interface inherits from another interface, or a class inherits from another class.

Back to Question without Answer

19. QID - 2.1362 : Working with Methods

Which of the following statements are true?

Correct Options are : B E

~~**A.**~~ private keyword can never be applied to a class.

> private, protected and public can be applied to an Inner class.

B. synchronized keyword can never be applied to a class.

~~**C.**~~ synchronized keyword may be applied to a non-primitive variable.

> It can only be applied to a method or a block.

~~**D.**~~ final keyword can never be applied to a class.

> It can be applied to class, variable and methods.

E. A final variable can be shadowed in a subclass.

Explanation:

final keyword when applied to a class means the class cannot be subclassed, when applied to a method means the method cannot be overridden (it can be overloaded though) and when applied to a variable means that the variable is a constant.

Back to Question without Answer

20. QID - 2.865 : Working with Java Data Types - Variables and Objects

What will the following program print when compiled and run?

```
class Data {
    private int x = 0;
    private String y = "Y";
    public Data(int k){
        this.x = k;
    }
    public Data(String k){
        this.y = k;
    }
    public void showMe(){
        System.out.println(x+y);
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        new Data(10).showMe();
        new Data("Z").showMe();
    }
}
```

Correct Option is : B

~~**A.**~~ 0Z
10Y

B. 10Y
0Z

You are creating two different Data objects in the code. The first one uses a constructor that takes an integer and the second one uses a constructor that takes a String.

Thus, when you call showMe on the first object, it prints 10Y because "Y" is the default value of y as per the given code. When you call showMe on the second object, it prints 0Z because 0 is the default value of x as per the given code.

~~**C.**~~ It will not compile.

~~**D.**~~ It will throws an exception at run time.

Explanation:

Note that + operator is overloaded for String. So if you have a String as any operand for +, a new combined String will be created by concatenating the values of both the operands. Therefore, x+y will result in a String that concatenates integer x and String y.

Back to Question without Answer

21. QID - 2.1234 : Java Basics

Consider the following code:

```
public abstract class TestClass{
    public abstract void m1();
    public abstract void m2(){
        System.out.println("hello");
    }
}
```

Which of the following corrections can be applied to the above code (independently) so that it compiles without any error?

Correct Options are : A C

A. Replace the method body of m2() with a ; (semi-colon).

~~**B.**~~ Replace the ; at the end of m1() with a method body.

C. Remove abstract from m2().

> A method that has a body cannot be abstract. In other words, an abstract method cannot have a body. So either remove the method body (as in m1()) or remove abstract keyword.

~~**D.**~~ Remove abstract from the class declaration.

Back to Question without Answer

22. QID - 2.974 : Working with Methods

What will be the contents of s1 and s2 at the time of the println statement in the main method of the following program?

```
import java.util.*;
public class TestClass{
   public static void main(String args[]){
      Stack s1 = new Stack ();
      Stack s2 = new Stack ();
      processStacks (s1,s2);
      System.out.println (s1 + "    "+ s2);
   }
   public static void processStacks(Stack x1, Stack x2){
      x1.push (new Integer ("100")); //assume that the method push a
      x2 = x1;
   }
 }
```

Correct Option is : B

~~**A.**~~ [100] [100]

B. [100] []

~~**C.**~~ [] [100]

~~**D.**~~ [] []

Explanation:

. Primitives are always passed by value.

. Object "references" are passed by value. So it looks like the object is passed by reference but actually it is the value of the reference that is passed.

An example:

```
    Object o1 = new Object(); //Let us say, the object is stored at
memory location 15000.
        //Since o1 actually stores the address of the memory
location where the object is stored, it contains 15000.
```

Now, when you call someMethod(o1); the value 15000 is passed to the method.

Inside the method someMethod():

```
    someMethod( Object localVar) {
            /*localVar now contains 15000, which means it also
points to the same memory location where the object is stored.
            Therefore, when you call a method on localVar, it will
be executed on the same object.
            However, when you change the value of localVar itself,
for example if you do localVar=null,
            it then it starts pointing to a different memory
location. But the original variable o1 still
            contains 15000 so it still points to the same object. */
        }
```

If you need even more detailed explanation, please check
http://www.javaranch.com/campfire/StoryPassBy.jsp

This is what happens in this question.

You created two objects in main method:

```
s1 ------------> [ EMPTY ] STACK 1 OBJECT
```

s1 actually contains 15000 (say)

```
s2 ------------> [ EMPTY ] STACK 2 OBJECT
```

s2 actually contains 25000 (say)

inside the method `processStacks()` :

Step 1:

```
s1 ----> [ EMPTY ] STACK 1 OBJECT <----x1 Local variable
```

s1 and x1 both contain 15000 (say)

```
s2 ----> [ EMPTY ] STACK 2 OBJECT <----x2 Local variable
```

s2 and x2 both contain 25000 (say)

Step 2;

```
s1 -----> [ 100 ] STACK 1 OBJECT <----x1 Local variable
```

Because x1 is referring to the same memory location.

```
s2 -----> [ EMPTY ] STACK 2 OBJECT <---x2 Local variable
```

Step 3: After doing `x2 = x1`

```
s1 ---> [ 100 ] STACK 1 OBJECT <---- x1 and x2 Local variables
```

s1 and x1 both contain 15000 (say) and x2 now also contains 15000.

```
s2 ------------> [ EMPTY ] STACK 2 OBJECT
```

But s2 still contains 25000.

Note that it is the local variable x2 that is pointing to the same object as x1, which is s1 stack object. The original s2 (of the main method) is still pointing to the same object which is empty.

So when you come back to the main method, you print s1 (which has now 100) and s2 (which is still empty).

Back to Question without Answer

23. QID - 2.1204 : Working with Methods

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
  public static void main(String args[ ] ){
    Object a, b, c ;
    a = new String("A");
    b = new String("B");
    c = a;
    a = b;
    System.out.println(""+c);
  }
}
```

Correct Option is : B

~~**A.**~~ The program will print `java.lang.String@XXX`, where XXX is the memory location of the object a.

B. The program will print A

~~**C.**~~ The program will print B

~~**D.**~~ The program will not compile as a,b and c are of type Object.

> String also IS an Object ! You can always assign a subclass object to a super class reference without a cast.

~~**E.**~~ The program will print `java.lang.String@XXX`, where XXX is the hash code of the object a.

Explanation:

The variables a, b and c contain references to actual objects. Assigning to a reference only changes the reference value, and not the object pointed to by the reference. So, when `c = a` is executed c starts pointing to the string object containing A. and when `a = b` is executed, a starts pointing to the string object containing B but the object containing A still remains same and c still points to it. So the program prints A and not B.

The Object class's toString generates a string using: `getClass().getName() + '@' + Integer.toHexString(hashCode())`
But in this case, String class overrides the `toString()` method that returns just the actual string value.

Back to Question without Answer

24. QID - 2.892 : Working with Java Data Types - Variables and Objects

Which of the following are valid classes?

Correct Option is : A

A.
```
public class ImaginaryNumber extends Number {
}
```

> `Number` is not a final class so you can extend it.

~~**B.**~~
```
public class ThreeWayBoolean extends Boolean {
}
```

~~**C.**~~
```
public class NewSystem extends System {
}
```

~~**D.**~~
```
public class ReverseString extends String {
}
```

Explanation:

`String, StringBuilder, and StringBuffer` - all are final classes.

1. Remember that wrapper classes for primitives (`java.lang.Boolean, java.lang.Integer, java.lang.Long, java.lang.Short` etc.) are also final and so they cannot be extended.

2. `java.lang.Number`, however, is not final. `Integer, Long, Double` etc. extend Number.

3. `java.lang.System` is final as well.

Back to Question without Answer

25. QID - 2.906 : Working with Methods

Given:

```
class Triangle{
    public int base;
    public int height;
    private final double ANGLE;

    public  void setAngle(double a){  ANGLE = a;  }

    public static void main(String[] args) {
        Triangle t = new Triangle();
        t.setAngle(90);
    }
}
```

Correct Option is : D

~~**A.**~~ the value of `ANGLE` will not be set to `90` by the `setAngle` method.

~~**B.**~~ An exception will be thrown at run time.

~~**C.**~~ The code will work as expected setting the value of `ANGLE` to `90`.

D. The code will not compile.

Explanation:

The given code has two problems:
1. If you declare a field to be `final`, it must be explicitly initialized by the time the creation of an object of the class is complete. So you can either initialize it

immediately:

```
private final double ANGLE = 0;
```

or you can initialize it in the constructor or an instance block.

2. Since `ANGLE` is `final`, you can't change its value once it is set. Therefore the `setAngle` method will also not compile.

Back to Question without Answer

26. QID - 2.1294 : Working with Methods

What will the code shown below print when run?

```
public class TestClass{
    static class Wrapper{
        int w = 10;
    }

    static Wrapper changeWrapper(Wrapper w){
        w = new Wrapper();
        w.w += 9;
        return w;
    }


    public static void main(String[] args){
        Wrapper w = new Wrapper();
        w.w = 20;
        changeWrapper(w);
        w.w += 30;
        System.out.println(w.w);
        w = changeWrapper(w);
        System.out.println(w.w);
     }
}
```

Correct Options are : B F

~~**A.** 9~~

B. 19

~~**C.** 30~~

~~**D.** 20~~

~~**E.** 29~~

F. 50

Explanation:

Remember that when you pass an object in a method, only its reference is passed by value. So when `changeWrapper()` does `w = new Wrapper();` and then `w.w +=9;` it does not affect the original wrapper object that was passed to this method. Therefore, it prints `50`.

Calling `w = changeWrapper(w);` replaces the original Wrapper object with the one created in the `changeWrapper(w);` method. Therefore, in the second print statement, it prints `19`.

Back to Question without Answer

27. QID - 2.1020 : Java Basics

What does the zeroth element of the string array passed to the standard main method contain?

Correct Option is : D

~~**A.**~~ The name of the class.

~~**B.**~~ The string "java".

~~**C.**~~ The number of arguments.

D. The first argument of the argument list, if present.

~~**E.**~~ None of the above.

Explanation:

Note that if no argument is passed the args parameter is NOT null but a valid non-null String array of length zero.

Back to Question without Answer

28. QID - 2.965 : Using Operators and Decision Constructs

Consider the following program:

```
public class TestClass{
   public static void main(String[] args)  {     calculate(2);     }
   public static void calculate(int x){
      String val;
      switch(x){
         case 2:
         default:
         val = "def";
      }
      System.out.println(val);
   }
}
```

What will happen if you try to compile and run the program?

Correct Options are : B D

~~**A.**~~ It will not compile saying that variable `val` may not have been initialized..

B. It will compile and print `def`

~~**C.**~~ As such it will not compile but it will compile if `calculate(2);` is replaced by `calculate(3);`

D. It will compile for any int values in `calculate(...);`

Explanation:

When you try to access a local variable, the compiler makes sure that it is initialized in all the cases. If it finds that there is a case in which it may not be initialized then it flags an error. For example:

```
int i;
if( somecondition) i = 20;
int k = i;
```

Here, if some condition returns `false`, then i remains uninitialized hence the compiler flags an error.
In the given question:
As there is no `break` after `case 2`, `val` will always be initialized in the `switch` block. So it will compile and run fine.
Note that it will not compile if `break` is placed after `case 2` because the compiler will figure out that in certain cases `val` may be left uninitialized.

Back to Question without Answer

29. QID - 2.1110 : Java Basics

Which of these statements are true?

Correct Options are : B C

~~**A.**~~ A static method can call other non-static methods in the same class by using the 'this' keyword.

> 'this' reference is not available within a static method.

B. A class may contain both static and non-static variables and both static and non-static methods.

C. Each object of a class has its own copy of each non-static member variable.

~~**D.**~~ Instance methods may access local variables of static methods.

> local variables can only be accessed in the method they are defined. So you cannot access them anywhere outside that method.

~~**E.**~~ All methods in a class are implicitly passed a 'this' parameter when called.

> All non-static/instance methods in a class are implicitly passed a 'this' parameter when called.

Explanation:

The keyword 'this' can only be used within non-static methods. static methods cannot access non static fields or methods.

Note : you can't do something like
this = new Object();

Back to Question without Answer

30. QID - 2.1081 : Using Operators and Decision Constructs

Which of the following code snippets will print exactly 10?

```
1.  Object t = new Integer(106);
    int k = ((Integer) t).intValue()/10;
    System.out.println(k);

2.  System.out.println(100/9.9);

3.  System.out.println(100/10.0);

4.  System.out.println(100/10);

5.  System.out.println(3 + 100/10*2-13);
```

Correct Options are : A D E

A. 1

~~B.~~ 2

> Since one of the operands (9.9) is a double, it wil perform a real division and will print 10.1010101010101

~~C.~~ 3

> Since one of the operands (10.0) is a double, it will perform a real division and will print 10.0

D. 4

E. 5

Explanation:

1. `int k = ((Integer) t).intValue()/10;`
Since both the operands of / are ints, it is a integer division. This means the resulting value is truncated (and not rounded). Therefore, the above statement will print 10 and not 11.
5. `3 + 100/10*2-13` will be parsed as: `3 + (100/10)*2-13`. This is because the precedence of / and * is same (and is higher than + and -) and since the expression is evaluated from left to right, the operands are grouped on first come first served basis. [This is not the right terminology but you will be able to answer the questions if you remember this rule.]

Back to Question without Answer

31. QID - 2.930 : Using Operators and Decision Constructs

Expression `(s instanceof java.util.Date)` will return false if 's' was declared as a variable of class `java.lang.String`.

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

It will not even compile because the compiler knows that 's' (which is declared as of class `String`) can NEVER refer to an object of class `java.util.Date`. So, it will not accept this code.
Had 's' been declared as a variable of type `Object`, this code would have compiled because compiler sees that at run time it is possible for s to refer to an object of class `Date`.

Back to Question without Answer

32. QID - 2.1129 : Working with Inheritance

Consider the following class:

```
public class PortConnector{
   public PortConnector(int port) throws IOException{
   ...lot of valid code.
   }
   ...other valid code.
}
```

You want to write another class `CleanConnector` that extends from `PortConnector`. Which of the following statements should hold true for `CleanConnector` class?

Correct Option is : F

~~**A.**~~ It is not possible to define `CleanConnector` that does not throw `IOException` at instantiation.

> It is possible. You can also throw a superclass of `IOException` from the `CleanConnector`'s constructor. For example, the following is valid:
>
> ```
> class CleanConnector extends PortConnector {
>    public CleanConnector(int port) throws Exception {
>          super(port);
>    }
> }
> ```

~~**B.**~~ `PortConnector` class itself is not valid because you cannot throw any exception from a constructor.

> A constructor is free to throw any exception.

~~**C.**~~ `CleanConnector`'s constructor cannot throw any exception other than `IOException`.

It can throw any exception but it must also throw `IOException` (or its super class). So the following is valid:

```
class CleanConnector extends PortConnector {
   public CleanConnector(int port) throws IOException,
FileNotFoundException, SomeOtherCheckedException {
         super(port);
   }
}
```

D. `CleanConnector`'s constructor cannot throw any exception other than subclass of `IOException`.

As described above, it can throw any exception but it must throw `IOException` (or its superclass) as well.

E. `CleanConnector`'s constructor cannot throw any exception other than superclass of `IOException`.

As described above, it can throw any exception but it must throw `IOException` (or its superclass) as well.

F. None of these.

Observe that the rule for overriding a method is opposite to the rule for constructors. An overriding method cannot throw a superclass exception, while a constructor of a subclass cannot throw subclass exception (Assuming that the same exception or its super class is not present in the subclass constructor's throws clause). For example:

```
class A{
  public A() throws IOException{ }
  void m() throws IOException{ }
}
```

```
  class B extends A{
    //IOException is valid here, but FileNotFoundException is
invalid
    public B() throws IOException{ }

    //FileNotFoundException is valid here, but Exception is
invalid
    void m() throws FileNotFoundException{ }
}
```

(Note: `FileNotFoundException` is a subclass of `IOException`, which is a subclass of `Exception`)

If the subclass constructor's throws clause includes the same exception or its superclass, then it can throw any other exception as well.

Explanation:

As PortConnector has only one constructor, there is only one way to instantiate it. Now, to instantiate any subclass of PortConnector, the subclass's constructor should call super(int). But that throws throws IOException. And so it (or its super class) must be defined in the throws clause of subclass's constructor. Note that you cannot do something like:

```
public CleanConnector(){
   try{ super(); }catch(Exception e){} //WRONG : call to super must b
}
```

Remember: Constructor must declare all the checked exceptions declared in the base constructor (or the super classes of the checked exceptions). They may add other exception. This behavior is exactly opposite from that of methods. The overriding method cannot throw any exception other than overridden method. It may throw subclasses of those exceptions.

Back to Question without Answer

33. QID - 2.1188 : Working with Methods

What should be the return type of the following method?

```
public RETURNTYPE methodX( byte by){
    double d = 10.0;
    return (long) by/d*3;
}
```

Correct Option is : C

~~**A.**~~ int

~~**B.**~~ long

C. double

~~**D.**~~ float

~~**E.**~~ byte

Explanation:

Note that the cast (long) applies to '`by`' not to the whole expression.
`( (long) by ) / d * 3;`
Now, division operation on long gives you a `double`. So the return type should be `double`.

Back to Question without Answer

34. QID - 2.1192 : Working with Java Data Types - String, StringBuilder

What will be the output of the following lines ?

```
System.out.println("" +5 + 6);    //1
System.out.println(5 + "" +6);    // 2
System.out.println(5 + 6 +"");    // 3
System.out.println(5 + 6);        // 4
```

Correct Option is : A

A. 56, 56, 11, 11

~~**B.**~~ 11, 56, 11, 11

~~**C.**~~ 56, 56, 56, 11

~~**D.**~~ 56, 56, 56, 56

~~**E.**~~ 56, 56, 11, 56

Explanation:

In line 1, `"" + 5 + 6 => "5"+6 => "56"`
In line 2, `5 + "" +6  => "5"+6 => "56"`
In line 3, `5 + 6 +"" => 11+"" => "11"`
In line 4, `5 + 6 => 11 => "11"`

Back to Question without Answer

35. QID - 2.1047 : Working with Inheritance

An overriding method must have a same parameter list and the same return type as that of the overridden method.

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

This would have been true prior to Java 1.5. But from Java 1.5, an overriding method is allowed to change the return type to any subclass of the original return type, also known as covariant return type. This does not apply to primitives, in which case, the return type of the overriding method must match exactly to the return type of the overridden method.

Back to Question without Answer

36. QID - 2.1355 : Constructors

Given the following code, which of these constructors can be added to class B without causing a compile time error?

```
class A{
   int i;
   public A(int x) { this.i = x; }
}
class B extends A{
   int j;
   public B(int x, int y) { super(x);  this.j = y; }
}
```

Correct Options are : C E

~~**A.**~~ `B( ) { }`

~~**B.**~~ `B(int y ) { j = y; }`

C. `B(int y ) { super(y*2 ); j = y; }`

~~**D.**~~ `B(int y ) { i = y; j = y*2; }`

E. `B(int z ) { this(z, z); }`

Explanation:

1.To construct an instance of a sub class, its super class needs need to be constructed first. Since an instance can only be created via a constructor, some constructor of the

super class has to be invoked.
Either you explicitly call it or the compiler will add `super()` (i.e. no args constructor) as the first line of the sub class constructor.

Now, if the super class (here, A) does not have a no args constructor, the call `super()` will fail. Hence, choices `B( ) { }`, `B(int y ) { j = y; }` and `B(int y ) { i = y; j = y*2; }` are not valid and choice `B(int y ) { super(y*2 ); j = y; }` is valid because it explicitly calls `super( int )` which is available in A.

2.Instead of calling `super(...)` you can also call another constructor of the base class in the first line (as given in choice `B(int z ) { this(z, z); }`). Here, `this(int, int)` is called in the first line which in turn calls `super(int)`. So the super class A is correctly instantiated.

Back to Question without Answer

37. QID - 2.914 : Using Loop Constructs

Consider the following code:

```
public static void main(String[] args) {
   int[] values = { 10, 30, 50 };
        for( int val : values ){
            int x = 0;
            while(x<values.length){
             System.out.println(x+" "+val);
              x++;
            }
        }
    }
```

How many times is 2 printed out?

Correct Option is : D

~~**A.**~~ 0

~~**B.**~~ 1

~~**C.**~~ 2

D. 3

Explanation:

This is a simple while loop nested inside a for loop. The `for` loop loops three times - once for each value in `values` array.
Since, `values.length` is 3, `x` is incremented two times for each for loop iteration

before the condition `x<values.length` returns `false`.

Therefore, it prints:

```
0 10
1 10
2 10
0 30
1 30
2 30
0 50
1 50
2 50
```

Back to Question without Answer

38. QID - 2.989 : Working with Java Data Types - String, StringBuilder

Consider the following code:

```
public class Logger{
    private StringBuilder sb = new StringBuilder();

    public void logMsg(String location, String message){
    sb.append(location);
    sb.append("-");
    sb.append(message);
    }

    public void dumpLog(){
    System.out.println(sb.toString());
    //Empty the contents of sb here
    }

}
```

Which of the following options will empty the contents of the StringBuilder referred to by variable sb in method dumpLog()?

Correct Option is : A

A. `sb.delete(0, sb.length());`

~~**B.**~~ `sb.clear();`

~~**C.**~~ `sb.empty();`

~~**D.**~~ `sb.removeAll();`

~~**E.**~~ `sb.deleteAll();`

Explanation:

`public StringBuilder delete(int start, int end)`
Removes the characters in a substring of this sequence. The substring begins at the specified start and extends to the character at index end - 1 or to the end of the sequence if no such character exists. If start is equal to end, no changes are made.
Parameters:
start - The beginning index, inclusive.
end - The ending index, exclusive.
Returns:
This object.
Throws:
StringIndexOutOfBoundsException - if start is negative, greater than length(), or greater than end.

Back to Question without Answer

39. QID - 2.1212 : Using Operators and Decision Constructs

Which of the following expressions will evaluate to true if preceded by the following code?

```
String a = "java";
    char[] b = { 'j', 'a', 'v', 'a' };
    String c = new String(b);
    String d = a;
```

Correct Options are : A C D

A. `(a == d)`

~~B.~~ `(b == d)`

b and d can not even be compared because they are of different types.

C. `(a == "java")`

D. `a.equals(c)`

Note that a == c will be false because doing 'new' creates an entirely new object.

Back to Question without Answer

40. QID - 2.1312 : Working with Methods

Which of the following code fragments are valid method declarations?

Correct Option is : B

~~**A.**~~ void method1{ }

> It does not have () after method1.

B. void method2( ) { }

~~**C.**~~ void method3(void){ }

> The keyword void is not a valid type for a parameter.

~~**D.**~~ method4{ }

> Methods must specify a return type and '( )'. If the method does not want to return a value, it should specify void.

~~**E.**~~ method5(void){ }

> If the method does not take any parameter, it should have empty brackets instead of void.

Explanation:

A valid method declaration MUST specify a return type, all other things are optional.

Back to Question without Answer

41. QID - 2.1076 : Creating and Using Arrays

What would be the result of compiling and running the following program?

```
class SomeClass{
   public static void main(String args[]){
      int size = 10;
      int[] arr = new int[size];
      for (int i = 0 ; i < size ; ++i) System.out.println(arr[i]);
   }
}
```

Correct Option is : E

~~**A.**~~ The code will fail to compile, because the `int[]` array declaration is incorrect.

~~**B.**~~ The program will compile, but will throw an `IndexArrayOutOfBoundsException` when run.

~~**C.**~~ The program will compile and run without error, and will print nothing.

~~**D.**~~ The program will compile and run without error and will print `null` ten times.

> Here, all the array elements are initialized to have 0.

E. The program will compile and run without error and will print `0` ten times.

> It correctly will declare and initialize an array of length `10` containing `int` values initialized to have `0`.

Explanation:

Elements of Arrays of primitive types are initialized to their default value (i.e. 0 for integral types, 0.0 for float/double and false for boolean)
Elements of Arrays of non-primitive types are initialized to null.

Back to Question without Answer

42. QID - 2.1195 : Working with Inheritance

Which letters will be printed when the following program is run?

```
public class TestClass{
  public static void main(String args[]){
    B b = new C();
    A a = b;
    if (a instanceof A) System.out.println("A");
    if (a instanceof B) System.out.println("B");
    if (a instanceof C) System.out.println("C");
    if (a instanceof D) System.out.println("D");
  }
}
class A { }
class B extends A { }
class C extends B { }
class D extends C { }
```

Correct Options are : A B C

A. A will be printed.

B. B will be printed.

C. C will be printed.

~~**D.**~~ D will be printed.

Explanation:

The program will print A, B and C when run. The object denoted by reference a is of

type C. The object is also an instance of A and B, since C is a subclass of B and B is a subclass of A. The object is not an instance of D.

Back to Question without Answer

43. QID - 2.1304 : Working with Java Data Types - String, StringBuilder

Which of these are not part of the StringBuilder class?

Correct Option is : A

A. `trim( )`

> This method is in String class.

~~**B.**~~ `ensureCapacity(int )`

> Ensures that the capacity of the buffer is at least equal to the specified minimum.

~~**C.**~~ `append(boolean)`

> It has all sorts of overloaded append methods !!!

~~**D.**~~ `reverse( )`

~~**E.**~~ `setLength(int)`

> Sets the length of this String buffer. This string buffer is altered to represent a new character sequence whose length is specified by the argument. For every nonnegative index k less than newLength, the character at index k in the new character sequence is the same as the character at index k in the old sequence if k is less than the length of the old character sequence; otherwise, it is the null character '' (\u0000). In other words, if the newLength argument is less than the current length of the string buffer, the string buffer is truncated to contain exactly the number of characters given by the newLength argument.
> If the newLength argument is greater than or equal to the current length, sufficient

null characters ('\u0000') are appended to the string buffer so that length becomes the newLength argument.
The newLength argument must be greater than or equal to 0.
Parameters:
newLength - the new length of the buffer.
Throws:
IndexOutOfBoundsException - if the newLength argument is negative.

Back to Question without Answer

44. QID - 2.1142 : Java Basics

Which of the given options should be inserted at line 1 so that the following code can compile without any errors?

```
package objective1;
// 1
public class StaticImports{

    public StaticImports(){
    out.println(MAX_VALUE);
    }

}
```

(Assume that java.lang.Integer has a static field named MAX_VALUE)

Correct Options are : A D

A. `import static java.lang.Integer.*;`

~~B.~~ `static import java.lang.System.out;`

~~C.~~ `static import Integer.MAX_VALUE;`

D. `import static java.lang.System.*;`

~~E.~~ `static import java.lang.System.*;`

Explanation:

The order of keywords for a static import must be "`import static ...` ".
You can either import all the static member using `import static java.lang.Integer.*` or one specific member using `import static java.lang.Integer.MAX_VALUE;`
You must specify the full package name of the class that you are importing (just like the regular import statement). So, `import static Integer.*;` is wrong.

Back to Question without Answer

45. QID - 2.1296 : Creating and Using Arrays

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    String str = "111";
    boolean[] bA = new boolean[1];
    if( bA[0] ) str = "222";
    System.out.println(str);
  }
}
```

Correct Option is : A

A. 111

~~**B.**~~ 222

~~**C.**~~ It will not compile as `bA[0]` is uninitialized.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

All the arrays are initialized to contain the default values of their type. This means, `int[] iA = new int[10];` will contain 10 integers with a value of 0. `Object[] oA = new Object[10];` will contain 10 object references pointing to `null`.

`boolean[] bA = new boolean[10]` will contain 10 booleans of value `false`. So, as `bA[0]` is `false`, the if condition fails and `str` remains `111`.

Back to Question without Answer

46. QID - 2.1123 : Working with Inheritance

Consider the contents of following two files:

```
//File A.java
package a;
public class A{
   A(){ }
   public void  print(){ System.out.println("A"); }
}

//File B.java
package b;
import a.*;
public class B extends A{
   B(){ }
   public void  print(){ System.out.println("B"); }
   public static void main(String[] args){
      new B();
   }
}
```

What will be printed when you try to compile and run class B?

Correct Option is : C

~~**A.**~~ It will print A.

~~**B.**~~ It will print B.

C. It will not compile.

Because A() is not accessible in B.

~~**D.**~~ It will compile but will not run.

~~**E.**~~ None of the above.

Explanation:

Note that there is no modifier for A's constructor. So it has default access. This means only classes in package a can use it. Also note that class B is in a different package and is extending from A. In B's constructor the compiler will automatically add super() as the first line. But since A() is not accessible in B, this code will not compile.

Back to Question without Answer

47. QID - 2.1145 : Working with Inheritance

Assume the following declarations:

```
class A{ }
class B extends A{ }
class C extends B{ }

class X{
   B getB(){ return new B(); }
}

class Y extends X{
  // method declaration here
}
```

Which of the following methods can be inserted in class Y?

Correct Options are : B C

A. ~~`public C getB(){ return new B(); }`~~

> Its return type is specified as C, but it is actually returning an object of type B. Since B is NOT a C, this will not compile.

B. `protected B getB(){ return new C(); }`

> Since C is-a B, this is valid. Also, an overriding method can be made less restrictive. protected is less restrictive than 'default' access.

C. `C getB(){ return new C(); }`

> Covariant returns are allowed in Java 1.5, which means that an overriding method can change the return type of the overridden method to a subclass of the

original return type. Here, C is a subclass of B. So this is valid.

D. ~~`A getB(){ return new A(); }`~~

An overriding method cannot return a superclass object of the return type defined in the overridden method. A subclass is allowed in Java 1.5.

Back to Question without Answer

48. QID - 2.1005 : Handling Exceptions

Consider the following code:

```
class A {
    public void doA(int k) throws Exception {  // 0
        for(int i=0; i< 10; i++) {
            if(i == k) throw new Exception("Index of k is "+i); // 1
        }
    }
    public void doB(boolean f) { // 2
        if(f) {
            doA(15); // 3
        }
        else return;
    }
    public static void main(String[] args) { // 4
        A a = new A();
        a.doB(args.length>0); // 5
    }
 }
```

Which of the following statements are correct?

Correct Option is : D

~~**A.**~~ This will compile and run without any errors or exception.

~~**B.**~~ This will compile if `throws Exception` is added at line //2

~~**C.**~~ This will compile if `throws Exception` is added at line //4

D. This will compile if `throws Exception` is added at line //2 as well as //4

~~E.~~ This will compile if line marked // 1 is enclosed in a try - catch block.

> Even if // 1 is enclosed in a try block, the method still has `throws Exception` in its declaration, which will force the caller of this method to either declare `Exception` in its throws clause or put the call within a try block.

Explanation:

Any checked exceptions must be either handled using a try block or the method that generates the exception must declare that it throws that exception.
In this case, doA() declares that it throws `Exception`. `doB()` is calling `doA` but it is not handling the exception generated by `doA()`. So, it must declare that it throws `Exception`. Now, the `main()` method is calling `doB()`, which generates an exception (due to a call to `doA()`). Therefore, `main()` must also either wrap the call to `doB()` in a try block or declare it in its `throws` clause.

The `main(String[] args)` method is the last point in your program where any unhandled checked exception can bubble up to. After that the exception is thrown to the JVM and the JVM kills the thread.

Back to Question without Answer

49. QID - 2.831 : Working with Methods

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
        System.out.println(this.myValue);
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
        System.out.println(this.myValue);
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showOne(100);
        ct.showTwo(200);
    }
}
```

Correct Option is : C

~~**A.**~~ 0 followed by 100.

~~**B.**~~ 100 followed by 100.

C. 0 followed by 200.

~~**D.**~~ 100 followed by 200.

Explanation:

There are a couple of important concepts in this question:

1. Within an instance method, you can access the current object of the same class using '`this`'. Therefore, when you access `this.myValue`, you are accessing the instance member `myValue` of the `ChangeTest` instance.

2. Within the `showOne()` method, there are two variables accessible with the same name myValue. If you declare a local variable (or a method parameter) with the same name as the instance field name, the local variable "shadows" the member field. One is the method parameter and another is the member field of ChangeTest object. Ideally, you should be able to access the member field in the method directly by using the name of the member (in this example, `myValue`). However, because of shadowing, when you use myValue, it refers to the local variable instead of the instance field.

Therefore, when you do : `myValue = myValue;` you are actually assigning the value contained in method parameter myValue to itself. You are not changing the member field myValue. Hence, when you do `System.out.println(this.myValue);` in the next line, it prints 0.

Now, in `showTwo()`, you are assigning the value contained in `myValue` (i.e. 200) to `this.myValue`, which is the instance member. Therefore, in the next line, when you print `this.myValue`, it prints 200.

Back to Question without Answer

50. QID - 2.1113 : Working with Inheritance

Which of the following is a legal return type of a method overriding the given method:

```
public Object  myMethod() {...}
```
(Select the best option.)

Correct Option is : C

~~**A.**~~ Object

~~**B.**~~ String

C. Return type can be any object since all objects can be cast to Object.

> Note that the return type cannot be a primitive such as int or char. It must be a class. So it can be Integer or Character as well.

~~**D.**~~ void

~~**E.**~~ None of the above.

Explanation:

Since the original method is returning Object, the Overriding method can return any object type because all classes in Java ultimately extend from Object. Since 1.5, Java allows covariant return types, which means an overriding method can have its return type as any subclass of the original return type of the overridden method.

Back to Question without Answer

51. QID - 2.1040 : Using Operators and Decision Constructs

What will the following class print ?

```
class InitTest{
   public static void main(String[] args){
      int a = 10;
      int b = 20;
      a += (a = 4);
      b = b + (b = 5);
      System.out.println(a+ ",  "+b);
   }
}
```

Correct Option is : E

~~**A.**~~ It will print 8, 25

~~**B.**~~ It will print 4, 5

~~**C.**~~ It will print 14, 5

~~**D.**~~ It will print 4, 25

E. It will print 14, 25

Explanation:

`a += (a =4)` is same as `a = a + (a=4).`
First, a's value of 10 is kept aside and (a=4) is evaluated. The statement (a=4) assigns 4 to a and the whole statement returns the value 4. Thus, 10 and 4 are added and

assigned back to a.

Same logic applies to `b = b + (b = 5);` as well.

Back to Question without Answer

52. QID - 2.1215 : Working with Inheritance

Consider the following classes :

```
class A{
   public void mA(){ };
}

class B extends A {
   public void mA(){ }
   public void mB() { }
}

class C extends B {
   public void mC(){ }
}
```

and the following declarations:

```
A x = new B(); B y = new B(); B z = new C();
```

Which of the following calls are polymorphic calls?

Correct Options are : A C E

A. `x.mA();`

~~B.~~ `x.mB();`

C. `y.mA();`

~~D.~~ `z.mC();`

E. `z.mB();`

Explanation:

A Polymorphic call means that irrespective of the type of the variable, the method of the actual class of the object referred by the variable is invoked. In java, all non-private method calls are polymorphic because which method is invoked is decided at runtime based on the class of the object instead of compile time.

In this case, `x.mB()` is invalid call. It will not even compile because the class of x is A, which does not contain method `mB()`. Even though the object referred to by x is of class B which does contain `mB()`. `z.mC()` is invalid for the same reason.

Back to Question without Answer

53. QID - 2.968 : Working with Java Data Types - String, StringBuilder

Which of these expressions will obtain the substring "456" from a string defined by `String str = "01234567"`?

Correct Option is : A

A. `str.substring(4, 7)`

~~B.~~ `str.substring(4)`

It will return "4567".

~~C.~~ `str.substring(3, 6)`

It will return "345".

~~D.~~ `str.substring(4, 6)`

It will return "45".

~~E.~~ `str.substring(4, 3)`

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: -1

Explanation:

Read this carefully:
public String substring(int beginIndex, int endIndex)
Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the

length of the substring is endIndex-beginIndex.

"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)

Back to Question without Answer

54. QID - 2.1102 : Constructors

Under what situations does a class get a default constructor?

Correct Option is : C

~~**A.**~~ All classes in Java get a default constructor.

> No. If a class defines a constructor explicitly, it will not get the default constructor.

~~**B.**~~ You have to define at least one constructor to get the default constructor.

> A default (no args one) will be given if the class doesn't define any.

C. If the class does not define any constructors explicitly.

> In this case, the compiler will add a no args constructor for this class.

~~**D.**~~ All classes get default constructor from Object class.

> Constructors are NEVER inherited.

~~**E.**~~ None of the above.

Back to Question without Answer

55. QID - 2.829 : Working with Methods

What will the following program print when compiled and run:

```
public class TestClass {
    public static void main(String[] args) {
            someMethod();
    }

    static void someMethod(Object parameter)  {
          System.out.println("Value is "+parameter);
    }
}
```

Correct Option is : A

A. It will not compile.

To call someMethod(Object parameter), you must pass exactly one parameter. So someMethod() is not a valid call to this method and the code will not compile.
Note that parameter will not be assigned a null or default value.

However, if the method were declared to take variable number of arguments, it would have been valid to call it without any parameters. For example:

```
public static void someMethod(Object... params){
    System.out.println(params.length);
}
```

In this case, calling someMethod() without any parameter will print 0. i.e. the length of params array will be 0. params will NOT be null.

B. ~~`Value is null`~~

~~**C.**~~ `Value is`

~~**D.**~~ It will throw a `NullPointerException` at run time.

Back to Question without Answer

56. QID - 2.913 : Creating and Using Arrays

Which of the following statements about an array are correct?

Correct Option is : E

~~**A.**~~ An array can dynamically grow in size.

> Arrays cannot grow in size once created. ArrayLists can do that.

~~**B.**~~ Arrays can be created only for primitive types.

> You can have arrays for objects also. For example:
> ```
> Object[] objArray = new Object[4];
> String[] arrayOfStrings = { "a", "b" };
> ```

~~**C.**~~ Every array has a built in property named 'size' which tells you the number of elements in the array.

> It is named `length` and not `size`. `ArrayList` has a method named `size()` that returns the number of elements in the ArrayList.
>
> ```
> String[] sa = { "a", "b" };
> int k = sa.length; //k will be assigned a value of 2.
>
> ArrayList al = new ArrayList();
> int k = al.size(); //k will be assigned a value of 0.
> ```

~~**D.**~~ Every array has in implicit method named '`length`' which tells you the number of elements in the array.

E. Element indexing starts at 0.

Back to Question without Answer

57. QID - 2.1211 : Handling Exceptions

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      Exception e = null;
      throw e;
   }
}
```

Correct Option is : A

A. The code will fail to compile.

~~**B.**~~ The program will fail to compile, since it cannot throw a `null`.

~~**C.**~~ The program will compile without error and will throw an `Exception` when run.

~~**D.**~~ The program will compile without error and will throw `java.lang.NullPointerException` when run

~~**E.**~~ The program will compile without error and will run and terminate without any output.

Explanation:

You are throwing an exception and there is no try or catch block, or a throws clause. So it will not compile.
If you do either put a try catch block or declare a throws clause for the method then it will throw a `NullPointerException` at run time because e is `null`.

A method that throws a 'checked' exception i.e. an exception that is not a subclass of `Error` or `RuntimeException`, either must declare it in throws clause or put the code that throws the exception in a try/catch block.

Back to Question without Answer

58. QID - 2.1101 : Working with Inheritance

Consider the following code:

```
class Base{
 private float f = 1.0f;
 void setF(float f1){ this.f = f1; }
}
class Base2 extends Base{
 private float f = 2.0f;
 //1
}
```

Which of the following options is a valid example of overriding?

Correct Options are : A C

A. `protected void setF(float f1){ this.f = 2*f1; }`

protected is less restrictive than default, so it is valid.

~~B.~~ `public void setF(double f1){ this.f = (float) 2*f1; }`

Since the parameter type is different, it is overloading not overriding.

C. `public void setF(float f1){ this.f = 2*f1; }`

public is less restrictive than default, so it is valid.

~~D.~~ `private void setF(float f1){ this.f = 2*f1; }`

private is more restrictive than default, so it is NOT valid.

E. ~~`float setF(float f1){ this.f = 2*f1; return f;}`~~

return types must match.

Explanation:

An overriding method can be made less restrictive than the overridden method. The restrictiveness of access modifiers is as follows:
private>default>protected>public (where private is most restrictive and public is least restrictive).

Note that there is no modifier named default. The absence of any access modifiers implies default access.

Back to Question without Answer

59. QID - 2.837 : Working with Inheritance

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Bird) System.out.println("f is a Bird");
        if(e instanceof Flyer) System.out.println("e is a Flyer");
        if(b instanceof Flyer) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an exception when run.

C. f is a Bird
e is a Flyer

At run time, f points to an object of class Eagle. Now, Eagle extends Bird so f

instanceof Bird returns true.
e points to an object of class Eagle. Eagle extends Bird, which in turn implements Flyer. So an Eagle is a Flyer. Therefore, e instanceof Flyer will also return true.

b points to an object of class Bat, which does not extend from Bird. Therefore, b instanceof Flyer returns false.

~~**D.** f is a Bird~~

~~**E.** e is a Flyer~~

Explanation:

Note that there is no compilation issue with `b instanceof Flyer` because `Flyer` is an interface and it is possible for b to point to an object of a class that is a sub class of `Bat` and also implements `Flyer`. So the compiler doesn't complain. If you make Bat class as final, `b instanceof Flyer` will not compile because the compiler knows that it is not possible for b to point to an object of a class that implements `Flyer`.

Back to Question without Answer

60. QID - 2.845 : Java Basics

The options below contain the complete contents of a file (the name of the file is not specified).
Which of these options can be run with the following command line once compiled?

```
java main
```

Correct Option is : D

~~A.~~
```
//in file main.java
class main {
   public void main(String[] args) {
       System.out.println("hello");
   }
}
```

The main method should be static.

~~B.~~
```
//in file main.java
   public static void main4(String[] args) {
       System.out.println("hello");
   }
```

You cannot have a method on its own. It must be a part of a class.

~~C.~~
```
//in file main.java
public class anotherone{
}
class main {
   public static void main(String[] args) {
       System.out.println("hello");
   }
}
```

A public class must exist in a file by the same name. So this code is invalid

because anotherone is a public class but the name of the file is main. It would have been valid if the name of the file were anotherone.java.

A non public class may exist in any file. This implies that there can be only one public class in a file.

D.
```
//in file main.java
class anothermain{
   public static void main(String[] args) {
       System.out.println("hello2");
   }
}
class main {
   public final static void main(String[] args) {
       System.out.println("hello");
   }
}
```

class main's main method will be executed. final is a valid modifier for the standard main method.

Note that final means a method cannot be overridden. Although static methods can never be overridden. (they can be shadowed), making a static method final prevents the subclass from implementing the same static method.

Explanation:

Observe that the given code does not follow the standard Java naming convention. The class names should start with a capital letter.

There are questions in the exam that contain similar non-conventional and confusing names and that is why we have kept a few questions like that in this question bank.

Back to Question without Answer

61. QID - 2.859 : Using Operators and Decision Constructs

What will the following code print when run?

```
public class TestClass {

    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "c" : System.out.println( "cat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("c");
    }
}
```

Correct Option is : C

~~**A.**~~ apple
cat
none

~~**B.**~~ apple
cat

C. cat
none

Since there is a case condition that matches the input string "c", that case statement will be executed directly. This prints "cat". Since there is no break after this case statement and the next case statement, the control will fall through the next one (which is default :) and so "none" will be printed as well.

~~**D.** cat~~

Explanation:

In the JDK 7 release, you can use a String object in the expression of a switch statement:

```
public String getTypeOfDayWithSwitchStatement(String dayOfWeekArg) {
     String typeOfDay;
     switch (dayOfWeekArg) {
         case "Monday":
             typeOfDay = "Start of work week";
             break;
         case "Tuesday":
         case "Wednesday":
         case "Thursday":
             typeOfDay = "Midweek";
             break;
         case "Friday":
             typeOfDay = "End of work week";
             break;
         case "Saturday":
         case "Sunday":
             typeOfDay = "Weekend";
             break;
         default:
             throw new IllegalArgumentException("Invalid day of the
week: " + dayOfWeekArg);
     }
     return typeOfDay;
}
```

The switch statement compares the String object in its expression with the expressions

associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

Back to Question without Answer

62. QID - 2.1038 : Using Operators and Decision Constructs

What will be the output of the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true ;
      boolean bool2 = false;
      boolean bool  = false;
      bool = ( bool2 &  method1(i++) ); //1
      bool = ( bool2 && method1(i++) ); //2
      bool = ( bool1 |  method1(i++) ); //3
      bool = ( bool1 || method1(i++) ); //4
      System.out.println(i);
   }
   public static boolean method1(int i){
       return i>0 ? true : false;
   }
}
```

Correct Option is : B

~~**A.**~~ It will print 1.

B. It will print 2.

~~**C.**~~ It will print 3.

~~**D.**~~ It will print 4.

~~**E.**~~ It will print 0.

Explanation:

& and | do not short circuit the expression but && and || do.
As the value of all the expressions (1 through 4) can be determined just by looking at the first part, && and || do not evaluate the rest of the expression, so method1() is not called for 2 and 4.
Hence the value of i is incremented only twice.

Back to Question without Answer

63. QID - 2.1219 : Working with Inheritance

What will be the output of compiling and running the following program:

```
class TestClass implements I1, I2{
   public void m1() { System.out.println("Hello"); }
   public static void main(String[] args){
      TestClass tc = new TestClass();
      ( (I1) tc).m1();
   }
}
interface I1{
   int VALUE = 1;
   void m1();
}
interface I2{
   int VALUE = 2;
   void m1();
}
```

Correct Option is : A

A. It will print `Hello`.

~~**B.**~~ There is no way to access any `VALUE` in TestClass.

~~**C.**~~ The code will work fine only if `VALUE` is removed from one of the interfaces.

> It works even now.

~~**D.**~~ It will not compile.

~~**E.**~~ None of the above.

Explanation:

Having ambiguous fields does not cause any problems but referring to such fields in an ambiguous way will cause a compile time error. So you cannot call :

`System.out.println(VALUE)` as it will be ambiguous.

as there is no ambiguity in referring the field:

```
TestClass tc = new TestClass();
System.out.println(( ( I1) tc).VALUE);
```

So, any of the `VALUE` fields can be accessed by casting.

Back to Question without Answer

64. QID - 2.1326 : Creating and Using Arrays

What will happen when the following code is compiled and run?

```
class AX{
  static int[] x = new int[0];
  static{
   x[0] = 10;
  }
  public static void main(String[] args){
    AX ax = new AX();
  }
}
```

Correct Option is : C

~~**A.**~~ It will throw `NullPointerException` at runtime.

~~**B.**~~ It will throw `ArrayIndexOutOfBoundsException` at runtime.

C. It will throw `ExceptionInInitializerError` at runtime.

The following is the output when the program is run:

```
java.lang.ExceptionInInitializerError
Caused by: java.lang.ArrayIndexOutOfBoundsException: 0
        at AX.<clinit>(SM.java:6)
Exception in thread "main"
Java Result: 1
```

Note that the program ends with `ExceptionInInitializerError` because any exception thrown in a static block is wrapped into `ExceptionInInitializerError` and then that `ExceptionInInitializerError` is thrown.

~~**D.**~~ It will not compile.

Explanation:

Even though the line `x[0] = 10;` will throw java.lang.ArrayIndexOutOfBoundsException, JVM will wrap it and rethrow java.lang.ExceptionInInitializerError.

Back to Question without Answer

65. QID - 2.1132 : Java Basics

Which of the following is/are illegal Java identifier(s)?

Correct Option is : C

~~**A.**~~ num

~~**B.**~~ int123

> Can contain digits but cannot start with a digit.

C. 2Next

> Cannot start with a digit.

~~**D.**~~ _interface

> _ and $ are valid at any position.

~~**E.**~~ a$_123

> _ and $ are valid at any position.

Explanation:

A valid java identifier is composed of a sequence of java letters and digits, the first of which must be a letter.
(According to JLS.)
It is not clear from the objectives whether this topic is included or not. Regardless, it is good to know how to declare a valid identifier.

Back to Question without Answer

66. QID - 2.1228 : Working with Methods

What will be the result of attempting to compile and run the following class?

```
public class InitTest{
   static String s1 = sM1("a");{
      s1 = sM1("b");
   }
   static{
      s1 = sM1("c");
   }
   public static void main(String args[]){
      InitTest it = new InitTest();
   }
   private static String sM1(String s){
      System.out.println(s);  return s;
   }
}
```

Correct Option is : B

~~**A.**~~ The program will fail to compile.

B. The program will compile without error and will print `a, c` and `b` in that order when run.

~~**C.**~~ The program will compile without error and will print `a, b` and `c` in that order when run.

~~**D.**~~ The program will compile without error and will print `c, a` and `b` in that order when run.

~~**E.**~~ The program will compile without error and will print `b, c` and `a` in that order when run.

Explanation:

First, static statements/blocks are called IN THE ORDER they are defined. (Hence, a and c will be printed.)
Next, instance initializer statements/blocks are called IN THE ORDER they are defined. Finally, the constructor is called. So, then it prints b.

Back to Question without Answer

67. QID - 2.992 : Java Basics

Given the following program, which statement is true?

```
class SomeClass{
   public static void main( String args[ ] ){
      if (args.length == 0 ){
         System.out.println("no arguments") ;
      }
      else{
         System.out.println( args.length + " arguments") ;
      }
   }
}
```

Correct Option is : C

~~**A.**~~ The program will fail to compile.

~~**B.**~~ The program will throw a `NullPointerException` when run with zero arguments.

C. The program will print `no arguments` and `1 arguments` when called with zero and one arguments.

> The word java and class name are not a part of the argument list.

~~**D.**~~ The program will print `no arguments` and `2 arguments` when called with zero and one arguments.

~~**E.**~~ The program will print `no arguments` and `3 arguments` when called with zero and one arguments.

> When the program is called with no arguments, the `args` array will be of length zero.

Explanation:

When the program is called with no arguments, the `args` array will be of length zero. Unlike in C/C++, args[0] is not the name of the program or class. This is because the name of the class is always the same that was defined in the Java file. So there is no need for passing its name as an argument to main method.

Back to Question without Answer

68. QID - 2.1036 : Creating and Using Arrays

Consider the following class...

```
class Test{
   public static void main(String[ ] args){
      int[] a = { 1, 2, 3, 4 };
      int[] b = { 2, 3, 1, 0 };
      System.out.println( a [ (a = b)[3] ] );
   }
}
```

What will it print when compiled and run ?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw `ArrayIndexOutOfBoundsException` when run.

C. It will print 1.

~~**D.**~~ It will print 3.

~~**E.**~~ It will print 4

Explanation:

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated. In the expression `a[(a=b)[3]]`, the expression a is fully evaluated before the

expression `(a=b)[3];` this means that the original value of a is fetched and remembered while the expression `(a=b)[3]` is evaluated. This array referenced by the original value of a is then subscripted by a value that is element 3 of another array (possibly the same array) that was referenced by b and is now also referenced by a. So, it is actually `a[0] = 1`.
Note that if evaluation of the expression to the left of the brackets completes abruptly, no part of the expression within the brackets will appear to have been evaluated.

Back to Question without Answer

69. QID - 2.904 : Working with Methods

Given:

```
class Triangle{
    public int base;
    public int height;
    private static double ANGLE;

    public static double getAngle();

    public static void Main(String[] args) {
        System.out.println(getAngle());
    }
}
```

Identify the correct statements:

Correct Option is : C

~~**A.**~~ It will not compile because it does not implement `setAngle` method.

> There is no requirement that a class has to have a setter as well as a getter.

~~**B.**~~ It will not compile because `ANGLE` cannot be private.

> Any field can be made private.

C. It will not compile because `getAngle()` has no body.

~~**D.**~~ It will not compile because `ANGLE` field is not initialized.

> Since it is a static field, it will get a default value of 0.0.

~~**E.**~~ It will not compile because of the name of the method `Main` instead of `main`.

A class can have a method named Main. Although, since it is not same as main, it will not be considered the standard main method that the JVM can invoke when the program is executed.

Back to Question without Answer

70. QID - 2.995 : Working with Inheritance

Which of these statements about interfaces are true?

Correct Options are : A C E

A. Interfaces are abstract by default.

> Because they don't have any implementation and so can't be instantiated.

~~**B.**~~ An interface can have static methods.

C. All methods in an interface are abstract although you need not declare them to be so.

~~**D.**~~ Fields of an interface may be declared as transient or volatile but not synchronized.

E. interfaces cannot be final.

Explanation:

Here are the rules:
1. Every interface is implicitly `abstract`. This modifier is obsolete for interfaces and should not be used in new Java programs.
2. An interface can extend any number of other interfaces and can be extended by any number of other interfaces
3. Every field declaration in the body of an interface is implicitly `public`, `static` and `final`. It is permitted, but strongly discouraged as a matter of style, to redundantly specify any or all of these modifiers for such fields. A constant declaration in an interface must not include any of the modifiers `synchronized`, `transient` or

`volatile`, or a compile-time error occurs.
4.It is possible for an interface to inherit more than one field with the same name. Such a situation does not in itself cause a compile-time error. However, any attempt within the body of the interface to refer to either field by its simple name will result in a compile-time error, because such a reference is ambiguous.
5. Every method declaration in the body of an interface is implicitly `public` and `abstract`, so its body is always represented by a semicolon, not a block.
6. A method in an interface cannot be declared static, because in Java static methods cannot be abstract.
7. A method in an interface cannot be declared `native` or `synchronized`, or a compile-time error occurs, because those keywords describe implementation properties rather than interface properties. However, a method declared in an interface may be implemented by a method that is declared `native` or `synchronized` in a class that implements the interface.

Back to Question without Answer

71. QID - 2.1050 : Using Loop Constructs

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      for (i=1 ;  i<5  ; i++) continue;  //(1)
      for (i=0 ;       ; i++) break;       //(2)
      for (    ; i<5?false:true ;    );     //(3)
   }
}
```

Correct Option is : A

A. The code will compile without error and will terminate without problems when run.

B. The code will fail to compile, since the `continue` can't be used this way.

C. The code will fail to compile, since the `break` can't be used this way

D. The code will fail to compile, since the `for` statement in the line 2 is invalid.

E. The code will compile without error but will never terminate.

the condition part is 'false' so the control will never go inside the loop.

Explanation:

A continue statement can occur in and only in a for, while or do-while loop. A

continue statement means: Forget about the rest of the statements in the loop and start the next iteration.
So,
`for (i=1 ;  i<5  ; i++) continue;` just increments the value of i up to 5 because of i++.

`for (i=0 ;       ; i++) break;` iterates only once because of the break so the value of i becomes 0.

`for (    ; i<5?false:true ;    );` never iterates because i is less than 5 (it is 0 because of //2) and the condition expression is false!

At the end of the code, the value of i is 0.

Back to Question without Answer

72. QID - 2.1313 : Using Operators and Decision Constructs

What is the result of executing the following code when the value of i is 5:

```
switch (i){
    default:
    case 1:
        System.out.println(1);
    case 0:
        System.out.println(0);
    case 2:
        System.out.println(2);
        break;
    case 3:
        System.out.println(3);
}
```

Correct Option is : A

A. It will print `1 0 2`

~~**B.**~~ It will print `1 0 2 3`

~~**C.**~~ It will print `1 0`

~~**D.**~~ It will print `1`

~~**E.**~~ Nothing will be printed.

Explanation:

Here are the rules:

The type of the Expression must be char, byte, short, or int or a compile-time error occurs. Java 7 allows String as well.
The class of the switch variable may also be a Wrapper class i.e. Character, Byte, Short, or Integer.

All of the following must be true, or a compile-time error will result:
1.Every case constant expression associated with a switch statement must be assignable (5.2) to the type of the switch Expression.
2.No two of the case constant expressions associated with a switch statement may have the same value.
3.At most one default label may be associated with the same switch statement.
Basically it looks for a matching case or if no match is found it goes to default. (If default is also not found it does nothing)
Then it executes the statements till it reaches a break or end of the switch statement.
Here it goes to default and executes till it reaches first break. So it prints 1 0 2.

Note that the switch statement compares the String object in its expression with the expressions associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

Back to Question without Answer

73. QID - 2.1162 : Java Basics

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){}
}
```

Correct Option is : C

~~**A.**~~ The class `TestClass` cannot be declared `abstract`.

> Any class can be declared `abstract`.

~~**B.**~~ The variable `j` cannot be declared `transient`.

C. The variable `k` cannot be declared `synchronized`.

> Variables cannot be declared `synchronized`. Only methods can be declared `synchronized`.

~~**D.**~~ The constructor `TestClass( )` cannot be declared final.

> It is not a constructor, it is a simple method. Notice `void` return type.

~~**E.**~~ The method `f( )` cannot be declared static.

~~**F.**~~ This code has no errors.

Back to Question without Answer

74. QID - 2.1308 : Working with Methods

What is the result of compiling and running the following code ?

```
public class TestClass{
   static int si = 10;
   public static void main (String args[]){
      new TestClass();
   }
   public TestClass(){
      System.out.println(this);
   }
   public String toString(){
      return "TestClass.si = "+this.si;
   }
}
```

Correct Option is : D

~~A.~~ The class will not compile because you cannot override `toString()` method.

> You sure can. `toString()` is defined as public and non-final method in Object class.

~~B.~~ The class will not compile as `si` being `static`, `this.si` is not a valid statement.

> static member can be accessed by static and non-static methods both. Non-static can only be accessed by non-static.

~~C.~~ It will print `TestClass@nnnnnnnn`, where `nnnnnnn` is the hash code of the TestClass object referred to by 'this'.

> It would have been correct if `toString()` were not overridden. This is the behavior of the `toString()` provided by Object class.

D. It will print `TestClass.si = 10;`

~~**E.**~~ None of the above.

Explanation:

The `toString` method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@', and the unsigned hexadecimal representation of the hash code of the object. In other words, this method returns a string equal to the value of:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

Back to Question without Answer

75. QID - 2.932 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Options are : B C

~~**A.**~~ The modulus operator % can only be used with integer operands.

> It can be used on floating points operands also. For example, 5.5 % 3 = 2.5

B. & can have integral as well as boolean operands.

> unlike &&, & will not "short circuit" the expression if used on boolean parameters.

C. The arithmetic operators *, / and % have the same level of precedence.

~~**D.**~~ && can have integer as well as boolean operands.

> !, && and || operate only on booleans.

~~**E.**~~ ~ can have integer as well as boolean operands.

> ~ Operates only on integral types

Explanation:

Note :
integral types means byte, short, int, long, and char

Reference: http://java.sun.com/docs/books/jls/third_edition/html/typesValues.html

"The types of the Java programming language are divided into two categories: primitive types and reference types. The primitive types (§4.2) are the boolean type and the numeric types. The numeric types are the integral types byte, short, int, long, and char, and the floating-point types float and double. The reference types (§4.3) are class types, interface types, and array types. There is also a special null type. An object (§4.3.1) is a dynamically created instance of a class type or a dynamically created array. The values of a reference type are references to objects. All objects, including arrays, support the methods of class Object (§4.3.2). String literals are represented by String objects (§4.3.3)."

Back to Question without Answer

76. QID - 2.971 : Creating and Using Arrays

What will be the result of trying to compile and execute of the following program?

```
public class TestClass{
   public static void main(String args[] ){
      int i = 0 ;
      int[] iA = {10, 20} ;
      iA[i] = i = 30 ;
      System.out.println(""+ iA[ 0 ] + " " + iA[ 1 ] + "  "+i) ;
    }
}
```

Correct Option is : D

~~**A.**~~ It will throw ArrayIndexOutofBoundsException at Runtime.

~~**B.**~~ Compile time Error.

~~**C.**~~ It will prints 10 20 30

D. It will prints 30 20 30

~~**E.**~~ It will prints 0 20 30

Explanation:

The statement iA[i] = i = 30 ; will be processed as follows:
iA[i] = i = 30; => iA[0] = i = 30 ; => i = 30; iA[0] = i ; => iA[0] = 30 ;

Here is what JLS says on this:

1 Evaluate Left-Hand Operand First
2 Evaluate Operands before Operation
3 Evaluation Respects Parentheses and Precedence
4 Argument Lists are Evaluated Left-to-Right

For Arrays: First, the dimension expressions are evaluated, left-to-right. If any of the expression evaluations completes abruptly, the expressions to the right of it are not evaluated.

Back to Question without Answer

77. QID - 2.1201 : Working with Inheritance

Consider the following interface definition:

```
interface Bozo{
         int type = 0;
         public void jump();
}

Now consider the following class:

public class Type1Bozo implements Bozo{
         public Type1Bozo(){
            type = 1;
         }

         public void jump(){
            System.out.println("jumping..."+type);
         }

         public static void main(String[] args){
            Bozo b = new Type1Bozo();
            b.jump();
         }
}
```

What will the program print when compiled and run?

Correct Option is : C

~~**A.**~~ jumping...0

~~**B.**~~ jumping...1

C. This program will not compile.

~~**D.**~~ It will throw an exception at runtime.

Explanation:

Fields defined in an interface are ALWAYS considered as public, static, and final. (Methods are public and abstract.) Even if you don't explicitly define them as such. In fact, you cannot even declare a field to be private or protected in an interface. Therefore, you cannot assign any value to 'type' outside the interface definition.

Back to Question without Answer

78. QID - 2.868 : Working with Java Data Types - String, StringBuilder

How can you initialize a StringBuilder to have a capacity of at least 100 characters?

Correct Options are : A D

A. `StringBuilder sb = new StringBuilder(100);`

> public StringBuilder(int capacity)
> Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.

~~B.~~ `StringBuilder sb = StringBuilder.getInstance(100);`

~~C.~~
```
StringBuilder sb = new StringBuilder();
sb.setCapacity(100);
```

D.
```
StringBuilder sb = new StringBuilder();
sb.ensureCapacity(100);
```

> public void ensureCapacity(int minimumCapacity)
> Ensures that the capacity is at least equal to the specified minimum. If the current capacity is less than the argument, then a new internal array is allocated with greater capacity. The new capacity is the larger of:
> The minimumCapacity argument.
> Twice the old capacity, plus 2.
> If the minimumCapacity argument is nonpositive, this method takes no action and simply returns.

Explanation:

Observe that the question says "at least 100 characters". In the exam, you may get a question that says "100 characters", in that case, ensureCapacity() may not be a valid option.

Back to Question without Answer

79. QID - 2.854 : Working with Methods

What will be printed when the following code is compiled and run?

```
class A {
    public int getCode(){ return 2;}
}

class AA extends A {
  public long getCode(){ return 3;}
}

public class TestClass {

    public static void main(String[] args) throws Exception {
         A a = new A();
        A aa = new AA();
        System.out.println(a.getCode()+" "+aa.getCode());
    }

    public int getCode() {
        return 1;
    }
}
```

Correct Option is : D

~~**A.**~~ 2 3

~~**B.**~~ 2 2

~~**C.**~~ It will throw an exception at run time.

D. The code will not compile.

Class AA is trying to override `getCode()` method of class A but its return type is incompatible with the A's `getCode`.

When the return type of the overridden method (i.e. the method in the base/super class) is a primitive, the return type of the overriding method (i.e. the method in the sub class) must match the return type of the overridden method.

In case of Objects, the base class method can have a covariant return type, which means, it can return either return the same class or a sub class object. For example, if base class method is:

```
public A getA(){ ... }
```

a subclass can override it with:

`public AA getA(){ ... }` because AA is a subclass of A.

Back to Question without Answer

80. QID - 2.990 : Overloading methods

Consider the following classes...

```
class Teacher{
      void teach(String student){
          /* lots of code */
      }
}
class Prof extends Teacher{
        //1
}
```

Which of the following methods can be inserted at line //1 ?

Correct Options are : A B C D

A. `public void teach() throws Exception`

> It overloads the `teach()` method instead of overriding it.

B. `private void teach(int i) throws Exception`

> It overloads the `teach()` method instead of overriding it.

C. `protected void teach(String s)`

> This overrides Teacher's teach method. The overriding method can have more visibility. (It cannot have less. Here, it cannot be private.)

D. `public final void teach(String s)`

> Overriding method may be made final.

~~**E.**~~ `public abstract void teach(String s)`

This is wrong because class Prof has not been declared as abstract. Note that otherwise it is OK to override a method by an abstract method.

Explanation:

Note that 'protected' is less restrictive than default 'no modifier'. So choice 3 is valid. "public abstract void teach(String s)" would have been valid if class Prof had been declared abstract.

Back to Question without Answer

81. QID - 2.1153 : Working with Java Data Types - Variables and Objects

What will the following program print?

```
public class TestClass{
  static String str;
  public static void main(String[] args){
     System.out.println(str);
  }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will compile but throw an exception at runtime.

C. It will print 'null' (without quotes).

~~**D.**~~ It will print nothing.

~~**E.**~~ None of the above.

Explanation:

All member fields (static and non-static) are initialized to their default values. Objects are initialized to null (String is also an object), numeric types to 0 (or 0.0) and boolean to false.

Back to Question without Answer

82. QID - 2.1317 : Using Operators and Decision Constructs

Which of the following are NOT valid operators in Java?

Correct Options are : A B D E

A. sizeof

> It is in C++ but not in java because size of everything is known at compile time and is not machine dependent.

B. <<<

> For left shifts there is no difference between shifting signed and unsigned values so there is only one leftshift '<<' in java.

~~**C.**~~ instanceof

D. mod

> No such thing.

E. equals

> boolean equals(Object o) is a method in java.lang.Object. It is not an operator.

Back to Question without Answer

83. QID - 2.843 : Creating and Using Arrays

Given the following declaration:

```
int[][] twoD = { { 1, 2, 3} , { 4, 5, 6, 7}, { 8, 9, 10 } };
```

What will the following lines of code print?

```
System.out.println(twoD[1].length);
System.out.println(twoD[2].getClass().isArray());
System.out.println(twoD[1][2]);
```

Correct Option is : A

A.
```
4
true
6
```

B.
```
3
true
3
```

C.
```
3
false
3
```

D.
```
4
false
6
```

Explanation:

In Java, array numbering starts from 0. So in this case, `twoD` is an array containing 3

other arrays.
`twoD[0]` is `{ 1, 2, 3}` , `twoD[1]` is `{ 4, 5, 6, 7}`, and `twoD[2]` is `{ 8, 9, 10 }`.

Thus, `twoD[1].length` is `4` and `twoD[1][2]` is the third element in `{ 4, 5, 6, 7}`, which is `6`.

In Java, arrays are just like regular Objects and arrays of different types have different class names. For example, the class name of an `int[]` is `[I` and the class name for `int[][]` is `[[I`.

For array classes, the `isArray()` method of a Class returns true. For example, `twoD.getClass().isArray()` will return true.

There are a few questions in the exam that require you to know about this.

Back to Question without Answer

84. QID - 2.1121 : Using Loop Constructs

What will the following code print?

```
void crazyLoop(){
  int c = 0;
  JACK: while (c < 8){
    JILL: System.out.println(c);
    if (c > 3) break JACK; else c++;
  }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ It will print numbers from 0 to 8

~~**D.**~~ It will print numbers from 0 to 3

E. It will print numbers from 0 to 4

Explanation:

This is a straight forward loop that contains a labelled break statement. A labelled break breaks out of the loop that is marked with the given label. Therefore, a labelled break is used to break out from deeply nested loops to the outer loops. Here, there is only one nested loop so the break; and break JACK; are same, but consider the

following code:

```
public static void crazyLoop(){
  int c = 0;
  JACK: while (c < 8){
    JILL: System.out.println(c);
    for(int k = 0; k<c; k++){
        System.out.println(" k = "+k+" c = "+c);
        if (c > 3) break JACK;
    }
    c++;
  }
}
```

This code prints:

```
c = 0
c = 1
  k = 0 c = 1
c = 2
  k = 0 c = 2
  k = 1 c = 2
c = 3
  k = 0 c = 3
  k = 1 c = 3
  k = 2 c = 3
c = 4
  k = 0 c = 4
```

As you can see, in this case, break JACK; will break out from the outer most loop (the while loop). If break JACK; is replaced by break; it will print:

```
c = 0
c = 1
  k = 0 c = 1
c = 2
  k = 0 c = 2
  k = 1 c = 2
c = 3
  k = 0 c = 3
  k = 1 c = 3
```

```
   k = 2 c = 3
c = 4
   k = 0 c = 4
c = 5
   k = 0 c = 5
c = 6
   k = 0 c = 6
c = 7
   k = 0 c = 7
```

This shows that a break without a label only breaks out of the current loop.

Back to Question without Answer

85. QID - 2.1199 : Working with Java Data Types - Variables and Objects

Given:

```
public class TestClass{
  public static int getSwitch(String str){
      return (int) Math.round( Double.parseDouble(str.substring(1, st
  }
  public static void main(String args []){
    switch(getSwitch(args[0])){
      case 0 : System.out.print("Hello ");
      case 1 : System.out.print("World"); break;
      default : System.out.print(" Good Bye");
    }
  }
}
```

What will be printed by the above code if it is run with command line:
`java TestClass --0.50`
(There are two minuses before 0.)

Correct Option is : C

~~**A.**~~ Hello

~~**B.**~~ World

C. Hello World

~~**D.**~~ Hello World Good Bye

~~**E.**~~ Good Bye

Explanation:

`str.substring(1, str.length()-1)` => `"--0.50".substring(1, (6-1) )` => `-0.5`
`Math.round(-0.5) = 0.0`
so `getSwitch(...)` returns 0 if passed an argument of `"--0.50"`.
Now, there is no "break" in case 0 of switch. so the control falls through to the next case (i.e. case 1) after printing Hello. At case 1, it prints World. And since there is a break. default is not executed.

Back to Question without Answer

86. QID - 2.1261 : Working with Inheritance

Consider the following code:

```
public class SubClass extends SuperClass{
     int i, j, k;
     public SubClass( int m, int n )     {  i = m ;  j = m ;  } //1
     public SubClass( int m )  {   super(m );   } //2
 }
```

Which of the following constructors MUST exist in SuperClass for SubClass to compile correctly?

Correct Options are : C D

~~**A.**~~ It is ok even if no explicit constructor is defined in SuperClass

> The //2 will fail as it needs a constructor taking an int!

~~**B.**~~ `public SuperClass(int a, int b)`

> It is not used anywhere so it is not necessary.

C. `public SuperClass(int a)`

> Because it is called in the second constructor of SubClass.

D. `public SuperClass()`

> The default no args constructor will not be provided because SuperClass has to define one arg constructor.

~~**E.**~~ only `public SuperClass(int a)` is required.

You'll have to explicitly define a no args constructor because it is needed in the first constructor of SubClass.

Back to Question without Answer

87. QID - 2.1062 : Working with Java Data Types - Variables and Objects

The following code snippet will print 4.

```
int i1 = 1, i2 = 2, i3 = 3;
int i4 = i1 + (i2=i3 );
System.out.println(i4);
```

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

First the value of i1 is evaluated (i.e. 1). Now, `i2` is assigned the value of `i3` i.e. `i2` becomes `3`. Finally `i4` gets 1 +3 i.e. 4.

Back to Question without Answer

88. QID - 2.1269 : Using Operators and Decision Constructs

Which of the following will not give any error at compile time and run time?

Correct Options are : A C D E

A. `if (8 == 81) {}`

> `8 == 81` is a valid expression that returns false.

B. ~~`if (x = 3) {}`~~ // assume that x is an int

> Because the exp. x = 3 does not return a boolean.

C. `if (true) {}`

D. `if (bool = false) {}` //assume that bool is declared as a boolean

> Because the expression 'bool = false' returns a boolean (which happens to be false)

E. `if (x == 10 ? true:false) { }` // assume that x is an int

Explanation:

All an `if(...)` needs is a `boolean`.
`x = 3` is not valid because the return value of this expression is 3 which is not a `boolean`.

Back to Question without Answer

89. QID - 2.852 : Working with Java Data Types - String, StringBuilder

What will the following code print?

```
System.out.println("12345".charAt(6));
```

Correct Option is : F

~~**A.**~~ `5`

~~**B.**~~ `null`

~~**C.**~~ `-1`

~~**D.**~~ It will throw an `ArrayIndexOutOfBoundsException`.

~~**E.**~~ It will throw a `StringOutOfBoundsException`.

> There is no such exception. The correct name is `StringIndexOutOfBoundsException`. But that is also not the correct answer.

F. It will throw an `IndexOutOfBoundsException`

> As per the API documentation of this method (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html#charAt(int)), this method throws `IndexOutOfBoundsException`. Although, in practice the method throws `StringIndexOutOfBoundsException`, which is a subclass of `IndexOutOfBoundsException`, the implementation is free to throw some other subclass of `IndexOutOfBoundsException`. Thus, you should rely only on the

published API documentation instead of what it actually throws.

Explanation:

Since indexing starts with 0, the maximum value that you can pass to charAt is length-1.

As per the API documentation for `charAt`, it throws `IndexOutOfBoundsException` if you pass an invalid value (that is, if the index argument is negative or not less than the length of this string).

Both - `ArrayIndexOutOfBoundsException` and `StringIndexOutOfBoundsException`, extend `IndexOutOfBoundsException` and although in practice, the `charAt` method throws `StringIndexOutOfBoundsException`, it is not a valid option because the implementation is free to throw some other exception as long as it is an `IndexOutOfBoundsException`.

(There are questions on the exam on this aspect.)

Back to Question without Answer

90. QID - 2.1373 : Overloading methods

What should be placed in the two blanks so that the following code will compile without errors:

```
class XXX{
    public void m() throws Exception{
        throw new Exception();
    }
}
class YYY extends XXX{
    public void m() {
    }
}
public class TestClass {
    public static void main(String[] args) {
        ______ obj = new ______();
        obj.m();
    }
}
```

Correct Option is : C

~~**A.**~~ XXX and YYY

~~**B.**~~ XXX and XXX

C. YYY and YYY

~~**D.**~~ YYY and XXX

~~**E.**~~ Nothing will make the code compile.

Explanation:

1. The overriding method may choose to have no `throws` clause even if the overridden method has a `throws` clause.
2. Whether a call needs to be wrapped in a try/catch or whether the enclosing method requires a throws clause depends on the class of the reference and not the class of the actual object.

Here, if you define s of type XXX, the call `s.m()` will have to be wrapped into a try/catch because `main()` doesn't have a throws clause. But if you define s of class YYY, there is no need of try catch because YYY's `m()` does not throw an exception. Now, if the class of s is YYY, you cannot assign it an object of class XXX because XXX is a superclass of YYY. So the only option is to do:

```
YYY s = new YYY();
```

Back to Question without Answer

Test 5

This test contains a few questions on certain aspects of inner classes and primitive wrapper, which may be too advanced for this certification. These questions are labelled accordingly in their problem statements. You may ignore these questions if you are short on time.

01. QID - 2.1120

What will the following program snippet print?

```
int i=0, j=11;
do{
   if(i > j) { break; }
   j--;
}while( ++i < 5);
System.out.println(i+"  "+j);
```

Select 1 option

A. 5 5

B. 5 6

C. 6 6

D. 6 5

E. 4 5

Check Answer

02. QID - 2.1122

Note: This question may be considered too advanced for this exam. What will the code shown below print when compiled and run with the following command line?

```
java WarZone self
```

```
interface XMen {
   void shoot(String a);
}

public class WarZone {
   public static void main(String[] args){
       XMen x = null;
       if(args.length() > 0){
          x = new XMen(){
             public void shoot(String s){
                for(int i=0; i<s.length; i++){
                  System.out.println("shot : "+s.charAt(i));
                }
             }
          };
       }

       if(x != null){
          x.shoot(args[0]);
       }
   }
}
```

Select 1 option

A. It will not compile because interface XMen cannot be instantiated.

B. It will print `shot :` 4 times, one at each line.

C. It will print "`shot : s`", "`shot : e`", "`shot : l`", "`shot : f`" one by one on 4 lines.

D. It will compile but will throw an exception at runtime.

E. None of these options is correct.

Check Answer

03. QID - 2.1018

Note: This question may be considered too advanced for this exam. Given:

```
class MySuper{
    public MySuper(int i){ }
}

abstract class MySub extends MySuper{
    public MySub(int i){ super(i); }
    public abstract void m1();
}

class MyTest{
    public static void main(String[] args){
        MySub ms = new MySub(){
            public void m1() { System.out.println("In MySub.m1()");
        };
        ms.m1();
    }
}
```

What will be the output when the above code is compiled and run?

Select 1 option

A. It will not compile.

B. It will throw an exception at run time.

C. It will print `In MySub.m1()`

D. It will compile and run without any exception but will not print anything.

Check Answer

04. QID - 2.1003

Given the following code, which statements are true?

```
interface Automobile { String describe(); }

class FourWheeler implements Automobile{
   String name;
   public String describe(){ return " 4 Wheeler " + name; }
}

class TwoWheeler extends FourWheeler{
    String name;
    public String describe(){ return " 2 Wheeler " + name; }
}
```

Select 3 options

A. An instance of `TwoWheeler` is also an instance of `FourWheeler`.

B. An instance of `TwoWheeler` is a valid instance of `Automobile`.

C. The use of inheritance is not justified here because a `TwoWheeler` is not really a `FourWheeler`.

D. The code will compile only if `name` is removed from `TwoWheeler`.

E. The code will fail to compile.

Check Answer

05. QID - 2.1252

What will be the output when the following program is run?

```
public class TestClass{
    char c;
    public void m1(){
        char[ ] cA = { 'a' , 'b'};
        m2(c, cA);
        System.out.println( ( (int)c)  + ", " + cA[1] );
    }
    public void m2(char c, char[ ] cA){
        c = 'b';
        cA[1] = cA[0] = 'm';
    }
    public static void main(String args[]){
        new TestClass().m1();
    }
}
```

Select 1 option

A. Compile time error.

B. , m

C. 0 , m

D. b , b

E. b , m

Check Answer

06. QID - 2.1106

Is it possible to create arrays of length zero?

Select 1 option

A. Yes, you can create arrays of any type with length zero.

B. Yes, but only for primitive datatypes.

C. Yes, but only for arrays of object references.

D. Yes, and it is same as a null Array.

E. No, arrays of length zero do not exist in Java.

Check Answer

07. QID - 2.1260

What will be the output of the following class...

```
class Test{
   public static void main(String[] args){
      int j = 1;
      try{
         int i = doIt() / (j = 2);
      } catch (Exception e){
         System.out.println(" j = " + j);
      }
   }
   public static int doIt() throws Exception {  throw new Exception('
}
```

Select 1 option

A. `It will print j = 1;`

B. `It will print j = 2;`

C. The value of j cannot be determined.

D. It will not compile.

E. None of the above.

Check Answer

08. QID - 2.1353

Which of the lines will cause a compile time error in the following program?

```
public class MyClass{
   public static void main(String args[]){
      char c;
      int i;
      c = 'a';//1
      i = c;  //2
      i++;    //3
      c = i;  //4
      c++;    //5
   }
}
```

Select 1 option

A. line 1

B. line 2

C. line 3

D. line 4

E. line 5

Check Answer

09. QID - 2.1097

What will be the output of the following program?

```
class TestClass{
   public static void main(String[] args) throws Exception{
      try{
         amethod();
         System.out.println("try");
      }
      catch(Exception e){
         System.out.println("catch");
      }
      finally   {
         System.out.println("finally");
      }
      System.out.println("out");
   }
   public static void amethod(){ }
}
```

Select 1 option

A. `try finally`

B. `try finally out`

C. `try out`

D. `catch finally out`

E. It will not compile because `amethod()` does not throw any exception.

Check Answer

10. QID - 2.1074

Consider the following class definition:

```
public class TestClass{
   public static void main(){  new TestClass().sayHello(); }   //1
   public static void sayHello(){ System.out.println("Static Hello W
   public void sayHello() { System.out.println("Hello World "); }  /
}
```

What will be the result of compiling and running the class?

Select 1 option

A. It will print `Hello World`.

B. It will print `Static Hello World`.

C. Compilation error at line 2.

D. Compilation error at line 3.

E. Runtime Error.

Check Answer

11. QID - 2.956

Consider the following class...

```
class MyString extends String{
   MyString(){ super(); }
}
```

The above code will not compile.

Select 1 option

A. True

B. False

Check Answer

12. QID - 2.1057

What will the following code print?

```
void crazyLoop(){
   int c = 0;
   JACK: while (c < 8){
       JILL: System.out.println(c);
       if (c > 3) break JILL; else c++;
   }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. It will print numbers from 0 to 8

D. It will print numbers from 0 to 3

E. It will print numbers from 0 to 4

Check Answer

13. QID - 2.1156

What will the following class print when executed?

```
class Test{
    static boolean a;
    static boolean b;
    static boolean c;
    public static void main (String[] args){
        boolean bool = (a = true) || (b = true) && (c = true);
        System.out.print(a + ", " + b + ", " + c);
    }
}
```

Select 1 option

A. true, false, true

B. true, true, false

C. true, false, false

D. true, true, true

Check Answer

14. QID - 2.1249

Consider the following class:

```
public class ArgsPrinter{
   public static void main(String args){
      for(int i=0; i<3; i++){
         System.out.print(args+" ");
      }
   }
}
```

What will be printed when the above class is run using the following command line:
`java ArgsPrinter 1 2 3 4`

Select 1 option

A. `1 2 3`

B. `ArgsPrinter 1 2`

C. `java ArgsPrinter 1 2`

D. `1 1 1`

E. None of these.

Check Answer

15. QID - 2.947

Which line contains a valid constructor in the following class definition?

```
public class TestClass{
   int i, j;
   public TestClass getInstance() {  return new TestClass();     }  /,
   public void TestClass(int x, int y) {   i = x;   j = y;    }     /,
   public TestClass TestClass() {   return new TestClass();    }    /,
   public ~TestClass() {     }                    //4
}
```

Select 1 option

A. Line 1

B. Line 2

C. Line 3

D. Line 4

E. None of the above.

Check Answer

16. QID - 2.893

Given:

```
interface Worker {
   void  performWork();
}

class FastWorker  implements Worker {
   public void  performWork(){ }
}
```

You are creating a class that follows "program to an interface" principle. Which of the following line of code will you most likely be using?

Select 1 option

A.
```
public FastWorker getWorker(){
return new Worker();
}
```

B.
```
public FastWorker getWorker(){
return new FastWorker();
}
```

C.
```
public Worker getWorker(){
return new FastWorker();
}
```

D.
```
public Worker getWorker(){
return new Worker();
}
```

Check Answer

17. QID - 2.1155

Which line will print the string "MUM"?

```
public class TestClass{
    public static void main(String args []){
        String s = "MINIMUM";
        System.out.println(s.substring(4, 7)); //1
        System.out.println(s.substring(5)); //2
        System.out.println(s.substring(s.indexOf('I', 3))); //3
        System.out.println(s.substring(s.indexOf('I', 4))); //4
    }
}
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. None of these.

Check Answer

18. QID - 2.1117

Given the following classes, what will be the output of compiling and running the class Truck?

```
class Automobile{
   public void drive() {  System.out.println("Automobile: drive");
}

public class Truck extends Automobile{
   public void drive() {  System.out.println("Truck: drive");   }
   public static void main (String args [ ]){
      Automobile  a = new Automobile();
      Truck t  = new Truck();
      a.drive(); //1
      t.drive(); //2
      a = t;     //3
      a.drive(); //4
   }
}
```

Select 1 option

A. Compiler error at line 3.

B. Runtime error at line 3.

C. It will print:

```
Automobile: drive
Truck: drive
Automobile: drive
```

in that order.

D. It will print:

```
Automobile: drive
Truck: drive
Truck: drive
```

in that order.

E. It will print:

```
Automobile: drive
Automobile: drive
Automobile: drive
```

in that order.

Check Answer

19. QID - 2.860

What will the following code print when run?

```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

Select 1 option

A. bat
big bat

B. big bat
none

C. big bat

D. bat

E. The code will not compile.

Check Answer

20. QID - 2.1033

A try statement must always have a ..... associated with it.

Select 1 option

A. catch

B. throws

C. finally

D. catch, finally or both

E. throw

Check Answer

21. QID - 2.1331

Note: This question may be considered too advanced for this exam.

Which statements can be inserted at line 1 in the following code to make the program write x on the standard output when run?

```
public class AccessTest{
   String a = "x";
   static char b = 'x';
   String  c = "x";
   class Inner{
      String  a = "y";
      String  get(){
         String c = "temp";
         // Line 1
         return c;
      }
   }

   AccessTest() {
     System.out.println(  new Inner().get()  );
   }

   public static void main(String args[]) {  new AccessTest();  }
}
```

Select 3 options

A. `c = c;`

B. `c = this.a;`

C. `c = ""+AccessTest.b;`

D. `c = AccessTest.this.a;`

E. `c = ""+b;`

Check Answer

22. QID - 2.1109

Which of the following statements are true?

Select 2 options

A. method `length()` of String class is a final method.

B. You can make mutable subclasses of the String class.

C. `StringBuilder` extends `String`.

D. StringBuilder is a final class.

E. String class is not final.

Check Answer

23. QID - 2.1126

Note: This question may be considered too advanced for this exam.
Given:

```
String mStr = "123";
long m = // 1
```

Which of the following options when put at //1 will assign `123` to `m`?

Select 3 options

A. `new Long(mStr);`

B. `Long.parseLong(mStr);`

C. `Long.longValue(mStr);`

D. `(new Long()).parseLong(mStr);`

E. `Long.valueOf(mStr).longValue();`

Check Answer

24. QID - 2.950

Consider the following method -

```
public float parseFloat( String s ){
   float f = 0.0f;
   try{
      f = Float.valueOf( s ).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      f = Float.NaN ;
      return f;
   }
   finally{
      f = 10.0f;
      return f;
   }
}
```

What will it return if the method is called with the input "0.0" ?

Select 1 option

A. It will not compile.

B. It will return 10.0

C. It will return Float.Nan

D. It will return 0.0

E. None of the above.

Check Answer

25. QID - 2.1072

What, if anything, is wrong with the following code?

```
// Filename: TestClass.java
class TestClass implements T1, T2{
   public void m1(){}
}
interface T1{
   int VALUE = 1;
   void m1();
}
interface T2{
   int VALUE = 2;
   void m1();
}
```

Select 1 option

A. `TestClass` cannot implement them both because it leads to ambiguity.

B. There is nothing wrong with the code.

C. The code will work fine only if `VALUE` is removed from one of the interfaces.

D. The code will work fine only if `m1()` is removed from one of the interfaces.

E. None of the above.

Check Answer

26. QID - 2.1027

Note: This question may be considered too advanced for this exam. What will the following code print when run?

```
public class TestClass{
    public static Integer wiggler(Integer x){
       Integer y = x + 10;
       x++;
       System.out.println(x);
       return y;
    }

    public static void main(String[] args){
       Integer dataWrapper = new Integer(5);
       Integer value = wiggler(dataWrapper);
       System.out.println(dataWrapper+value);
    }
}
```

Select 1 option

A. 5 and 20

B. 6 and 515

C. 6 and 20

D. 6 and 615

E. It will not compile.

Check Answer

27. QID - 2.847

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        while(--k){
            System.out.println(k);
        }
    }
}
```

Select 1 option

A. 1

B. 1
0

C. 2
1

D. 2
1
0

E. It will keeping printing numbers in an infinite loop.

F. It will not compile.

Check Answer

28. QID - 2.1008

Given the following pairs of method declarations, which of the statements are true?

```
1.
void perform_work(int time){ }
int  perform_work(int time, int speed){ return time*speed ;}

2.
void perform_work(int time){ }
int  perform_work(int speed){return speed ;}

3.
void perform_work(int time){ }
void Perform_work(int time){ }
```

Select 2 options

A. The first pair of methods will compile correctly and overload the method 'perform_work'.

B. The second pair of methods will compile correctly and overload the method 'perform_work'.

C. The third pair of methods will compile correctly and overload the method 'perform_work'.

D. The second pair of methods will not compile correctly.

E. The third pair of methods will not compile correctly.

Check Answer

29. QID - 2.993

Which is the first line that will cause compilation to fail in the following program?

```
// Filename: A.java
class A{
   public static void main(String args[]){
      A a = new A();
      B b = new B();
      a = b;  // 1
      b = a;  // 2
      a = (B) b; // 3
      b = (B) a; // 4
   }
}
class B extends A { }
```

Select 1 option

A. At Line 1.

B. At Line 2.

C. At Line 3.

D. At Line 4.

E. None of the above.

Check Answer

30. QID - 2.1060

Identify the valid for loop constructs assuming the following declarations:

```
Object o = null;
Collection c = //valid collection object.
int[][] ia = //valid array
```

Select 2 options

A. `for(o : c){ }`

B. `for(final Object o2 :c){ }`

C. `for(int i : ia) { }`

D. `for(Iterator it : c.iterator()){ }`

E. `for(int i : ia[0]){ }`

Check Answer

31. QID - 2.1315

Note: This question may be considered too advanced for this exam.

Given the declaration

```
interface Worker { void perform_work(); }
```

which of the following methods/classes are valid?

Select 2 options

A.
```
Worker getWorker(int i){
   return new Worker(){   public void perform_work() {
        System.out.println(i); }    };
}
```

B.
```
Worker getWorker(final int i){
   return new Worker() {   public void perform_work() {
   System.out.println(i); }    };
}
```

C.
```
Worker getWorker(int i){
   int x = i;
   class MyWorker implements Worker  {
         public void perform_work() { System.out.println(x); }    };
   return new MyWorker();
}
```

D.
```
Worker getWorker(final int i){
   class MyWorker implements Worker {
       public void perform_work() { System.out.println(i); }    };
   return new MyWorker();
}
```

Check Answer

32. QID - 2.1058

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10, 20);  //1
   }

   public void test1(int i, int... j){ System.out.println("1"); }
   public void test1(int... i ){ System.out.println("2"); }
   public void test1(int i, int j){ System.out.println("3"); }

   public static void main(String[] args){
     new Varargs().test();
   }
}
```

What will the program print?

Select 1 option

A. 1

B. 2

C. 3

D. It will not compile.

E. Exception at runtime.

Check Answer

33. QID - 2.928

Given the following set of member declarations, which of the following is true?

```
int a;     //  (1)
static int a;     //  (2)
int f( )    { return a; }     //  (3)
static int f( ) { return a; }     //  (4)
```

Select 2 options

A. Declarations (1) and (3) cannot occur in the same class definition.

B. Declarations (2) and (4) cannot occur in the same class definition.

C. Declarations (1) and (4) cannot occur in the same class definition.

D. Declarations (2) and (3) cannot occur in the same class definition.

E. Declarations (1) and (2) cannot occur in the same class definition.

Check Answer

34. QID - 2.1222

Consider the following class hierarchy

```
class A{
   public void m1() {    }
}
class B extends A{
   public void m1() {    }
}
class C extends B{
   public void m1(){
      /*  //1
      ... lot of code.
      */
   }
}
```

Select 2 options

A. You cannot access class A's `m1()` from class C for the same object (i.e. `this`).

B. You can access class B's `m1()` using `super.m1()` from `class C`.

C. You can access `class A`'s `m1()` using `( (A) this ).m1()` from `class C`.

D. You can access `class A`'s `m1()` using `super.super.m1()` from `class C`.

Check Answer

35. QID - 2.897

Given the following code :

```
public class TestClass {

    int[][] matrix = new int[2][3];

    int a[] = {1, 2, 3};
    int b[] = {4, 5, 6};

    public int compute(int x, int y){
        //1 : Insert Line of Code here
    }

    public void loadMatrix(){
        for(int x=0; x<matrix.length; x++){
            for(int y=0; y<matrix[x].length; y++){
                //2: Insert Line of Code here
            }
        }
    }
}
```

What can be inserted at //1 and //2?

Select 1 option

A. `return a(x)*b(y);`
and
`matrix(x, y) = compute(x, y);`

B. `return a[x]*b[y];`
and
`matrix[x, y] = compute(x, y);`

C. `return a[x]*b[y];`
and
`matrix[x][y] = compute(x, y);`

D. `return a(x)*b(y);`
and
`matrix(x)(y) = compute(x, y);`

E. `return a[x]*b[y];`
and
`matrix[[x][y]] = compute(x, y);`

Check Answer

36. QID - 2.1026

Given the class

```
// Filename: Test.java
public class Test{
   public static void main(String args[]){
      for(int i = 0; i< args.length; i++){
         System.out.print("  "+args[i]);
      }
   }
}
```

Now consider the following 3 options for running the program:

```
a: java Test
b: java Test param1
c: java Test param1 param2
```

Which of the following statements are true?

Select 2 options

A. The program will throw `java.lang.ArrayIndexOutofBoundsException` on option a.

B. The program will throw `java.lang.NullPointerException` on option a.

C. The program will print `Test param1` on option b.

D. It will print `param1 param2` on option c.

E. It will not print anything on option a.

Check Answer

37. QID - 2.961

Which one of these is a proper definition of a class Car that cannot be sub-classed?

Select 1 option

A. `class Car { }`

B. `abstract class Car { }`

C. `native class Car { }`

D. `static class Car { }`

E. `final class Car { }`

Check Answer

38. QID - 2.957

What will the following program print?

```
public class TestClass{
  static int someInt = 10;
  public static void changeIt(int a){
    a = 20;
  }
  public static void main(String[] args){
    changeIt(someInt);
    System.out.println(someInt);
  }
}
```

Select 1 option

A. 10

B. 20

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

39. QID - 2.1218

Complete the code using blue labels on the right so that the output will 210.

(You may leave some blanks empty.)

public int a = a + offset;

return a; u.update(a, 111);

a = u.update(a, 111);

return; public void

```
3  public class Updater
4  {
5   [          ] update(int a, int offset)
6      {
7          [          ]
8          [          ]
9      }
10
11     public static void main(String[] args)
12     {
13         Updater u = new Updater();
14
15         int a = 99;
16
17         [          ]
18
19         System.out.println(a);
20
21     }
22 }
23
```

Check Answer

40. QID - 2.967

What will the following code print when compiled and run?

```
abstract class Calculator{
   abstract void calculate();
   public static void main(String[] args){
      System.out.println("calculating");
      Calculator x = null;
      x.calculate();
   }
}
```

Select 1 option

A. It will not compile.

B. It will not print anything and will throw `NullPointerException`

C. It will print `calculating` and then throw `NullPointerException`.

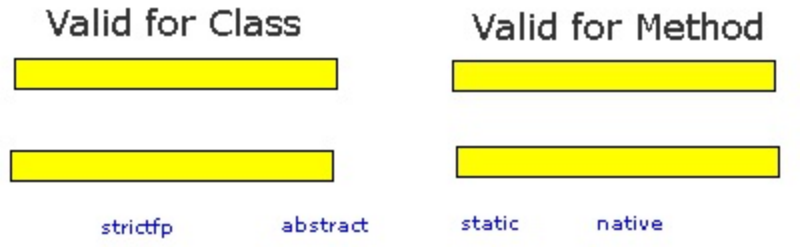
D. It will print `calculating` and will throw `NoSuchMethodError`

E. It will print `calculating` and will throw `MethodNotImplementedException`

Check Answer

41. QID - 2.1139

Idenfity valid modifiers for a top level class and method declarations.
(Assume that the class is not declared inside another class.)



Check Answer

42. QID - 2.1231

Given the following interface definition, which definitions are valid?

```
interface I1{
   void setValue(String s);
   String getValue();
}
```

Select 2 options

A.
```
class A extends I1{
   String s;
   void setValue(String val) { s = val; }
   String getValue() { return s; }
}
```

B.
```
interface I2 extends I1{
   void analyse();
}
```

C.
```
abstract class B implements I1{
   int getValue(int i) { return 0; }
}
```

D.
```
interface I3 implements I1{
   void perform_work();
}
```

Check Answer

43. QID - 2.830

Consider the following method :

```
public void myMethod(int m, Object p, double d){
  ... valid code here
}
```

Assuming that there is no other method with the same name, which of the following options are correct regarding the above method?

Select 1 option

A. If this method is called with two parameters, the value of d in the method will be 0.0.

B. If this method is called with one parameter, the value of p and d in the method will be null and 0.0 respectively.

C. If this method is called with one parameter, the call will throw a NullPointerException.

D. If this method is called with one parameter, the call will throw a NullPointerException only if the code in the method tries to access p.

E. If this method is called with two parameters, the code will not compile.

Check Answer

44. QID - 2.1099

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int x  = 0;
      labelA:   for (int i=10; i<0; i--){
         int j = 0;
         labelB:
         while (j < 10){
            if (j > i) break labelB;
            if (i == j){
               x++;
               continue labelA;
            }
            j++;
         }
         x--;
      }
      System.out.println(x);
   }
}
```

Select 1 option

A. It will not compile.

B. It will go in infinite loop when run.

C. The program will write 10 to the standard output.

D. The program will write 0 to the standard output.

E. None of the above.

Check Answer

45. QID - 2.1094

What will the following program print when run?

```
public class TestClass{
  public static void main(String[] args){
     try{
        System.exit(0);
     }
     finally{
         System.out.println("finally is always executed!");
     }
  }
}
```

Select 1 option

A. It will print "finally is always executed!"

B. It will not compile as there is no catch block.

C. It will not print anything.

D. An exception will be thrown

E. None of the above.

Check Answer

46. QID - 2.1031

Identify the exceptions that will be received when the code snippets on the left hand side are executed.

1.
```
String s = null;
System.out.println(s.length());
```

2.
```
int[] ia = new int[]{ 1, 2, 3};
System.out.println(ia[3]);
```

3.
```
Class.forName("java.lang.String");
```

4.
```
public class X {
static {
throw new NullPointerException();
}
}
```

ArrayIndexOutOfBoundsException

ExceptionInInitializerError

ClassNotFoundException

NullPointerException

Will Not Compile.

No Exception Will Be Thrown.

Check Answer

47. QID - 2.1028

Identify the exceptions that SHOULD be thrown in the situations shown below.

```
1.
public void closeReportManager(ReportManager rep)
{
 if(!rep.isClosed()) report.close();
 else throw new [          ]
}


2.
List validFormats = Arrays.asList("HTML", "JAVA", "JSP");
public void reformat(String format)
{
 if(validFormats.contains(format)) applyFormatting(format);
 else throw new [          ]
}
```

NullPointerException(); IllegalStateException();

IllegalArgumentException(); Exception();

Check Answer

48. QID - 2.1321

Consider the following class :

```
public class Parser{
   public static void main( String[] args){
       try{
           int i = 0;
           i =  Integer.parseInt( args[0] );
       }
       catch(NumberFormatException e){
          System.out.println("Problem in " + i );
       }
   }
}
```

What will happen if it is run with the following command line:

```
java Parser one
```

Select 1 option

A. It will print `Problem in 0`

B. It will throw an exception and end without printing anything.

C. It will not even compile.

D. It will not print anything if the argument is '1' instead of 'one'.

E. None of the above.

Check Answer

49. QID - 2.1240

Given:

```
byte b = 1;
char c = 1;
short s = 1;
int i = 1;
```

which of the following expressions are valid?

Select 3 options

A. `s = b * b ;`

B. `i = b << b ;`

C. `s <<= b ;`

D. `c = c + b ;`

E. `s += i ;`

Check Answer

50. QID - 2.941

What will the following code print?

```
String abc = "";
abc.concat("abc");
abc.concat("def");
System.out.print(abc);
```

Select 1 option

A. abc

B. abcdef

C. def

D. It will print empty string (or in other words, nothing).

E. It will not compile because there is no concat() method in String class.

Check Answer

51. QID - 2.1203

A method with no access modifier can be overridden by a method marked protected.

Select 1 option

A. True

B. False

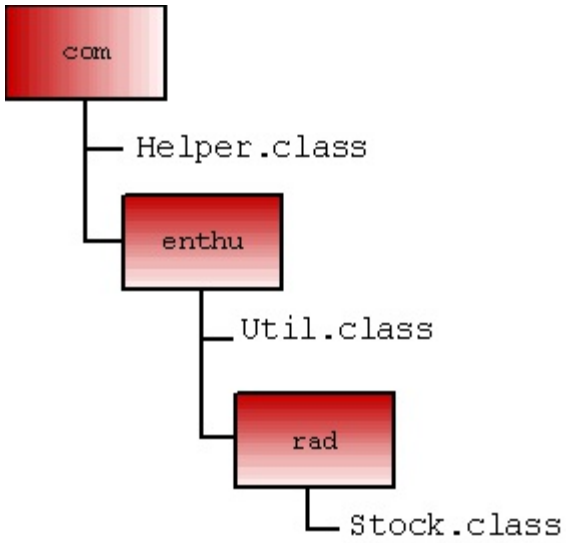
Check Answer

52. QID - 2.1063

Consider the directory structure shown in Image 1 that displays available folders and classes and the code given below:

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public void store() throws IOException{
    return Util.store(stock);
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements MUST be added to the above class?



Select 4 options

A. `package com.enthu.rad.*;`

B. `import com.enthu.*;`

C. `package com.enthu.rad;`

D. `import com.*;`

E. `import java.io.*;`

F. It is not required to import java.io.* or import java.io.IOException because java.io package is imported automatically.

Check Answer

53. QID - 2.1030

Identify the exceptions that SHOULD be thrown in the situations shown below.

(Assume that CLUBS, DIAMONDS, HEARTS, SPADES are valid elements of an enum.)

```
1.
switch(suit) {
 case CLUBS:
 ...
 break;
 case DIAMONDS:
  ...
 break;
 case HEARTS:
 ...
 break;
 case SPADES:
 ...
 break;
 default : throw new [          ]
}

2.
public void applyCode(String code)
{
 if(code.startsWith("XA")) apply(code.substring(2));
 else throw new [          ]
}
```

AssertionError(); IllegalStateException(); IllegalArgumentException(); Exception();

Check Answer

54. QID - 2.1273

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      A o1 = new C( );
      B o2 = (B) o1;
      System.out.println(o1.m1( ) );
      System.out.println(o2.i );
   }
}
class A { int i = 10;  int m1( ) { return i; } }
class B extends A { int i = 20;  int m1() { return i; } }
class C extends B { int i = 30;  int m1() { return i; } }
```

Select 1 option

A. The program will fail to compile.

B. Class cast exception at runtime.

C. It will print 30, 20.

D. It will print 30, 30.

E. It will print 20, 20.

Check Answer

55. QID - 2.1059

Identify valid for constructs...
Assume that Math.random() returns a double between 0.0 and 1.0 (not including 1.0).

Select 3 options

A.
```
for(;Math.random()<0.5;){
    System.out.println("true");
}
```

B.
```
for(;;Math.random()<0.5){
    System.out.println("true");
}
```

C.
```
for(;;Math.random()){
    System.out.println("true");
}
```

D.
```
for(;;){
    Math.random()<.05? break : continue;
}
```

E.
```
for(;;){
    if(Math.random()<.05) break;
}
```

Check Answer

56. QID - 2.981

For object o1 of class A to access a member(field or method) of object o2 of class B, when the member has no access modifier, class B must be...

Select 1 option

A. a subclass of A

B. in the same package as A is in.

C. a Super class of A

D. a subclass but may not be in the same package.

E. in the same package and must be a Subclass of A.

Check Answer

57. QID - 2.1108

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int k = 9, s = 5;
      switch(k){
         default :
         if( k == 10) { s = s*2; }
         else{
            s = s+4;
            break;
         }
         case 7 : s = s+3;
      }
      System.out.println(s);
   }
}
```

Select 1 option

A. 5

B. 9

C. 12

D. It will not compile.

Check Answer

58. QID - 2.1284

What will the following class print when run?

```
public class Sample{
   public static void main(String[] args)  {
     String s1 = new String("java");
     StringBuilder s2 = new StringBuilder("java");
     replaceString(s1);
     replaceStringBuilder(s2);
     System.out.println(s1 + s2);
  }
  static void replaceString(String s) {
     s = s.replace('j', 'l');
  }
  static void replaceStringBuilder(StringBuilder s) {
     s.append("c");
  }
}
```

Select 1 option

A. `javajava`

B. `lavajava`

C. `javajavac`

D. `lavajavac`

E. None of these.

Check Answer

59. QID - 2.1283

What is the correct declaration for an abstract method 'add' accessible to any class, takes no arguments and returns nothing?
(Use only one space between words)

Select 1 option

A. `public void add();`

B. `abstract add();`

C. `abstract null add();`

D. `abstract public void add(){ }`

E. `abstract public void add() throws Exception;`

Check Answer

60. QID - 2.1143

Which of the given statements are correct for a method that overrides the following method:

```
public Set getSet(int a) {...}
```

Select 3 options

A. Its return type must be declared as `Set`.

B. It may return HashSet.

C. It can declare any Exception in throws clause

D. It can declare any RuntimeException in throws clause.

E. It can be abstract.

Check Answer

61. QID - 2.1292

Given the definitions of I and Klass, complete the definition of SubClass so that it extends from Klass and implements I.
Use minimum number of elements.

```
interface I
{
    void m1();
}

abstract class Klass
{
    void m1() { };
}

class SubClass [ ] [ ] [ ] [ ]
{
    [ ] [ ] [ ] [ ] [ ] [ ]
}
```

extends implements Klass

I public void m1(){ }

Check Answer

62. QID - 2.1163

The following is a valid member variable declaration:

```
private static final transient int i = 20;
```

Select 1 option

A. True

B. False

Check Answer

63. QID - 2.902

Given:

```
class Triangle{
    public int base;
    public int height;
    public double area = 0;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }
    public void updateArea(){
        double a = base*height/2;
        area = a;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

Which variables are not accessible from anywhere within given class code (except from where they are declared)?

Select 1 option

A. base, height, area

B. area, b, h

C. base, height

D. b, h, a

Check Answer

64. QID - 2.1266

Which of the following statements are true?

Select 3 options

A. The condition expression in an if statement can contain method calls.

B. If a and b are of type boolean, the expression (a = b) can be used as the condition expression of an if statement.

C. An if statement can have either an 'if' clause or an 'else' clause.

D. The statement : if (false) ; else ; is illegal.

E. Only expressions which evaluate to a boolean value can be used as the condition in an if statement.

Check Answer

65. QID - 2.1330

Which of the following access control keywords can be used to enable all the subclasses to access a method defined in the base class?

Select 2 options

A. public

B. private

C. protected

D. No keyword is needed.

Check Answer

66. QID - 2.1184

Given:

```
StringBuilder b1 = new StringBuilder("snorkler");
StringBuilder b2 = new StringBuilder("yoodler");
```

Write the contents of b1 and b2 after the statements shown on the left are executed independent of each other.

Statements	Contents of b1	Contents of b2
`b1.append(b2.substring(2, 5) .toUpperCase());`		
`b2.insert(3, b1.append("a"));`		
`b1.replace(3, 4, b2.substring(4)) .append(b2.append(false));`		

Check Answer

67. QID - 2.1089

Drag and drop valid operators (shown in blue) in yellow boxes.
= operator can be used more than once.

`=` `+=` `*=`

```
public class TestClass
{
   public static void main(String[] args)
   {
     Short k = 9;     Integer i = 9;  Boolean b = false;
     char c = 'a';    String str = "123";

     i      [ ]      (int) k.shortValue();

     str    [ ]      b;

     b      [ ]      !b;

     c      [ ]      i;

   }
}
```

Check Answer

68. QID - 2.1004

Which of these statements concerning the use of modifiers are true?

Select 1 option

A. By default (i.e. no modifier) the member is only accessible to classes in the same package and subclasses of the class.

B. You cannot specify visibility of local variables.

C. Local variable always have default accessibility.

D. Local variables can be declared as private.

E. Local variables can only be declared as public.

Check Answer

69. QID - 2.1254

What will the following code print when run?

```
public class Test{
 static String j = "";
 public static void method( int i){
  try{
   if(i == 2){
     throw new Exception();
   }
   j += "1";
  }
  catch (Exception e){
   j += "2";
   return;
  }
  finally{
   j += "3";
  }
  j += "4";
 }
 public static void main(String args[]){
  method(1);
  method(2);
  System.out.println(j);
 }
}
```

Select 1 option

A. 13432

B. 13423

C. 14324

D. 12434

E. 12342

Check Answer

70. QID - 2.963

Consider the following code snippet:

```
for(int i=INT1; i<INT2; i++){
    System.out.println(i);
}
```

INT1 and INT2 can be any two integers.

Which of the following will produce the same result?

Select 1 option

A. `for(int i=INT1; i<INT2; System.out.println(++i));`

B. `for(int i=INT1; i++<INT2; System.out.println(i));`

C. `int i=INT1; while(i++<INT2) { System.out.println(i); }`

D. `int i=INT1; do { System.out.println(i); }while(i++<INT2);`

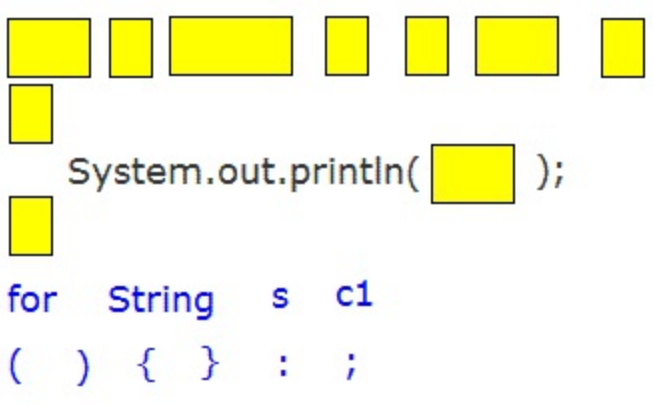
E. None of these.

Check Answer

71. QID - 2.1043

Assuming the following declarations, write a for loop that prints each string in the collection using the given elements.

```
Collection<String> c1 = new HashSet<String>();
```



Check Answer

72. QID - 2.1131

What would be the result of attempting to compile and run the following program?

```
class TestClass{
   static TestClass ref;
   String[] arguments;
   public static void main(String args[]){
      ref = new TestClass();
      ref.func(args);
   }
   public void func(String[] args){
      ref.arguments = args;
   }
}
```

Select 1 option

A. The program will fail to compile, since the static method `main` is trying to call the non-static method `func`.

B. The program will fail to compile, since the non-static method `func` cannot access the static member variable `ref`.

C. The program will fail to compile, since the argument `args` passed to the static method `main` cannot be passed on to the non-static method `func`.

D. The program will fail to compile, since method `func` is trying to assign to the non-static member variable '`arguments`' through the static member variable `ref`.

E. The program will compile and run successfully.

Check Answer

73. QID - 2.1214

Consider the following program:

```
public class TestClass{
  public static void main(String[] args){
    String tom = args[0];
    String dick = args[1];
    String harry = args[2];
  }
}
```

What will the value of 'harry' if the program is run from the command line:
java TestClass 111 222 333

Select 1 option

A. 111

B. 222

C. 333

D. It will throw an `ArrayIndexOutOfBoundsException`

E. None of the above.

Check Answer

74. QID - 2.943

Note: Option 4 of this question may be considered too advanced for this exam.

Which lines of code will not be acceptable to the compiler?

```
import java.*; //1
public abstract class InternalLogic //2
{
   float density = 20.0; //3
   public class Doer //4
   {
      void do() //5
      {
         //lot of valid code.
      }
   }
}
```

Select 2 options

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

75. QID - 2.966

Select the correct order of restrictiveness for access modifiers...
(First one should be least restrictive)

Select 1 option

A. public < protected < package (i.e. no modifier) < private

B. public < package (i.e. no modifier) < protected < private

C. public < protected < private < package (i.e. no modifier)

D. protected < package (i.e. no modifier) < private < public

E. depends on the implementation of the class or method.

Check Answer

76. QID - 2.1080

Which of the following are valid declarations of the standard main() method?

Select 2 options

A. `static void main(String args[ ]) { }`

B. `public static int main(String args[ ]) {}`

C. `public static void main (String args) { }`

D. `final static public void main (String[ ] arguments ) { }`

E. `public static void main (String[ ] args) { }`

Check Answer

77. QID - 2.1070

What is the result of executing the following fragment of code:

```
boolean b1 = false;
int i1 = 2;
int i2 = 3;
if (b1 = i1 == i2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print true

C. It will print false

D. Runtime error.

E. It will print nothing.

Check Answer

78. QID - 2.880

Which of the following are standard Java exception classes?

Select 2 options

A. `FileNotFoundException`

B. `InputException`

C. `CPUError`

D. `MemoryException`

E. `SecurityException`

Check Answer

79. QID - 2.1103

What will be printed by the following code if it is run with command line: java TestClass -0.50 ?

```
public class TestClass{
    public static double getSwitch(String str){
        return Double.parseDouble(str.substring(1, str.length()-1)
);
    }
    public static void main(String args []){
        switch(getSwitch(args[0])){
            case 0.0 : System.out.println("Hello");
            case 1.0 : System.out.println("World"); break;
            default : System.out.println("Good Bye");
        }
    }
}
```

Select 1 option

A. Hello

B. World

C. Hello World

D. Hello World Good Bye

E. None of the above.

Check Answer

80. QID - 2.912

Which of the following methods does not return any value?

Select 1 option

A.
```
public doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

B.
```
public null doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

C.
```
public doStuff() {
    //valid code not shown
}
```

D.
```
public void doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

E.
```
private doStuff() {
    //valid code not shown
}
```

Check Answer

81. QID - 2.1023

Identify the exceptions that will be received when the code snippets on the left hand side are executed.

1.
```
int[] ia = new int[]{ 1, 2, 3};
System.out.println(ia[-1]);
```

2.
```
public class X {
static int k = 0;
static {
k = 10/0;
}
}
```

3.
```
SomeClass sc = new SomeClass();
```
(Assume that SomeClass is not available in runtime classpath.)

4.
```
public class X {
static {
if(true) throw new NullPointerException();
}
}
```

NoClassDefFoundError NullPointerException ArrayAccessException

ExceptionInInitializerError ArrayIndexOutOfBoundsException

IllegalArrayAccessException NoSuchClassException

Check Answer

82. QID - 2.1250

Which statements, when inserted at line 1, will cause an exception at run time?

```
class B {}
class B1 extends B {}
class B2 extends B {}
public class ExtendsTest{
  public static void main(String args[]){
     B b = new B();
     B1 b1 = new B1();
     B2 b2 = new B2();
     // insert statement here
  }
}
```

Select 1 option

A. `b = b1;`

B. `b2 = b;`

C. `b1 = (B1) b;`

D. `b2 = (B2) b1;`

E. `b1 = (B) b1;`

Check Answer

83. QID - 2.976

Consider the following two java files:

```
//in file SM.java
package x.y;
public class SM{
    public static void foo(){ };
}


//in file TestClass.java
//insert import statement here //1
public class TestClass{
   public static void main(String[] args){
      foo();
   }
}
```

What should be inserted at //1 so that TestClass will compile and run?

Select 2 options

A. `import static x.y.*;`

B. `import static x.y.SM;`

C. `import static x.y.SM.foo;`

D. `import static x.y.SM.foo();`

E. `import static x.y.SM.*;`

Check Answer

84. QID - 2.1259

Note: Although Wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of Wrapper classes.

What will be the output of the following program?

```
public class EqualTest{
   public static void main(String args[]){
      Integer i = new Integer(1) ;
      Long m = new Long(1);
      if( i.equals(m)) System.out.println("equal");    // 1
      else System.out.println("not equal");
   }
}
```

Select 1 option

A. equal

B. not equal

C. Compile time error at //1

D. Runtime error at //1

E. None of the above.

Check Answer

85. QID - 2.931

Consider the following classes :

```
class A{
    public static void sM1() {  System.out.println("In base static"),
}
class B extends A{
Line 1 :   // public static void sM1() {  System.out.println("In sub
Line 2 :   // public  void sM1() {  System.out.println("In sub non-s
}
```

Which of the following statements are true?

Select 2 options

A. class B will not compile if line 1 is uncommented.

B. class B will not compile if line 2 is uncommented.

C. class B will not compile if line 1 and 2 are both uncommented.

D. Only Option 2 is correct.

E. Only Option 3 is correct.

Check Answer

86. QID - 2.1233

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        boolean flag = true;
        for(int i = 0; i < 3; i++){
            while(flag){
                c++;
                if(i>c || c>5) flag = false;
            }
        }
        System.out.println(c);
    }
}
```

Select 1 option

A. 3

B. 4

C. 5

D. 6

E. 7

Check Answer

87. QID - 2.1217

Consider the following classes in one file named A.java...

```
abstract class A{
   protected int m1(){ return 0; }
}
class B extends A{
   int m1(){ return 1; }
}
```

Which of the following statements are correct...

Select 1 option

A. The code will not compile as you cannot have more than 1 class in 1 file.

B. The code will not compile because class B does not override the method m1() correctly.

C. The code will not compile as A is an abstract class.

D. The code will not compile as A does not have any abstract method.

E. The code will compile fine.

Check Answer

88. QID - 2.946

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
    public static void main(String args[ ] ){
       int i = 1;
       int[] iArr = {1};
       incr(i) ;
       incr(iArr) ;
       System.out.println( "i = " + i + "  iArr[0] = " + iArr [ 0 ] );
    }
    public static void incr(int   n ) { n++ ; }
    public static void incr(int[ ] n ) { n [ 0 ]++ ; }
}
```

Select 1 option

A. `The code will print i = 1 iArr[0] = 1;`

B. `The code will print i = 1 iArr[0] = 2;`

C. `The code will print i = 2 iArr[0] = 1;`

D. `The code will print i = 2 iArr[0] = 2;`

E. The code will not compile.

Check Answer

89. QID - 2.827

What will be the output when the following program is run?

```
package exceptions;
public class TestClass{
    public static void main(String[] args) {
        try{
            hello();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void hello() throws MyException{
        int[] dear = new int[7];
        dear[0] = 747;
        foo();
    }

    static void foo() throws MyException{
        throw new MyException("Exception from foo");
    }
}

class MyException extends Exception {
    public MyException(String msg){
        super(msg);
    }
}
```

(Assume that line numbers printed in the messages given below are correct.)

Select 1 option

A. `Exception in thread "main"`
`java.lang.ArrayIndexOutOfBoundsException: 10`

```
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

B.
```
Error in thread "main" java.lang.ArrayIndexOutOfBoundsException
```

C.
```
exceptions.MyException: Exception from foo
```

D.
```
exceptions.MyException: Exception from foo
    at exceptions.TestClass.foo(TestClass.java:29)
    at exceptions.TestClass.hello(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

Check Answer

90. QID - 2.1181

What would be the result of attempting to compile and run the following code?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      B c = new C();
      System.out.println(c.max(10, 20));
   }
}
class A{
   int max(int x, int y)  { if (x>y) return x; else return y; }
}
class B extends A{
  int max(int x, int y)  {  return 2 * super.max(x, y) ; }
}
class C extends B{
  int max(int x, int y)  {  return super.max( 2*x, 2*y); }
}
```

Select 1 option

A. The code will fail to compile.

B. Runtime error.

C. The code will compile without errors and will print 80 when run.

D. The code will compile without errors and will print 40 when run.

E. The code will compile without errors and will print 20 when run.

Check Answer

Test 5 (Answered)

This test contains a few questions on certain aspects of inner classes and primitive wrapper, which may be too advanced for this certification. These questions are labelled accordingly in their problem statements. You may ignore these questions if you are short on time.

01. QID - 2.1120 : Using Loop Constructs

What will the following program snippet print?

```
int i=0, j=11;
do{
   if(i > j) { break; }
   j--;
}while( ++i < 5);
System.out.println(i+"  "+j);
```

Correct Option is : B

~~**A.**~~ 5 5

B. 5 6

~~**C.**~~ 6 6

~~**D.**~~ 6 5

~~**E.**~~ 4 5

Explanation:

++i < 5 means, increment the value of i and then compare with 5.
Now, Try to work out the values of i and j at every iteration.
To start with, i=0 and j=11. At the time of evaluation of the while condition, i and j are as follows:

1. j = 10 and i=1 (loop will continue because i<5) (Remember that comparison will happen AFTER increment i because it is ++i and not i++.
2. j = 9 and i=2 (loop will continue because i<5).
3. j = 8 and i=3 (loop will continue because i<5).
4. j = 7 and i=4 (loop will continue because i<5).
5. j = 6 and i=5 (loop will NOT continue because i not <5).

So it will print 5 6. (It is print i first and then j).

Back to Question without Answer

02. QID - 2.1122 : Java Basics

Note: This question may be considered too advanced for this exam. What will the code shown below print when compiled and run with the following command line?

```
java WarZone self
```

```
interface XMen {
   void shoot(String a);
}

public class WarZone {
   public static void main(String[] args){
       XMen x = null;
       if(args.length() > 0){
          x = new XMen(){
             public void shoot(String s){
                for(int i=0; i<s.length; i++){
                  System.out.println("shot : "+s.charAt(i));
                }
             }
          };
       }

       if(x != null){
          x.shoot(args[0]);
       }
   }
}
```

Correct Option is : E

~~**A.**~~ It will not compile because interface XMen cannot be instantiated.

An anonymous inner class that implements XMen interface is being created.

~~**B.** It will print `shot :` 4 times, one at each line.~~

~~**C.** It will print "`shot : s`", "`shot : e`", "`shot : l`", "`shot : f`" one by one on 4 lines.~~

> This would be correct, if `args.length()` is changed to `args.length` and `s.length` to `s.length()`.

~~**D.** It will compile but will throw an exception at runtime.~~

E. None of these options is correct.

> Read the question carefully. 'args' is an array and length is an attribute (not a method) of arrays. 's' is a String and there is no attribute length in a String but a method length(). Expect questions that try to confuse by adding misleading statements.

Back to Question without Answer

03. QID - 2.1018 : Constructors

Note: This question may be considered too advanced for this exam. Given:

```
class MySuper{
    public MySuper(int i){ }
}

abstract class MySub extends MySuper{
    public MySub(int i){ super(i); }
    public abstract void m1();
}

class MyTest{
    public static void main(String[] args){
        MySub ms = new MySub(){
            public void m1() { System.out.println("In MySub.m1()");
        };
        ms.m1();
    }
}
```

What will be the output when the above code is compiled and run?

Correct Option is : A

A. It will not compile.

When you define and instantiate an anonymous inner class for an abstract class as being done here, the anonymous class is actually a subclass of the abstract class (in this case, it is a subclass of MySub). Now, since the anonymous class does not define any constructor, the compiler will add the default no-args constructor to the anonymous class, which will try to call the no-args constructor of its super class MySub. But MySub doesn't have any no-args constructor. Therefore, it will not compile. The anonymous inner class should have been created like this:

```
MySub ms = new MySub( someInteger ){ ... };
```

~~**B.**~~ It will throw an exception at run time.

~~**C.**~~ It will print `In MySub.m1()`

~~**D.**~~ It will compile and run without any exception but will not print anything.

Back to Question without Answer

04. QID - 2.1003 : Working with Inheritance

Given the following code, which statements are true?

```
interface Automobile { String describe(); }

class FourWheeler implements Automobile{
   String name;
   public String describe(){ return " 4 Wheeler " + name; }
}

class TwoWheeler extends FourWheeler{
    String name;
    public String describe(){ return " 2 Wheeler " + name; }
}
```

Correct Options are : A B C

A. An instance of `TwoWheeler` is also an instance of `FourWheeler`.

B. An instance of `TwoWheeler` is a valid instance of `Automobile`.

C. The use of inheritance is not justified here because a `TwoWheeler` is not really a `FourWheeler`.

~~**D.** The code will compile only if `name` is removed from `TwoWheeler`.~~

~~**E.** The code will fail to compile.~~

Explanation:

The use of inheritance in this code is not justifiable, since conceptually, a `TwoWheeler` is-not-a `FourWheeler`.

Back to Question without Answer

05. QID - 2.1252 : Working with Methods

What will be the output when the following program is run?

```
public class TestClass{
    char c;
    public void m1(){
        char[ ] cA = { 'a' , 'b'};
        m2(c, cA);
        System.out.println( ( (int)c)  + ", " + cA[1] );
    }
    public void m2(char c, char[ ] cA){
        c = 'b';
        cA[1] = cA[0] = 'm';
    }
    public static void main(String args[]){
        new TestClass().m1();
    }
}
```

Correct Option is : C

~~**A.**~~ Compile time error.

> c is an instance variable of numeric type so it will be given a default value of 0, which prints as empty space.

~~**B.**~~ , m

> Without the cast to int, c would be printed as empty space and cA[1] is 'm'

C. 0 , m

> Because of the explicit cast to int in the println() call, c will be printed as 0.

~~**D.**~~ b , b

~~**E.**~~ b , m

Explanation:

Note that Arrays are Objects (i.e. `cA instanceof Object` is `true`) so are effectively passed by reference. So in `m1()` the change in `cA[1]` done by `m2()` is reflected everywhere the array is used.
`c` is a primitive type and is passed by value. In method `m2()` the passed parameter c is different than instance variable 'c' as local variable hides the instance variable. So instance member 'c' keeps its default (i.e. 0) value.

Back to Question without Answer

06. QID - 2.1106 : Creating and Using Arrays

Is it possible to create arrays of length zero?

Correct Option is : A

A. Yes, you can create arrays of any type with length zero.

> Java allows arrays of length zero to be created.

~~**B.**~~ Yes, but only for primitive datatypes.

~~**C.**~~ Yes, but only for arrays of object references.

~~**D.**~~ Yes, and it is same as a null Array.

> No. A null pointer is different from an array of length Zero. For example, if you have `int[] intArr = new int[0];` then `(intArr == null)` is `false`.

~~**E.**~~ No, arrays of length zero do not exist in Java.

Explanation:

Example: When a Java program is run without any program arguments, the `String[] args` argument to `main()` gets an array of length Zero.

Back to Question without Answer

07. QID - 2.1260 : Handling Exceptions

What will be the output of the following class...

```
class Test{
   public static void main(String[] args){
      int j = 1;
      try{
         int i = doIt() / (j = 2);
      } catch (Exception e){
         System.out.println(" j = " + j);
      }
   }
   public static int doIt() throws Exception {  throw new Exception('
}
```

Correct Option is : A

A. `It will print j = 1;`

B. `It will print j = 2;`

C. The value of j cannot be determined.

D. It will not compile.

E. None of the above.

Explanation:

If evaluation of the left-hand operand of a binary operator completes abruptly, no part

of the right-hand operand appears to have been evaluated.
So, as doIt() throws exception, j = 2 never gets executed.

Back to Question without Answer

08. QID - 2.1353 : Using Operators and Decision Constructs

Which of the lines will cause a compile time error in the following program?

```
public class MyClass{
   public static void main(String args[]){
      char c;
      int i;
      c = 'a';//1
      i = c;  //2
      i++;    //3
      c = i;  //4
      c++;    //5
   }
}
```

Correct Option is : D

~~**A.**~~ line 1

~~**B.**~~ line 2

~~**C.**~~ line 3

D. line 4

~~**E.**~~ line 5

Explanation:

1. A char value can ALWAYS be assigned to an int variable, since the int type is

wider than the char type. So line 2 is valid.

2. Line 4 will not compile because it is trying to assign an int to a char. Although the value of i can be held by the char but since 'i' is not a constant but a variable, implicit narrowing will not occur.

Here is the rule given in JLS:
A narrowing primitive conversion may be used if all of the following conditions are satisfied:
The expression is a constant expression of type int.
The type of the variable is byte, short, or char.
The value of the expression (which is known at compile time, because it is a constant expression) is representable in the type of the variable.
Note that narrowing conversion does not apply to long or double.
so, char ch = 30L; will fail although 30 is representable by a char.

Back to Question without Answer

09. QID - 2.1097 : Handling Exceptions

What will be the output of the following program?

```
class TestClass{
   public static void main(String[] args) throws Exception{
      try{
         amethod();
         System.out.println("try");
      }
      catch(Exception e){
         System.out.println("catch");
      }
      finally   {
         System.out.println("finally");
      }
      System.out.println("out");
   }
   public static void amethod(){ }
}
```

Correct Option is : B

A. `try finally`

B. `try finally out`

C. `try out`

D. `catch finally out`

E. It will not compile because `amethod()` does not throw any exception.

Explanation:

Since the method `amethod()` does not throw any exception, `try` is printed and the control goes to finally which prints `finally`. After that `out` is printed.

Back to Question without Answer

10. QID - 2.1074 : Working with Methods

Consider the following class definition:

```
public class TestClass{
   public static void main(){  new TestClass().sayHello(); }   //1
   public static void sayHello(){ System.out.println("Static Hello W
   public void sayHello() { System.out.println("Hello World "); }  /
}
```

What will be the result of compiling and running the class?

Correct Option is : D

~~**A.**~~ It will print `Hello World`.

~~**B.**~~ It will print `Static Hello World`.

~~**C.**~~ Compilation error at line 2.

D. Compilation error at line 3.

It will say, method sayHello() is already defined.

~~**E.**~~ Runtime Error.

Explanation:

You cannot have two methods with the same signature (name and parameter types) in one class.
Also, even if you put one `sayHello()` method in other class which is a subclass of

this class, it won't compile because you cannot override/hide a static method with a non static method and vice versa.

Back to Question without Answer

11. QID - 2.956 : Working with Java Data Types - String, StringBuilder

Consider the following class...

```
class MyString extends String{
   MyString(){ super(); }
}
```

The above code will not compile.

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

This will not compile because `String` is a final class and final classes cannot be extended.
There are questions on this aspect in the exam and so you should remember that `StringBuffer` and `StringBuilder` are also final. All Primitive wrappers are also final (i.e. `Boolean, Integer, Byte` etc).
`java.lang.System` is also final.

Back to Question without Answer

12. QID - 2.1057 : Using Loop Constructs

What will the following code print?

```
void crazyLoop(){
   int c = 0;
   JACK: while (c < 8){
       JILL: System.out.println(c);
       if (c > 3) break JILL; else c++;
   }
}
```

Correct Option is : A

A. It will not compile.

> Because `break JILL;` would be valid only when it is within the block of code under the scope of the label `JILL`.
> In this case, the scope of `JILL` extends only up till `System.out.println(c);` and `break JILL;` is out of the scope of the label.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ It will print numbers from 0 to 8

~~**D.**~~ It will print numbers from 0 to 3

~~**E.**~~ It will print numbers from 0 to 4

Back to Question without Answer

13. QID - 2.1156 : Using Operators and Decision Constructs

What will the following class print when executed?

```
class Test{
    static boolean a;
    static boolean b;
    static boolean c;
    public static void main (String[] args){
        boolean bool = (a = true) || (b = true) && (c = true);
        System.out.print(a + ", " + b + ", " + c);
    }
}
```

Correct Option is : C

~~**A.**~~ true, false, true

~~**B.**~~ true, true, false

C. true, false, false

~~**D.**~~ true, true, true

Explanation:

Java parses the expression from left to right. Once it realizes that the left operand of a conditional "or" operator has evaluated to true, it does not even try to evaluate the right side expression.

Back to Question without Answer

14. QID - 2.1249 : Java Basics

Consider the following class:

```
public class ArgsPrinter{
   public static void main(String args){
      for(int i=0; i<3; i++){
         System.out.print(args+" ");
      }
   }
}
```

What will be printed when the above class is run using the following command line:
`java ArgsPrinter 1 2 3 4`

Correct Option is : E

~~**A.**~~ `1 2 3`

~~**B.**~~ `ArgsPrinter 1 2`

~~**C.**~~ `java ArgsPrinter 1 2`

~~**D.**~~ `1 1 1`

E. None of these.

Explanation:

To run a class from the command line, you need a `main(String[] )` method that takes an array of Strings array not just a String. Therefore, an exception will be thrown

at runtime saying no `main(String[] )` method found.

Back to Question without Answer

15. QID - 2.947 : Constructors

Which line contains a valid constructor in the following class definition?

```
public class TestClass{
   int i, j;
   public TestClass getInstance() {  return new TestClass();     }   /,
   public void TestClass(int x, int y) {    i = x;    j = y;    }      /,
   public TestClass TestClass() {    return new TestClass();    }     /,
   public ~TestClass() {     }                      //4
}
```

Correct Option is : E

~~**A.**~~ Line 1

> This cannot be a constructor because even the name of the method (getInstance) is not same as the class name!

~~**B.**~~ Line 2

> Constructors cannot return anything. Not even void.

~~**C.**~~ Line 3

> Constructors cannot return anything. Not even void.

~~**D.**~~ Line 4

> This could have been a destructor in C++ world. And there nothing like this in java. Java has a finalize() method, which is similar to a destructor but does not work exactly as a destructor.

E. None of the above.

Back to Question without Answer

16. QID - 2.893 : Working with Methods

Given:

```
interface Worker {
   void  performWork();
}

class FastWorker  implements Worker {
   public void  performWork(){ }
}
```

You are creating a class that follows "program to an interface" principle. Which of the following line of code will you most likely be using?

Correct Option is : C

~~A.~~ `public FastWorker getWorker(){ return new Worker(); }`

> This will not compile because `Worker` is an interface and so it cannot be instantiated. Further, a `Worker` is not `FastWorker`. A `FastWorker` is a `Worker`.

~~B.~~ `public FastWorker getWorker(){ return new FastWorker(); }`

C. `public Worker getWorker(){ return new FastWorker(); }`

> This is correct because the caller of this method will not know about the actual class of the object that is returned by this method. It is only aware of the `Worker`

interface. Hence, if you change the implementation of this method to return a different type of Worker, say `SuperFastWorker`, other classes do not have to change their code.

~~**D.**~~
```
public Worker getWorker(){
return new Worker();
}
```

This will not compile because `Worker` is an interface and so it cannot be instantiated.

Explanation:

Although not mentioned explicitly in the exam objectives, there are a few questions on this topic in the exam.

Back to Question without Answer

17. QID - 2.1155 : Working with Java Data Types - String, StringBuilder

Which line will print the string "MUM"?

```
public class TestClass{
    public static void main(String args []){
        String s = "MINIMUM";
        System.out.println(s.substring(4, 7)); //1
        System.out.println(s.substring(5)); //2
        System.out.println(s.substring(s.indexOf('I', 3))); //3
        System.out.println(s.substring(s.indexOf('I', 4))); //4
    }
}
```

Correct Option is : A

A. 1

~~B.~~ 2

It will print UM.

~~C.~~ 3

It will print IMUM. as s.indexOf('I', 3) will return 3.

~~D.~~ 4

It will throw an exception as s.indexOf('I', 4) will return -1.

~~E.~~ None of these.

Explanation:

You should know how substring and indexOf methods of String class work.

String substring(int beginIndex)

Returns a new string that is a substring of this string.

String substring(int beginIndex, int endIndex)

Returns a new string that is a substring of this string.

int indexOf(int ch)

Returns the index within this string of the first occurrence of the specified character.

int indexOf(int ch, int fromIndex)

Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

int indexOf(String str)

Returns the index within this string of the first occurrence of the specified substring.

int indexOf(String str, int fromIndex)

Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

Back to Question without Answer

18. QID - 2.1117 : Working with Inheritance

Given the following classes, what will be the output of compiling and running the class Truck?

```
class Automobile{
   public void drive() {  System.out.println("Automobile: drive");
}

public class Truck extends Automobile{
   public void drive() {  System.out.println("Truck: drive");   }
   public static void main (String args [ ]){
      Automobile  a = new Automobile();
      Truck t  = new Truck();
      a.drive(); //1
      t.drive(); //2
      a = t;     //3
      a.drive(); //4
   }
}
```

Correct Option is : D

~~**A.**~~ Compiler error at line 3.

~~**B.**~~ Runtime error at line 3.

~~**C.**~~ It will print:

```
Automobile: drive
Truck: drive
Automobile: drive
```

in that order.

D. It will print:

```
Automobile: drive
Truck: drive
Truck: drive
```

in that order.

~~**E.**~~ It will print:

```
Automobile: drive
Automobile: drive
Automobile: drive
```

in that order.

Explanation:

Since `Truck` is a subclass of `Automobile`, `a = t` will be valid at compile time as well runtime. But a cast is needed to make for `t = (Truck) a;` This will be ok at compile time but if at run time 'a' does not refer to an object of class Truck then a `ClassCastException` will be thrown. Now, method to be executed is decided at run time and it depends on the actual class of object referred to by the variable. Here, at line 4, variable a refers to an object of class Truck. So Truck's `drive()` will be called which prints `Truck: drive`. This is polymorphism in action!

Back to Question without Answer

19. QID - 2.860 : Using Operators and Decision Constructs

What will the following code print when run?

```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

Correct Option is : B

~~**A.** bat~~
big bat

B. big bat
none

Since there is a case condition that matches the input string "B", that case statement will be executed directly. This prints "big bat". Since there is no break after this case statement and the next case statement, the control will fall through the next one (which is default :) and so "none" will be printed as well.

Note that "b" and "B" are different strings. "B" is not equal to "b".

~~C.~~ big bat

~~D.~~ bat

~~E.~~ The code will not compile.

Explanation:

As of JDK 7 release, you can use a String object in the expression of a switch statement:

```
public String getTypeOfDayWithSwitchStatement(String dayOfWeekArg) {
     String typeOfDay;
     switch (dayOfWeekArg) {
         case "Monday":
             typeOfDay = "Start of work week";
             break;
         case "Tuesday":
         case "Wednesday":
         case "Thursday":
             typeOfDay = "Midweek";
             break;
         case "Friday":
             typeOfDay = "End of work week";
             break;
         case "Saturday":
         case "Sunday":
             typeOfDay = "Weekend";
             break;
         default:
             throw new IllegalArgumentException("Invalid day of the
     }
     return typeOfDay;
}
```

The switch statement compares the String object in its expression with the expressions

associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

Back to Question without Answer

20. QID - 2.1033 : Handling Exceptions

A try statement must always have a ..... associated with it.

Correct Option is : D

~~**A.** catch~~

~~**B.** throws~~

~~**C.** finally~~

D. catch, finally or both

~~**E.** throw~~

Explanation:

A try without resources must have either a catch or a finally. It may have both as well. Thus, the following constructs are valid:

1.

```
try{
}
catch(Exception e){  }          // no finally
```

2.

```
try{
}
finally{  }          // no catch
```

3.

```
try{
}
catch(Exception e){  }
finally{  }
```

4. A catch can catch multiple exceptions:

```
try{
}
catch(Exception1|Exception2|Exception3 e){  }
```

Note: try with resources (which is not on this exam) may omit catch as well as finally blocks.

Back to Question without Answer

21. QID - 2.1331 : Working with Java Data Types - Variables and Objects

Note: This question may be considered too advanced for this exam.

Which statements can be inserted at line 1 in the following code to make the program write x on the standard output when run?

```
public class AccessTest{
   String a = "x";
   static char b = 'x';
   String  c = "x";
   class Inner{
      String  a = "y";
      String  get(){
         String c = "temp";
         // Line 1
         return c;
      }
   }

   AccessTest() {
     System.out.println(  new Inner().get()  );
   }

   public static void main(String args[]) {  new AccessTest();  }
}
```

Correct Options are : C D E

~~**A.**~~ `c = c;`

> It will reassign 'temp' to c!

~~**B.**~~ `c = this.a;`

It will assign "y" to c.

C. `c = ""+AccessTest.b;`

Because b is static.

D. `c = AccessTest.this.a;`

E. `c = ""+b;`

Back to Question without Answer

22. QID - 2.1109 : Working with Java Data Types - String, StringBuilder

Which of the following statements are true?

Correct Options are : A D

A. method `length()` of String class is a final method.

> Actually, String class itself is final and so all of its methods are implicitly final.

~~**B.**~~ You can make mutable subclasses of the String class.

> Both - `String` and `StringBuilder` are final classes. So is `StringBuffer`.

~~**C.**~~ `StringBuilder` extends `String`.

> `StringBuilder` extends `Object`

D. StringBuilder is a final class.

> String, StringBuilder, and StringBuffer - all are final classes.
>
> 1. Remember that wrapper classes (java.lang.Boolean, java.lang.Integer, java.lang.Long, java.lang.Short etc.) are also final and so they cannot be extended.
>
> 2. java.lang.Number, however, is not final. Integer, Long, Double etc. extend Number.
>
> 3. java.lang.System is final as well.

~~**E.**~~ String class is not final.

Back to Question without Answer

23. QID - 2.1126 : Working with Java Data Types - Variables and Objects

Note: This question may be considered too advanced for this exam.
Given:

```
String mStr = "123";
long m = // 1
```

Which of the following options when put at //1 will assign `123` to `m`?

Correct Options are : A B E

A. `new Long(mStr);`

> Auto unboxing will occur.

B. `Long.parseLong(mStr);`

~~C.~~ `Long.longValue(mStr);`

> longValue is a non-static method in Long class.

~~D.~~ `(new Long()).parseLong(mStr);`

> Long (or any wrapper class) does not have a no-args constructor, so new Long() is invalid.

E. `Long.valueOf(mStr).longValue();`

> `Long.valueOf(mStr)` returns a Long object containing 123. `longValue()` on the Long object returns 123.

Back to Question without Answer

24. QID - 2.950 : Handling Exceptions

Consider the following method -

```
public float parseFloat( String s ){
   float f = 0.0f;
   try{
      f = Float.valueOf( s ).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      f = Float.NaN ;
      return f;
   }
   finally{
      f = 10.0f;
      return f;
   }
}
```

What will it return if the method is called with the input "0.0" ?

Correct Option is : B

~~**A.**~~ It will not compile.

B. It will return 10.0

~~**C.**~~ It will return Float.Nan

~~**D.**~~ It will return 0.0

~~**E.**~~ None of the above.

Explanation:

finally block will always execute (except when there is a System.exit() in try or catch). And inside the finally block, it is setting f to 10.0. So no matter what the input is, this method will always return 10.0.

Back to Question without Answer

25. QID - 2.1072 : Working with Inheritance

What, if anything, is wrong with the following code?

```
// Filename: TestClass.java
class TestClass implements T1, T2{
   public void m1(){}
}
interface T1{
   int VALUE = 1;
   void m1();
}
interface T2{
   int VALUE = 2;
   void m1();
}
```

Correct Option is : B

~~**A.**~~ `TestClass` cannot implement them both because it leads to ambiguity.

B. There is nothing wrong with the code.

~~**C.**~~ The code will work fine only if `VALUE` is removed from one of the interfaces.

~~**D.**~~ The code will work fine only if `m1()` is removed from one of the interfaces.

~~**E.**~~ None of the above.

Explanation:

Having ambiguous fields or methods does not cause any problems by itself but

referring to such fields/methods in an ambiguous way will cause a compile time error. So you cannot call : `System.out.println(VALUE);` because it will be ambiguous (there are two `VALUE` definitions). But the following lines are valid :

```
TestClass tc = new TestClass();
System.out.println(( ( T1) tc).VALUE);
```

However, explicit cast is not required for calling the method m1() : `( ( T2) tc).m1();`
`tc.m1()` is also fine because even though `m1()` is declared in both the interfaces, the definition to both resolves unambiguously to only one `m1()`, which is defined in TestClass.

Back to Question without Answer

26. QID - 2.1027 : Working with Java Data Types - Variables and Objects

Note: This question may be considered too advanced for this exam. What will the following code print when run?

```
public class TestClass{
    public static Integer wiggler(Integer x){
       Integer y = x + 10;
       x++;
       System.out.println(x);
       return y;
    }

    public static void main(String[] args){
       Integer dataWrapper = new Integer(5);
       Integer value = wiggler(dataWrapper);
       System.out.println(dataWrapper+value);
    }
}
```

Correct Option is : C

~~**A.**~~ 5 and 20

~~**B.**~~ 6 and 515

C. 6 and 20

~~**D.**~~ 6 and 615

~~**E.**~~ It will not compile.

Explanation:

1. Wrapper objects are always immutable. Therefore, when `dataWrapper` is passed into `wiggler()` method, it is never changed even when `x++;` is executed. However, `x`, which was pointing to the same object as `dataWrapper`, is assigned a new Integer object (different from `dataWrapper`) containing 6.

2. If both the operands of the + operator are numeric, it adds the two operands. Here, the two operands are Integer 5 and Integer 15, so it unboxes them, adds them, and prints 20.

Back to Question without Answer

27. QID - 2.847 : Using Loop Constructs

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        while(--k){
            System.out.println(k);
        }
    }
}
```

Correct Option is : F

~~**A.**~~ 1

~~**B.**~~ 1
0

~~**C.**~~ 2
1

~~**D.**~~ 2
1
0

~~**E.**~~ It will keeping printing numbers in an infinite loop.

F. It will not compile.

In Java, a while or do/while construct takes an expression that returns a boolean. The expression `--i` is an integer, which is invalid and so the compilation fails.

You could change it to: `while( --i>0 ){ ... }`. In this case, `--i<0` is a boolean expression and is valid.

Back to Question without Answer

28. QID - 2.1008 : Overloading methods

Given the following pairs of method declarations, which of the statements are true?

```
1.
void perform_work(int time){ }
int  perform_work(int time, int speed){ return time*speed ;}

2.
void perform_work(int time){ }
int  perform_work(int speed){return speed ;}

3.
void perform_work(int time){ }
void Perform_work(int time){ }
```

Correct Options are : A D

A. The first pair of methods will compile correctly and overload the method 'perform_work'.

~~B.~~ The second pair of methods will compile correctly and overload the method 'perform_work'.

> You cannot have two methods with the same signature (i.e. same name and same parameter list) in the same class.
> Note that return type and names of the parameters don't matter while determining the signature.

~~C.~~ The third pair of methods will compile correctly and overload the method 'perform_work'.

D. The second pair of methods will not compile correctly.

~~**E.**~~ The third pair of methods will not compile correctly.

Both have different names (note the capital 'P') and so are different methods.

Explanation:

Overloading of a method occurs when the name of more than one methods is exactly same but the parameter lists are different.

The first and the third pairs of methods will compile correctly as they follow the above stated rule.

The second pair of methods will not compile correctly, since their method signatures are same and the compiler cannot differentiate between the two methods as it does not look for return type. Also, only name and input parameters are the part of method declaration . Names of the parameters don't matter.

Both methods in the first pair are named perform_work but have different parameter list so they overload this method name i.e. perform_work.

The method named 'perform_work' is distinct from the method named 'Perform_work', as identifiers in Java are case-sensitive.

Back to Question without Answer

29. QID - 2.993 : Working with Inheritance

Which is the first line that will cause compilation to fail in the following program?

```
// Filename: A.java
class A{
   public static void main(String args[]){
      A a = new A();
      B b = new B();
      a = b;  // 1
      b = a;  // 2
      a = (B) b; // 3
      b = (B) a; // 4
   }
}
class B extends A { }
```

Correct Option is : B

~~**A.** At Line 1.~~

B. At Line 2.

> Because 'a' is declared of class A and 'b' is of B which is a subclass of A. So an explicit cast is needed.

~~**C.** At Line 3.~~

~~**D.** At Line 4.~~

~~**E.** None of the above.~~

Explanation:

Casting a base class to a subclass as in : `b = (B) a;` is also called as narrowing (as you are trying to narrow the base class object to a more specific class object) and needs explicit cast.
Casting a sub class to a base class as in: `A a = b;` is also called as widening and does not need any casting.

For example, consider two classes: Automobile and Car, where Car extends Automobile
Now, `Automobile a = new Car();` is valid because a car is definitely an Automobile. So it does not need an explicit cast.

But, Car c = a; is not valid because 'a' is an Automobile and it may be a Car, a Truck, or a MotorCycle, so the programmer has to explicitly let the compiler know that at runtime 'a' will point to an object of class Car. Therefore, the programmer must use an explicit cast:

```
Car c = (Car) a;
```

Back to Question without Answer

30. QID - 2.1060 : Using Loop Constructs

Identify the valid for loop constructs assuming the following declarations:

```
Object o = null;
Collection c = //valid collection object.
int[][] ia = //valid array
```

Correct Options are : B E

~~**A.**~~ `for(o : c){ }`

> Cannot use an existing/predefined variable in the variable declaration part.

B. `for(final Object o2 :c){ }`

> final is the only modifier (excluding annotations) that is allowed here.

~~**C.**~~ `for(int i : ia) { }`

> Each element of ia is itself an array. Thus, they cannot be assigned to an int.

~~**D.**~~ `for(Iterator it : c.iterator()){ }`

> `c.iterator()` does not return any Collection. Note that the following would have been valid:
>
> ```
> Collection<Iterator> c = //some collection that contains
> Iterator objects
> for(Iterator it : c){ }
> ```

E. `for(int i : ia[0]){ }`

Since `ia[0]` is an array of ints, this is valid. (It may throw a `NullPointerException` or `ArrayIndexOutOfBoundsException` at runtime if `ia` is not appropriately initialized.)

Back to Question without Answer

31. QID - 2.1315 : Working with Inheritance

Note: This question may be considered too advanced for this exam.

Given the declaration

```
interface Worker { void perform_work(); }
```

which of the following methods/classes are valid?

Correct Options are : B D

~~**A.**~~
```
Worker getWorker(int i){
   return new Worker(){   public void perform_work() {
        System.out.println(i); }    };
}
```

Since method parameter i is not final, it cannot be accessed from perform_work().

B.
```
Worker getWorker(final int i){
   return new Worker() {   public void perform_work() {
   System.out.println(i); }    };
}
```

Since method paramter 'i' is final, it can be accessed from perform_work();

~~**C.**~~
```
Worker getWorker(int i){
   int x = i;
   class MyWorker implements Worker  {
         public void perform_work() { System.out.println(x); }    };
   return new MyWorker();
}
```

x is not accessible from within perform_work() either. In fact, i and x are

similar for all practical purposes in terms of accessibility.

D.
```
Worker getWorker(final int i){
   class MyWorker implements Worker {
       public void perform_work() { System.out.println(i); }    };
   return new MyWorker();
}
```

Explanation:

Do not get confused with option 2. You are not instantiating an interface, you are instantiating an anonymous class that implements the interface Worker.

FYI, if you have a nested static class `MyWorker` in `TestClass` as follows,

```
public class TestClass{
   public static class MyWorker implements Worker{
       public MyWorker(int i){ }
       public void perform_work(){ }
   }
}
```

`MyWorker` can be instantiated in any other class by doing:
`new TestClass.MyWorker( someInt );`

There is no instance of `TestClass` associated with the `MyWorker` class in this case. Inside `TestClass` you can instantiate MyWorker directly `(new MyWorker(someInt))` in static or non static context.

Remember: A nested class is any class whose declaration occurs within the body of another class or interface. A top level class is a class that is not a nested class. An inner class is a nested class that is not explicitly or implicitly declared static. A class defined inside an interface is implicitly static.

Back to Question without Answer

32. QID - 2.1058 : Overloading methods

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10, 20);  //1
   }

   public void test1(int i, int... j){ System.out.println("1"); }
   public void test1(int... i ){ System.out.println("2"); }
   public void test1(int i, int j){ System.out.println("3"); }

   public static void main(String[] args){
     new Varargs().test();
   }
}
```

What will the program print?

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 2

C. 3

~~**D.**~~ It will not compile.

~~**E.**~~ Exception at runtime.

Explanation:

In cases where multiple methods are applicable, the compiler always calls the most specific one. In this case, the third one is the most specific one.

If no method is more specific than the other, then the compilation fails. For example, if the class did not have the third method test1(int i, int j), then the remaining two methods would have been equally applicable and equally specific. In that case, it would not compile.

Back to Question without Answer

33. QID - 2.928 : Java Basics

Given the following set of member declarations, which of the following is true?

```
int a;     //  (1)
static int a;     //  (2)
int f( )    { return a; }     //  (3)
static int f( ) { return a; }     //  (4)
```

Correct Options are : C E

~~**A.**~~ Declarations (1) and (3) cannot occur in the same class definition.

~~**B.**~~ Declarations (2) and (4) cannot occur in the same class definition.

A static method can refer to a static field.

C. Declarations (1) and (4) cannot occur in the same class definition.

because method f() is static and a is not.

~~**D.**~~ Declarations (2) and (3) cannot occur in the same class definition.

E. Declarations (1) and (2) cannot occur in the same class definition.

variable names must be different.

Explanation:

Local variables can have same name as member variables. The local variables will simply shadow the member variables with the same names.

Declaration (4) defines a static method that tries to access a variable named 'a' which is not locally declared.
Since the method is static, this access will only be valid if variable 'a' is declared static within the class. Therefore declarations (1) and (4) cannot occur in the same definition.

Back to Question without Answer

34. QID - 2.1222 : Working with Inheritance

Consider the following class hierarchy

```
class A{
   public void m1() {    }
}
class B extends A{
   public void m1() {    }
}
class C extends B{
   public void m1(){
      /*  //1
      ... lot of code.
      */
   }
}
```

Correct Options are : A B

A. You cannot access class A's `m1()` from class C for the same object (i.e. `this`).

B. You can access class B's `m1()` using `super.m1()` from `class C`.

~~**C.**~~ You can access `class A`'s `m1()` using `( (A) this ).m1()` from `class C`.

> Note that selection of method to be executed depends upon the actual object class. So no matter what you do, in `class C` you can only access C's `m1()` even by casting this to B or A. So, this option will not work.

~~**D.**~~ You can access `class A`'s `m1()` using `super.super.m1()` from `class C`.

Explanation:

There is no construct like super.super. So, there is no way you can access `m1()` of A from C.

Back to Question without Answer

35. QID - 2.897 : Creating and Using Arrays

Given the following code :

```
public class TestClass {

    int[][] matrix = new int[2][3];

    int a[] = {1, 2, 3};
    int b[] = {4, 5, 6};

    public int compute(int x, int y){
        //1 : Insert Line of Code here
    }

    public void loadMatrix(){
        for(int x=0; x<matrix.length; x++){
            for(int y=0; y<matrix[x].length; y++){
                //2: Insert Line of Code here
            }
        }
    }
}
```

What can be inserted at //1 and //2?

Correct Option is : C

~~**A.**~~ `return a(x)*b(y);`

and

`matrix(x, y) = compute(x, y);`

> (and) are used to call a method on an object. To access array elements, you need to use [and].

~~**B.**~~ `return a[x]*b[y];`

and

```
matrix[x, y] = compute(x, y);
```

C. `return a[x]*b[y];`
and

```
matrix[x][y] = compute(x, y);
```

~~**D.**~~ `return a(x)*b(y);`
and

```
matrix(x)(y) = compute(x, y);
```

> `a(x), b(y),` and `matrix(x)(y)` are invalid because a, b, and matrix are not methods.

~~**E.**~~ `return a[x]*b[y];`
and

```
matrix[[x][y]] = compute(x, y);
```

> `[[x][y]]` is invalid syntax.

Explanation:

The correct syntax to access any element within an array is to use the square brackets - `[ ]`. Thus, to access the first element in an array, you would use `array[0]`.

For a multi dimensional array, to reach an individual item, you need to specify index for each dimension. For example, since matrix is a two dimensional array, matrix is an array of array and `matrix[0]` will give you the first array of the arrays. `matrix[0][0]` will give you the first element of the first array of the arrays.

Back to Question without Answer

36. QID - 2.1026 : Handling Exceptions

Given the class

```
// Filename: Test.java
public class Test{
   public static void main(String args[]){
      for(int i = 0; i< args.length; i++){
         System.out.print("  "+args[i]);
      }
   }
}
```

Now consider the following 3 options for running the program:

```
a: java Test
b: java Test param1
c: java Test param1 param2
```

Which of the following statements are true?

Correct Options are : D E

~~**A.**~~ The program will throw `java.lang.ArrayIndexOutofBoundsException` on option a.

~~**B.**~~ The program will throw `java.lang.NullPointerException` on option a.

~~**C.**~~ The program will print `Test param1` on option b.

> Unlike in C++, Name of the file is not passed in args because for a public class it is always same as the name of the class.

D. It will print `param1 param2` on option c.

E. It will not print anything on option a.

Explanation:

It will not throw `NullPointerException` because `args[]` is never `null`. If no argument is given (as in option a) then the length of args is 0.

Back to Question without Answer

37. QID - 2.961 : Working with Inheritance

Which one of these is a proper definition of a class Car that cannot be sub-classed?

Correct Option is : E

~~**A.**~~ `class Car { }`

> This can be subclassed.

~~**B.**~~ `abstract class Car { }`

> it cannot be instantiated but it can be subclassed.

~~**C.**~~ `native class Car { }`

> Classes and variables can't be declared native. Only methods can be native.

~~**D.**~~ `static class Car { }`

> package level classes can't be declared static.

E. `final class Car { }`

> final keyword prevents a class from being subclassed and a method from being overridden.

Explanation:

A class can be extended unless it is declared final. While declaring a method, static usually implies that it is also final, this is not true for classes.
An inner class can be declared static and still be extended. Notice the distinction. For

classes, final means it cannot be extended, while for methods, final means it cannot be overridden in a subclass.
The native keyword can only be used on methods, not on classes and instance variables.

Back to Question without Answer

38. QID - 2.957 : Working with Methods

What will the following program print?

```
public class TestClass{
  static int someInt = 10;
  public static void changeIt(int a){
    a = 20;
  }
  public static void main(String[] args){
    changeIt(someInt);
    System.out.println(someInt);
  }
}
```

Correct Option is : A

A. 10

~~**B.**~~ 20

~~**C.**~~ It will not compile.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

In case of primitives such an an int, it is the value of the primitive that is passed. For

example, in this question, when you pass `someInt` to `changeIt` method, you are actually passing the value `10` to the method, which is then assigned to method variable '`a`'. In the method, you assign `20` to '`a`'. However, this does not change the value contained in `someInt`. `someInt` still contains `10`. Therefore, `10` is printed.

Theoretically, java supports Pass by Value for everything (i.e. primitives as well as Objects).

. Primitives are always passed by value.
. Object "references" are passed by value. So it looks like the object is passed by reference but actually it is the value of the reference that is passed.

An example:

```
    Object o1 = new Object(); //Let us say, the object is stored at
memory location 15000.
        //Since o1 actually stores the address of the memory
location where the object is stored, it contains 15000.
```

Now, when you call someMethod(o1); the value 15000 is passed to the method.

Inside the method someMethod():

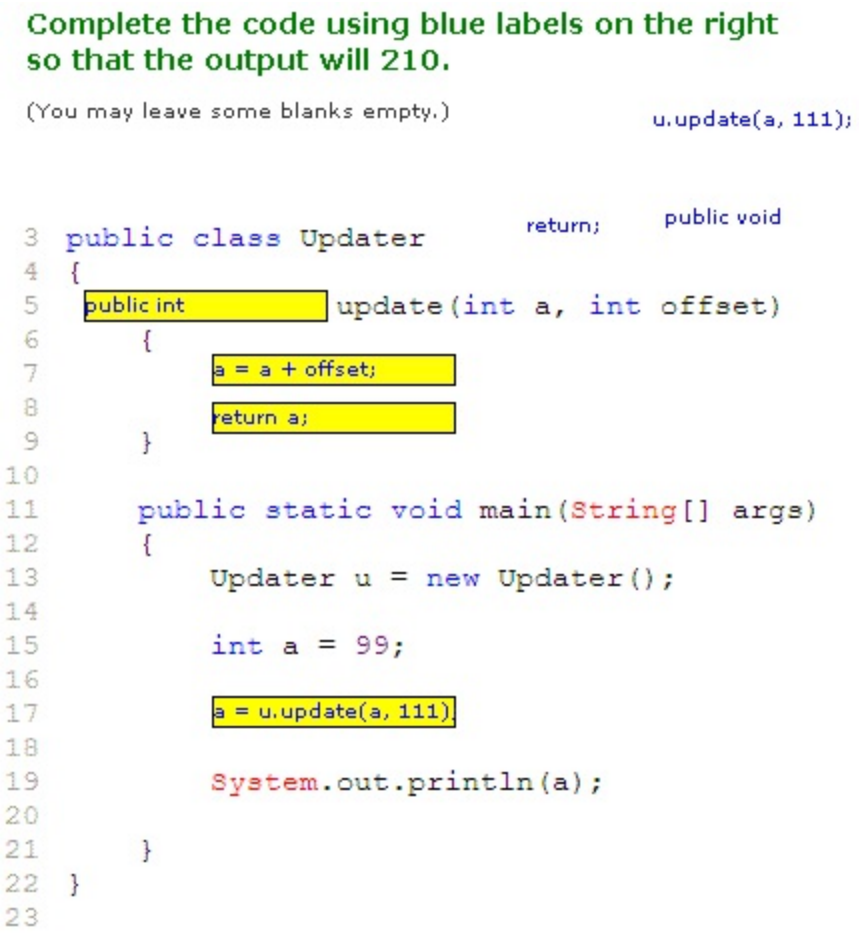
```
    someMethod( Object localVar) {
            /*localVar now contains 15000, which means it also
points to the same memory location where the object is stored.
            Therefore, when you call a method on localVar, it will
be executed on the same object.
            However, when you change the value of localVar itself,
for example if you do localVar=null,
            it then it starts pointing to a different memory
location. But the original variable o1 still
            contains 15000 so it still points to the same object. */
        }
```

If you need even more detailed explanation, please check

http://www.javaranch.com/campfire/StoryPassBy.jsp

Back to Question without Answer

39. QID - 2.1218 : Working with Methods



Explanation:

This question is based on the principle that primitives are always passed by value. Thus, when you pass a to `update()` method, the value of a is passed. The variable `a` in `update()` method is not same as the `a` in `main()`. It is a completely different variable and so updating `update()` method's `a` does not affect `main()'s a`. Therefore, we need to return the new value from `update()` method and assign it to `main()'s a`.

The following is the complete code listing:

```
public class Updater {
  public int update(int a, int offset){
    a = a + offset;
```

```
    return a;
  }

  public static void main(String[] args) {
    Updater u = new Updater();
    int a = 99;
    a = u.update(a, 111);
    System.out.println(a);
  }
}
```

Back to Question without Answer

40. QID - 2.967 : Handling Exceptions

What will the following code print when compiled and run?

```
abstract class Calculator{
   abstract void calculate();
   public static void main(String[] args){
      System.out.println("calculating");
      Calculator x = null;
      x.calculate();
   }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

> It will compile fine.

~~**B.**~~ It will not print anything and will throw `NullPointerException`

C. It will print `calculating` and then throw `NullPointerException`.

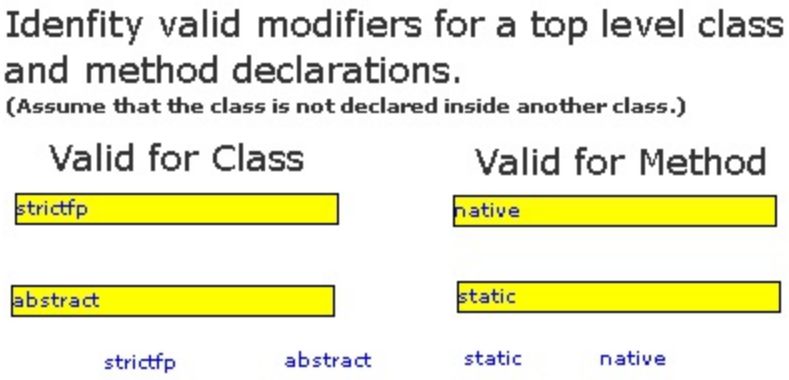
> After printing, when it tries to call `calculate()` on `x`, it will throw `NullPointerException` since `x` is `null`.

~~**D.**~~ It will print `calculating` and will throw `NoSuchMethodError`

~~**E.**~~ It will print `calculating` and will throw `MethodNotImplementedException`

Back to Question without Answer

41. QID - 2.1139 : Java Basics



Explanation:

Only methods can be 'native'.
A top level class (i.e. a class that is not nested inside any other class) cannot be declared as static.
Both - a class or a method can be declared as strictfp.

Back to Question without Answer

42. QID - 2.1231 : Working with Inheritance

Given the following interface definition, which definitions are valid?

```
interface I1{
   void setValue(String s);
   String getValue();
}
```

Correct Options are : B C

~~A.~~
```
class A extends I1{
   String s;
   void setValue(String val) { s = val; }
   String getValue() { return s; }
}
```

> Classes do not `extend` interfaces, they implement interfaces.

B.
```
interface I2 extends I1{
   void analyse();
}
```

C.
```
abstract class B implements I1{
   int getValue(int i) { return 0; }
}
```

~~D.~~
```
interface I3 implements I1{
   void perform_work();
}
```

> Interfaces do not `implement` anything, they can extend multiple interfaces.

Explanation:

The `getValue(int i)` method of class B in option c, is different than the method defined in the interface because their parameters are different. Therefore, this class does not actually implement the method of the interface and that is why it needs to be declared abstract.

Back to Question without Answer

43. QID - 2.830 : Working with Methods

Consider the following method :

```
public void myMethod(int m, Object p, double d){
  ... valid code here
}
```

Assuming that there is no other method with the same name, which of the following options are correct regarding the above method?

Correct Option is : E

~~**A.**~~ If this method is called with two parameters, the value of d in the method will be 0.0.

~~**B.**~~ If this method is called with one parameter, the value of p and d in the method will be null and 0.0 respectively.

~~**C.**~~ If this method is called with one parameter, the call will throw a NullPointerException.

~~**D.**~~ If this method is called with one parameter, the call will throw a NullPointerException only if the code in the method tries to access p.

E. If this method is called with two parameters, the code will not compile.

Explanation:

To call `myMethod(int m, Object p, double d)`, you must pass exactly three

parameters. If you try to pass less (or more) number of parameters, the code will not compile. Note that method parameters are not assigned default values.

It is possible to declare a method that can take variable number of parameters. For example:

```
public static void someMethod(Object... params){
        System.out.println(params.length);
    }
```

You can call this method by passing any number of parameters. In this case, calling `someMethod()` without any parameter will print 0. i.e. the length of params array will be 0. `params` will NOT be `null`.

Back to Question without Answer

44. QID - 2.1099 : Using Loop Constructs

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int x  = 0;
      labelA:   for (int i=10; i<0; i--){
         int j = 0;
         labelB:
         while (j < 10){
            if (j > i) break labelB;
            if (i == j){
               x++;
               continue labelA;
            }
            j++;
         }
         x--;
      }
      System.out.println(x);
   }
}
```

Correct Option is : D

~~**A.**~~ It will not compile.

~~**B.**~~ It will go in infinite loop when run.

~~**C.**~~ The program will write 10 to the standard output.

D. The program will write 0 to the standard output.

~~**E.**~~ None of the above.

Explanation:

This is just a simple code that is meant to confuse you.
Notice the for statement: for `(int i=10; i<0; i--)`. i is being initialized to 10 and the test is i<0, which is false. Therefore, the control will never get inside the for loop, none of the weird code will be executed, and x will remain 0, which is what is printed.

Back to Question without Answer

45. QID - 2.1094 : Handling Exceptions

What will the following program print when run?

```
public class TestClass{
  public static void main(String[] args){
     try{
        System.exit(0);
     }
     finally{
         System.out.println("finally is always executed!");
     }
  }
}
```

Correct Option is : C

~~**A.**~~ It will print "finally is always executed!"

~~**B.**~~ It will not compile as there is no catch block.

C. It will not print anything.

~~**D.**~~ An exception will be thrown

~~**E.**~~ None of the above.

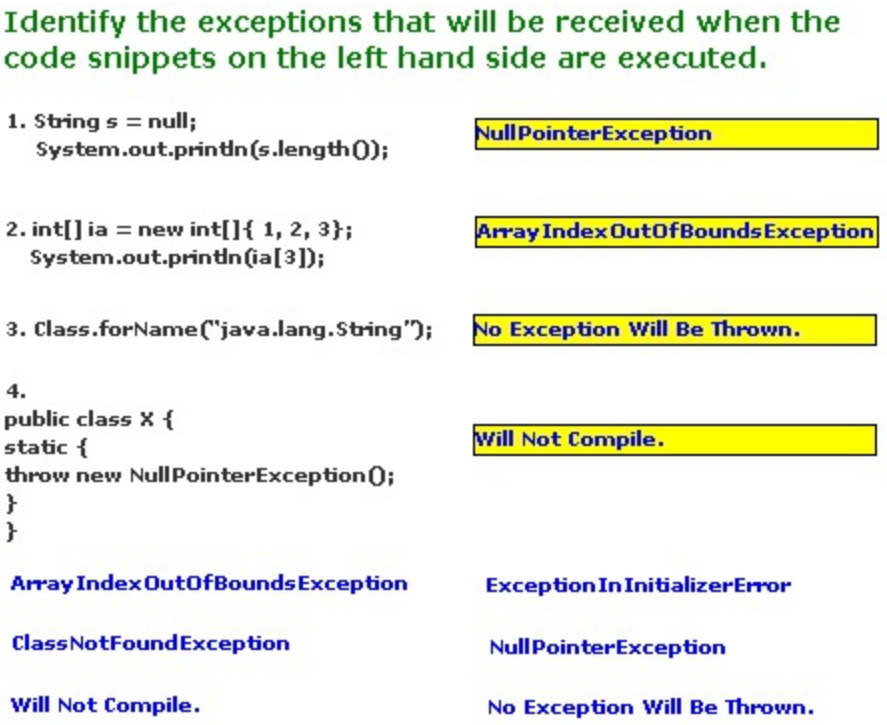
Explanation:

finally is always executed (even if you throw an exception in try or catch) but this is the exception to the rule.

When you call System.exit(...); The JVM exits so there is no way to execute the finally block.

Back to Question without Answer

46. QID - 2.1031 : Handling Exceptions



Explanation:

Please read ExceptionClassSummary document in the "Study References" section. Note that the question is not asking what exception you need to put in the catch(...) part or throws clause. It is just asking what exceptions will be thrown by the code fragments when they are executed.

Back to Question without Answer

47. QID - 2.1028 : Handling Exceptions

Identify the exceptions that SHOULD be thrown in the situations shown below.

```
1.
public void closeReportManager(ReportManager rep)
{
 if(!rep.isClosed()) report.close();
 else throw new IllegalStateException();
}


2.
List validFormats = Arrays.asList("HTML", "JAVA", "JSP");
public void reformat(String format)
{
 if(validFormats.contains(format)) applyFormatting(format);
 else throw new IllegalArgumentException();
}
```

NullPointerException(); IllegalStateException();

IllegalArgumentException(); Exception();

Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

48. QID - 2.1321 : Java Basics

Consider the following class :

```
public class Parser{
   public static void main( String[] args){
       try{
           int i = 0;
           i =  Integer.parseInt( args[0] );
       }
       catch(NumberFormatException e){
          System.out.println("Problem in " + i );
       }
   }
}
```

What will happen if it is run with the following command line:

```
java Parser one
```

Correct Option is : C

~~**A.**~~ It will print `Problem in 0`

~~**B.**~~ It will throw an exception and end without printing anything.

C. It will not even compile.

Because 'i' is defined in try block and so it is not visible in the catch block.

~~**D.**~~ It will not print anything if the argument is '1' instead of 'one'.

E. None of the above.

Back to Question without Answer

49. QID - 2.1240 : Using Operators and Decision Constructs

Given:

```
byte b = 1;
char c = 1;
short s = 1;
int i = 1;
```

which of the following expressions are valid?

Correct Options are : B C E

~~**A.**~~ `s = b * b ;`

b * b returns an int.

B. `i = b << b ;`

C. `s <<= b ;`

All compound assignment operators internally do an explicit cast.

~~**D.**~~ `c = c + b ;`

c + b returns an int

E. `s += i ;`

All compound assignment operators internally do an explicit cast.

Explanation:

Remember these rules for primitive types:
1. Anything bigger than an int can NEVER be assigned to an int or anything smaller than int (byte, char, or short) without explicit cast.
2. CONSTANT values up to int can be assigned (without cast) to variables of lesser size (for example, short to byte) if the value is representable by the variable.(that is, if it fits into the size of the variable).
3. operands of mathematical operators are ALWAYS promoted to AT LEAST int. (i.e. for byte * byte both bytes will be first promoted to int.) and the return value will be AT LEAST int.
4. Compound assignment operators (+=, *= etc) have strange ways so read this carefully:

A compound assignment expression of the form E1 op= E2 is equivalent to `E1 = (T) ((E1) op (E2))`, where T is the type of E1, except that E1 is evaluated only once. Note that the implied cast to type T may be either an identity conversion or a narrowing primitive conversion.
For example, the following code is correct:

```
short x = 3;
x += 4.6;
```

and results in x having the value 7 because it is equivalent to:

```
short x = 3;
x = (short)(x + 4.6);
```

Back to Question without Answer

50. QID - 2.941 : Working with Java Data Types - String, StringBuilder

What will the following code print?

```
String abc = "";
abc.concat("abc");
abc.concat("def");
System.out.print(abc);
```

Correct Option is : D

~~**A.**~~ abc

~~**B.**~~ abcdef

~~**C.**~~ def

D. It will print empty string (or in other words, nothing).

~~**E.**~~ It will not compile because there is no concat() method in String class.

Explanation:

Strings are immutable so doing abc.concat("abc") will create a new string "abc" but will not affect the original string "".

Back to Question without Answer

51. QID - 2.1203 : Working with Inheritance

A method with no access modifier can be overridden by a method marked protected.

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

An Overriding method is allowed to make the overridden method more accessible, and since protected is more accessible than default (package), this is allowed. Note that protected access will allow access to the subclass even if the subclass is in a different package but package access will not.

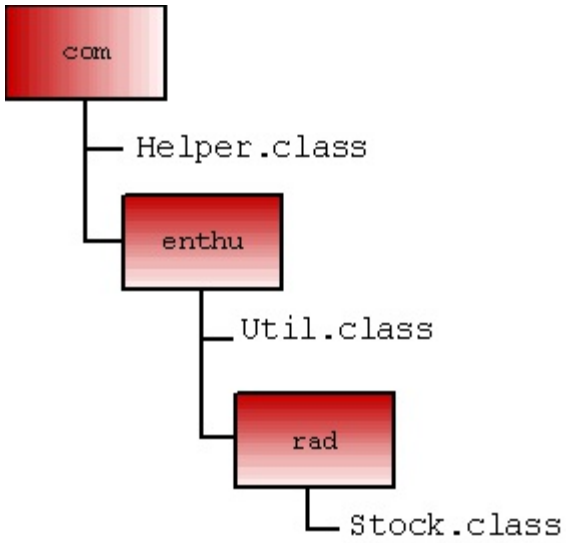
Back to Question without Answer

52. QID - 2.1063 : Java Basics

Consider the directory structure shown in Image 1 that displays available folders and classes and the code given below:

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public void store() throws IOException{
    return Util.store(stock);
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements MUST be added to the above class?



Correct Options are : B C D E

~~**A.**~~ `package com.enthu.rad.*;`

Bad syntax. A package statement can never have a *. It should specify the exact package name.

B. `import com.enthu.*;`

C. `package com.enthu.rad;`

Since there is no import statement available for com.enthu.rad package, you must put the given class in com.enthu.rad package so that it will be accessible. Classes of the same package are always available to each other.

D. `import com.*;`

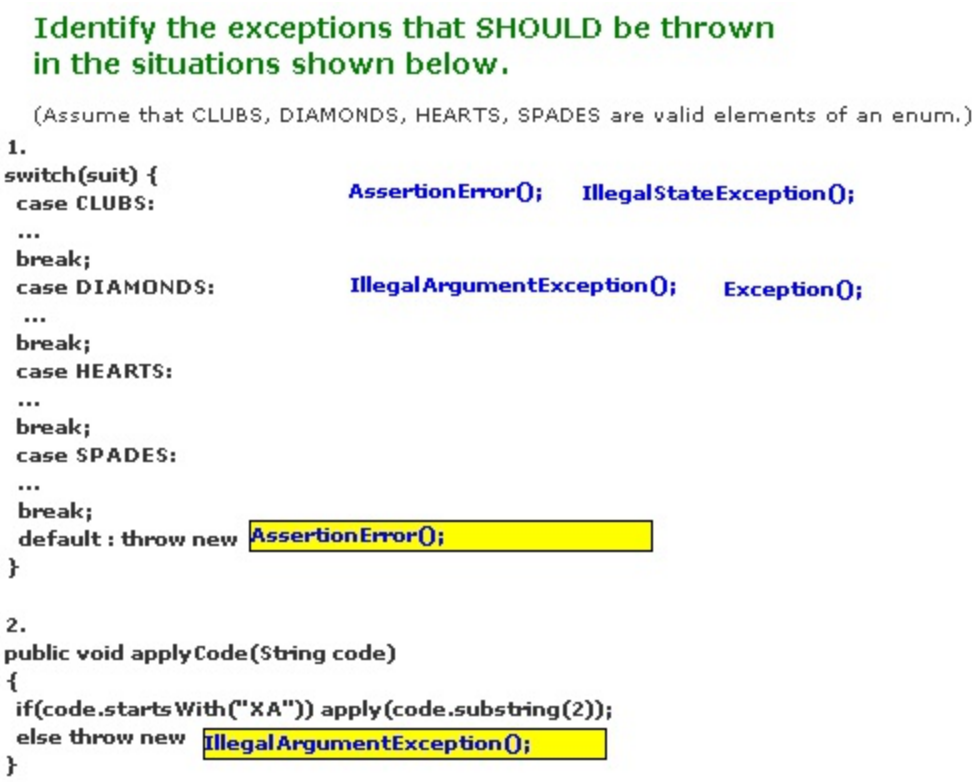
E. `import java.io.*;`

F. It is not required to import java.io.* or import java.io.IOException because java.io package is imported automatically.

Since the code is using IOException, the java.io package (or just java.io.IOException class) must be imported. Only java.lang package is imported automatically.

Back to Question without Answer

53. QID - 2.1030 : Handling Exceptions



Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

54. QID - 2.1273 : Working with Inheritance

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      A o1 = new C( );
      B o2 = (B) o1;
      System.out.println(o1.m1( ) );
      System.out.println(o2.i );
   }
}
class A { int i = 10;  int m1( ) { return i; } }
class B extends A { int i = 20;  int m1() { return i; } }
class C extends B { int i = 30;  int m1() { return i; } }
```

Correct Option is : C

~~**A.**~~ The program will fail to compile.

~~**B.**~~ Class cast exception at runtime.

C. It will print 30, 20.

~~**D.**~~ It will print 30, 30.

~~**E.**~~ It will print 20, 20.

Explanation:

Remember : variables are SHADOWED and methods are OVERRIDDEN.

Which variable will be used depends on the class that the variable is declared of.
Which method will be used depends on the actual class of the object that is referenced by the variable.
So, in line `o1.m1()`, the actual class of the object is C, so C's `m1()` will be used. So it returns 30.
In line o2.i, o2 is declared to be of class B, so B's i is used. So it returns 20.

Back to Question without Answer

55. QID - 2.1059 : Using Loop Constructs

Identify valid for constructs...
Assume that Math.random() returns a double between 0.0 and 1.0 (not including 1.0).

Correct Options are : A C E

A.
```
for(;Math.random()<0.5;){
    System.out.println("true");
}
```

The second expression in a for loop must return a boolean, which is happening here. So this is valid.

B.
```
for(;;Math.random()<0.5){
    System.out.println("true");
}
```

Here, the first part (i.e. the init part) and the second part (i.e. the expression/condition part) part of the for loop are empty. Both are valid. (When the expression/condition part is emtpy, it is interpreted as true.)

The third part (i.e. the update part) of the for loop does not allow every kind of statement. It allows only the following statements here: Assignment, PreIncrementExpression, PreDecrementExpression, PostIncrementExpression, PostDecrementExpression, MethodInvocation, and ClassInstanceCreationExpression. Thus, Math.random()<0.5 is not valid here, and so this will not compile.

C.
```
for(;;Math.random()){
    System.out.println("true");
}
```

This is a valid never ending loop that will keep printing true.

D.
```
for(;;){
    Math.random()<.05? break : continue;
}
```

This is an invalid use of ? : operator. Both sides of : should return the same type (excluding void). Here, break and continue do not return anything. However, the following would have been valid:
for(;Math.random()<.05? true : false;){ }

E.
```
for(;;){
    if(Math.random()<.05) break;
}
```

Explanation:

The three parts of a for loop are independent of each other. However, there are certain rules for each part. Please go through section 14.14.1 of JLS to understand it fully.

Back to Question without Answer

56. QID - 2.981 : Working with Methods - Access Modifiers

For object o1 of class A to access a member(field or method) of object o2 of class B, when the member has no access modifier, class B must be...

Correct Option is : B

~~**A.**~~ a subclass of A

> It cannot access the member if it is not in the same package.

B. in the same package as A is in.

> This is correct. B may or may not be a Subclass of A.

~~**C.**~~ a Super class of A

~~**D.**~~ a subclass but may not be in the same package.

> Has to be in the same package.

~~**E.**~~ in the same package and must be a Subclass of A.

Explanation:

No access modifier means "default" access which means only classes of the same package can access it.
Note that "default" is not a valid access specifier. The absence of any access modifier means default access.

Back to Question without Answer

57. QID - 2.1108 : Using Operators and Decision Constructs

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int k = 9, s = 5;
      switch(k){
         default :
         if( k == 10) { s = s*2; }
         else{
            s = s+4;
            break;
         }
         case 7 : s = s+3;
      }
      System.out.println(s);
   }
}
```

Correct Option is : B

~~**A.** 5~~

B. 9

Since 9 does not match any of the case labels, it is accepted by default block. In this block, the else part is executed, which sets s to the value of s+4, i.e. 9. Since there is a break in the else block, case 7: is not executed.

~~**C.** 12~~

~~**D.** It will not compile.~~

Back to Question without Answer

58. QID - 2.1284 : Working with Java Data Types - String, StringBuilder

What will the following class print when run?

```
public class Sample{
   public static void main(String[] args)  {
     String s1 = new String("java");
     StringBuilder s2 = new StringBuilder("java");
     replaceString(s1);
     replaceStringBuilder(s2);
     System.out.println(s1 + s2);
  }
  static void replaceString(String s) {
     s = s.replace('j', 'l');
  }
  static void replaceStringBuilder(StringBuilder s) {
     s.append("c");
  }
}
```

Correct Option is : C

A. ~~javajava~~

B. ~~lavajava~~

C. javajavac

D. ~~lavajavac~~

E. ~~None of these.~~

Explanation:

String is immutable while StringBuilder is not. So no matter what you do in `replaceString()` method, the original String that was passed to it will not change. On the other hand, StringBuilder methods, such as replace or append, change the StringBuilder itself. So, 'c' is appended to java in `replaceStringBuilder()` method.

Back to Question without Answer

59. QID - 2.1283 : Working with Methods

What is the correct declaration for an abstract method 'add' accessible to any class, takes no arguments and returns nothing?
(Use only one space between words)

Correct Option is : E

~~**A.**~~ `public void add();`

> An abstract method must have the `abstract` keyword and must not have a method body i.e. { }.

~~**B.**~~ `abstract add();`

> A method that is not supposed to return anything must specify `void` as its return type.

~~**C.**~~ `abstract null add();`

> A method that is not supposed to return anything must specify `void` as its return type. `null` is not a type, though it is a valid return value for any type.

~~**D.**~~ `abstract public void add(){ }`

> It is invalid because has a method body i.e. { }.

E. `abstract public void add() throws Exception;`

Back to Question without Answer

60. QID - 2.1143 : Working with Inheritance

Which of the given statements are correct for a method that overrides the following method:

```
public Set getSet(int a) {...}
```

Correct Options are : B D E

~~**A.**~~ Its return type must be declared as `Set`.

> Return type may also be a subclass/subinterface. So it can also return `SortedSet`, `TreeSet`, `HashSet`, or any other class that implements or subclasses a `Set`.

B. It may return HashSet.

~~**C.**~~ It can declare any Exception in throws clause

> Since the original (overridden) method does not have any throws clause, the overriding method cannot declare any checked exceptions.

D. It can declare any RuntimeException in throws clause.

> A method can throw any RuntimeException (such as a NullPointerException) even without declaring it in its throws clause.

E. It can be abstract.

> Yes, you can make it abstract!! You would have to make the class as abstract as well though.

Explanation:

To override a method in the subclass, the overriding method (i.e. the one in the subclass) MUST HAVE:
.same name
.same return type in case of primitives (a subclass is allowed for classes, this is also known as covariant return types).
.same type and order of parameters
.It may throw only those exceptions that are declared in the throws clause of the superclass's method or exceptions that are subclasses of the declared exceptions. It may also choose NOT to throw any exception.
The names of the parameter types do not matter. For example, void `methodX(int i)` is same as void `methodX(int k)`

Back to Question without Answer

61. QID - 2.1292 : Working with Inheritance

Given the definitions of I and Klass, complete the definition of SubClass so that it extends from Klass and implements I.
Use minimum number of elements.

```
interface I
{
    void m1();
}

abstract class Klass
{
    void m1() { };
}

class SubClass extends Klass implements I
{
    public void m1(){ }
}
```

Explanation:

Even though class Klass implements `m1()`, it does not declare that it implements I. Therefore, for a subclass to 'implement' I, it must have 'implements I' in its declaration.
Further, `m1()` in Klass is not public. So even though Subclass inherits m1() from Klass, it will not be a valid implementation of I because interface methods must be public. Therefore, SubClass must override `m1()` and make it public.

Back to Question without Answer

62. QID - 2.1163 : Java Basics

The following is a valid member variable declaration:

```
private static final transient int i = 20;
```

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

Although it does not make sense to make static variable transient as static variables are not serialized anyway, it is valid to do so.
You can apply all the modifiers to member variables except `abstract, native` and `synchronized`.
For methods, you cannot apply `transient` and `volatile`.

Back to Question without Answer

63. QID - 2.902 : Working with Methods

Given:

```
class Triangle{
    public int base;
    public int height;
    public double area = 0;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }
    public void updateArea(){
        double a = base*height/2;
        area = a;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

Which variables are not accessible from anywhere within given class code (except from where they are declared)?

Correct Option is : D

~~**A.**~~ base, height, area

~~**B.**~~ area, b, h

~~**C.**~~ base, height

D. b, h, a

b and h are method parameters and are only accessible in the method setBase and setHeight respectively.
a is a local variable and is accessible only in updateArea method.

base, height, and area are instance level and can be accessed from anywhere within the class where "this" is accessible.

Explanation:

"class level" means static fields and they can be accessed from anywhere (i.e. static as well as non-static methods) in the class (and from outside the class depending on their accessibility).
"instance level" means the instance fields and they can be accessed only from instance methods in the class.

Back to Question without Answer

64. QID - 2.1266 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Options are : A B E

A. The condition expression in an if statement can contain method calls.

> Yes, as long as the method returns a boolean value.

B. If a and b are of type boolean, the expression (a = b) can be used as the condition expression of an if statement.

~~**C.**~~ An if statement can have either an 'if' clause or an 'else' clause.

> An if-statement must always have an 'if' clause. 'else' is optional.

~~**D.**~~ The statement : if (false) ; else ; is illegal.

> if-clause and the else-clause can have empty statements. Empty statement (i.e. just a semi-colon) is a valid statement.

E. Only expressions which evaluate to a boolean value can be used as the condition in an if statement.

> Unlike C/C++ where you can use integers as conditions, in java, only booleans are allowed.

Explanation:

The expression `(a = b)` does not compare the variables a and b, but rather assigns the value of b to the variable a. The result of the expression is the value being assigned.

Since a and b are `boolean` variables, the value returned by the expression is also `boolean`. This allows the expressions to be used as the condition for an if-statement. if-clause and the else-clause can have empty statements. Empty statement (i.e. just ;) is a valid statement.
But this is illegal :

```
if (true) else;
```

because the if part doesn't contain any valid statement. (A statement cannot start with an else!)
So, the following is valid.

```
if(true) if(false);
```

because `if(false);` is a valid statement.

Back to Question without Answer

65. QID - 2.1330 : Working with Methods - Access Modifiers

Which of the following access control keywords can be used to enable all the subclasses to access a method defined in the base class?

Correct Options are : A C

A. public

It will allow everybody to access the method.

~~**B.**~~ private

It will not allow any other class to access the method.

C. protected

It will allow the subclasses and the classes in same package to access the method.

~~**D.**~~ No keyword is needed.

It will allow only the classes in same package to access the method. So Subclasses outside the package will not have access.

Back to Question without Answer

66. QID - 2.1184 : Working with Java Data Types - String, StringBuilder

Given:

```
StringBuilder b1 = new StringBuilder("snorkler");
StringBuilder b2 = new StringBuilder("yoodler");
```

Write the contents of b1 and b2 after the statements shown on the left are executed independent of each other.

Statements	Contents of b1	Contents of b2
`b1.append(b2.substring(2, 5) .toUpperCase());`	snorklerODL	yoodler
`b2.insert(3, b1.append("a"));`	snorklera	yoosnorkleradler
`b1.replace(3, 4, b2.substring(4)) .append(b2.append(false));`	snolerkleryoodlerfalse	yoodlerfalse

Explanation:

You need to understand how `append, insert, delete,` and `substring` methods of `StringBuilder/StringBuffer` work. Please go through JavaDoc API for these methods. This is very important for the exam. Observe that `substring()` does not modify the object it is invoked on but `append, insert` and `delete` do.

In the exam, you will find questions that use such quirky syntax, where multiple calls are chained together. For example: `sb.append("a").append("asdf").insert(2, "asdf")`. Make yourself familiar with this technique. If in doubt, just break it down into multiple calls. For example, the aforementioned statement can be thought of as:

```
sb.append("a");
sb.append("asdf");
sb.insert(2, "asdf")
```

Note that the method `substring()` in `StringBuilder/StringBuffer` returns a `String` (and not a reference to itself, unlike `append, insert,` and `delete`). So

another StringBuilder method cannot be chained to it. For example, the following is not valid: `sb.append("a").substring(0, 4).insert(2, "asdf");`

The following is valid though: `String str = sb.append("a").insert(2, "asdf").substring(0, 4);`

Back to Question without Answer

67. QID - 2.1089 : Using Operators and Decision Constructs

Drag and drop valid operators (shown in blue) in yellow boxes.
= operator can be used more than once.

`=` `+=` `*=`

```
public class TestClass
{
   public static void main(String[] args)
   {
     Short k = 9;     Integer i = 9;  Boolean b = false;
     char c = 'a';    String str = "123";

     i      [ = ]      (int) k.shortValue();
     str    [ += ]     b;
     b      [ = ]      !b;
     c      [ *= ]     i;

   }
}
```

Explanation:

1. `i = (int) k.shortValue();` --> You can use *= here but then you can't complete the 4th line.

2. `str += b;` --> You can't use =, or *= here. Only += is valid.

3. `b = !b;` --> You can't use anything other than = here.

4. `c *= i;` --> You can only use *= or +=. = is not valid. Further, if you use += here, you can't complete line 2.

Back to Question without Answer

68. QID - 2.1004 : Java Basics

Which of these statements concerning the use of modifiers are true?

Correct Option is : B

~~**A.**~~ By default (i.e. no modifier) the member is only accessible to classes in the same package and subclasses of the class.

> No, the member will be accessible only within the package.

B. You cannot specify visibility of local variables.

> They are always only accessible within the block in which they are declared.

~~**C.**~~ Local variable always have default accessibility.

> A local variable (aka automatic variable) means a variable declared in a method. They don't have any accessibility. They are accessible only from the block they are declared in.
> Remember, they are not initialized automatically. You have to initialize them explicitly.

~~**D.**~~ Local variables can be declared as private.

~~**E.**~~ Local variables can only be declared as public.

Explanation:

You cannot apply any modifier except final to a local variable. i.e. you cannot make them transient, volatile, static, public, and private.

But you can apply access modifiers (public private and protected) and final, transient, volatile, static to instance variables.
You cannot apply native and synchronized to any kind of variable.

Back to Question without Answer

69. QID - 2.1254 : Handling Exceptions

What will the following code print when run?

```
public class Test{
 static String j = "";
 public static void method( int i){
  try{
   if(i == 2){
     throw new Exception();
   }
   j += "1";
  }
  catch (Exception e){
   j += "2";
   return;
  }
  finally{
   j += "3";
  }
  j += "4";
 }
 public static void main(String args[]){
  method(1);
  method(2);
  System.out.println(j);
 }
}
```

Correct Option is : B

~~**A.**~~ 13432

B. 13423

~~**C.**~~ 14324

~~**D.**~~ 12434

~~**E.**~~ 12342

Explanation:

Try to follow the flow of control :
1. in method(1) : i is not 2 so, j gets "1" then finally is executed which makes j = "13" and then the last statement (j +=4) is executed which makes j = "134".
2. in method(2) : i is 2, so it goes in the if block which throws an exception. So none of the statements of try block are executed and control goes to catch which makes j = "1342", then finally makes j = "13423" and the control is returned. Note that the last statement (j+=4) is not executed as there was an exception thrown in the try block, which cause the control to go to the catch block, which in turn has a return.

Back to Question without Answer

70. QID - 2.963 : Using Loop Constructs

Consider the following code snippet:

```
for(int i=INT1; i<INT2; i++){
    System.out.println(i);
}
```

INT1 and INT2 can be any two integers.

Which of the following will produce the same result?

Correct Option is : E

~~**A.**~~ `for(int i=INT1; i<INT2; System.out.println(++i));`

Prints: 2 and 3

~~**B.**~~ `for(int i=INT1; i++<INT2; System.out.println(i));`

Prints: 2 and 3

~~**C.**~~ `int i=INT1; while(i++<INT2) { System.out.println(i); }`

Prints: 2 and 3

~~**D.**~~ `int i=INT1; do { System.out.println(i); }while(i++<INT2);`

Prints: 1 2 and 3

E. None of these.

Explanation:

In such a question it is best to take a sample data such as INT1=1 and INT2=3 and execute the loops mentally. Eliminate the wrong options. In this case, the original loop will print:
=====ORIGINAL ====
1

2

Outputs of all the options are given above (Ignoring the line breaks).

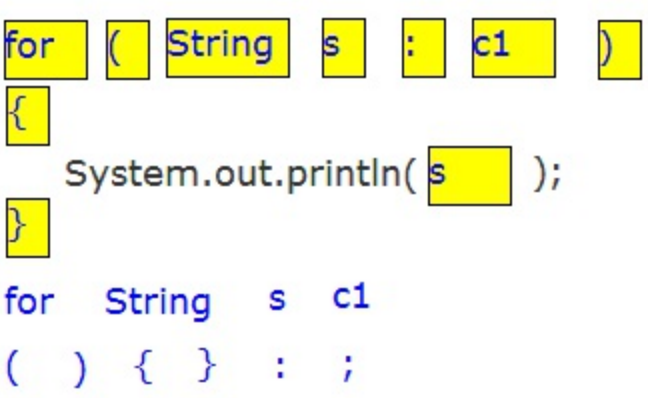
Thus, none of them is same as the original.

Back to Question without Answer

71. QID - 2.1043 : Using Loop Constructs

Assuming the following declarations, write a for loop that prints each string in the collection using the given elements.

```
Collection<String> c1 = new HashSet<String>();
```



Explanation:

This illustrates the basic usage of an enhanced for loop.

Back to Question without Answer

72. QID - 2.1131 : Java Basics

What would be the result of attempting to compile and run the following program?

```
class TestClass{
   static TestClass ref;
   String[] arguments;
   public static void main(String args[]){
      ref = new TestClass();
      ref.func(args);
   }
   public void func(String[] args){
      ref.arguments = args;
   }
}
```

Correct Option is : E

~~**A.**~~ The program will fail to compile, since the static method `main` is trying to call the non-static method `func`.

> The concept here is that a non-static method (i.e. an instance method) can only be called on an instance of that class. Whether the caller itself is a static method or not, is immaterial.
>
> The main method is calling `ref.func();` - this means the main method is calling a non-static method on an actual instance of the class TestClass (referred to by 'ref'). Hence, it is valid.
> It is not trying calling it directly such as `func()` or `this.func()`, in which case, it would have been invalid.

~~**B.**~~ The program will fail to compile, since the non-static method `func` cannot access the static member variable `ref`.

> Non static methods can access static as well as non static methods of a class.

~~**C.**~~ The program will fail to compile, since the argument `args` passed to the static method `main` cannot be passed on to the non-static method `func`.

> Non sense!

~~**D.**~~ The program will fail to compile, since method `func` is trying to assign to the non-static member variable '`arguments`' through the static member variable `ref`.

E. The program will compile and run successfully.

Back to Question without Answer

73. QID - 2.1214 : Creating and Using Arrays

Consider the following program:

```
public class TestClass{
  public static void main(String[] args){
    String tom = args[0];
    String dick = args[1];
    String harry = args[2];
  }
}
```

What will the value of 'harry' if the program is run from the command line:
java TestClass 111 222 333

Correct Option is : C

~~**A.**~~ 111

~~**B.**~~ 222

C. 333

~~**D.**~~ It will throw an `ArrayIndexOutOfBoundsException`

~~**E.**~~ None of the above.

Explanation:

java and classname are not part of the args array. So tom gets "111", dick gets "222" and harry gets "333".

Back to Question without Answer

74. QID - 2.943 : Java Basics

Note: Option 4 of this question may be considered too advanced for this exam.

Which lines of code will not be acceptable to the compiler?

```
import java.*; //1
public abstract class InternalLogic //2
{
   float density = 20.0; //3
   public class Doer //4
   {
      void do() //5
      {
         //lot of valid code.
      }
   }
}
```

Correct Options are : C E

~~**A.**~~ 1

Although it doesn't import anything but import java.*; is valid.

~~**B.**~~ 2

A class having no abstract method can still be abstract. But not vice-versa.

C. 3

20.0 is a double and needs a cast to be assigned to a float.

~~**D.**~~ 4

It is a valid inner class.

E. 5

do is a keyword so do() is not a valid method name.

Back to Question without Answer

75. QID - 2.966 : Working with Methods - Access Modifiers

Select the correct order of restrictiveness for access modifiers...
(First one should be least restrictive)

Correct Option is : A

A. public < protected < package (i.e. no modifier) < private

> That's right, protected is less restrictive than package.

~~**B.**~~ public < package (i.e. no modifier) < protected < private

~~**C.**~~ public < protected < private < package (i.e. no modifier)

> The default accessibility is more restrictive than protected, but less restrictive than private.

~~**D.**~~ protected < package (i.e. no modifier) < private < public

~~**E.**~~ depends on the implementation of the class or method.

Explanation:

Members with default accessibility are only accessible within the class itself and from classes in the same package.
Protected members are in addition accessible from subclasses. Members with private accessibility are only accessible
within the class itself.

Back to Question without Answer

76. QID - 2.1080 : Java Basics

Which of the following are valid declarations of the standard main() method?

Correct Options are : D E

~~**A.**~~ `static void main(String args[ ]) { }`

> Although practically correct but for the purpose of SCJP you should not select this option because the method is not declared public.

~~**B.**~~ `public static int main(String args[ ]) {}`

> This method returns an 'int' instead of 'void'.

~~**C.**~~ `public static void main (String args) { }`

> The method takes only one String instead of String[].

D. `final static public void main (String[ ] arguments ) { }`

E. `public static void main (String[ ] args) { }`

Explanation:

A valid declaration of the main() method must be 'public' and 'static', should return 'void', and should take a single array of Strings.
The order of the static and public keywords is irrelevant. But return type should always come just before the method name.
'final' does not change the method signature.
Also, in some of the JDK environments, even a private or protected main() method

may work however, for the purpose of Java Certification exam, it should be assumed that for the JVM to execute a class using the standard main method, the accessibility of the main method must be public.

Back to Question without Answer

77. QID - 2.1070 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
int i1 = 2;
int i2 = 3;
if (b1 = i1 == i2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Correct Option is : C

~~**A.**~~ Compile time error.

~~**B.**~~ It will print true

C. It will print false

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

All an if statement needs is a boolean. Now i1 == i2 returns false which is a boolean and since b1 = false is an expression and every expression has a return value (which is actually the Left Hand Side of the expression), it returns false which is again a boolean. Therefore, in this case, the else condition will be executed.

Back to Question without Answer

78. QID - 2.880 : Handling Exceptions

Which of the following are standard Java exception classes?

Correct Options are : A E

A. `FileNotFoundException`

~~B.~~ `InputException`

> There is an `java.io.IOException` but no InputException or OutputException.

~~C.~~ `CPUError`

~~D.~~ `MemoryException`

> There is an OutOfMemoryError but no MemoryException. There is also a StackOverflowError.

E. `SecurityException`

> Java has a `java.lang.SecurityException`. This exception extends `RuntimeException` and is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception.

Back to Question without Answer

79. QID - 2.1103 : Using Operators and Decision Constructs

What will be printed by the following code if it is run with command line: java TestClass -0.50 ?

```
public class TestClass{
    public static double getSwitch(String str){
        return Double.parseDouble(str.substring(1, str.length()-1)
);
    }
    public static void main(String args []){
        switch(getSwitch(args[0])){
            case 0.0 : System.out.println("Hello");
            case 1.0 : System.out.println("World"); break;
            default : System.out.println("Good Bye");
        }
    }
}
```

Correct Option is : E

~~**A.**~~ Hello

~~**B.**~~ World

~~**C.**~~ Hello World

~~**D.**~~ Hello World Good Bye

E. None of the above.

Explanation:

Observe that the method `getSwitch()` has been declared to return a double. Its return value is being used in the switch() statement. Therefore, the program will not even compile because double/float/long/boolean cannot be used in switch(...) statement.

Back to Question without Answer

80. QID - 2.912 : Working with Methods

Which of the following methods does not return any value?

Correct Option is : D

~~A.~~ `public doStuff() throws FileNotFoundException, IllegalArgumentException{ //valid code not shown }`

> It is missing the return type. Every method must have a return type specified in its declaration.
> It could be a valid constructor though if the class is named `doStuff` because the constructors don't return anything, not even `void`.

~~B.~~ `public null doStuff() throws FileNotFoundException, IllegalArgumentException{ //valid code not shown }`

> `null` can be a return value not a return type because null is not a type.

~~C.~~ `public doStuff() { //valid code not shown }`

> This is not a valid method because there is no return type declared. Although it can be a valid constructor if the name of the class is `doStuff`.

D. `public void doStuff() throws FileNotFoundException, IllegalArgumentException{ //valid code not shown }`

A method that does not return anything should declare its return type as `void`. Note that this is different from constructors. A constructor also doesn't return anything but for a constructor, you don't specify any return type at all. That is how a constructor is differentiated from a regular method.

E.
```
private doStuff() {
    //valid code not shown
}
```

This is not a valid method because there is no return type declared. Although it can be a valid constructor if the name of the class is `doStuff`.

Back to Question without Answer

81. QID - 2.1023 : Handling Exceptions

Identify the exceptions that will be received when the code snippets on the left hand side are executed.

```
1. int[] ia = new int[]{ 1, 2, 3};
   System.out.println(ia[-1]);
```

ArrayIndexOutOfBoundsException

```
2.
public class X {
static int k = 0;
static {
k = 10/0;
}
}
```

ExceptionInInitializerError

```
3. SomeClass sc = new SomeClass();
```
(Assume that SomeClass is not available in runtime classpath.)

NoClassDefFoundError

```
4.
public class X {
static {
if(true) throw new NullPointerException();
}
}
```

ExceptionInInitializerError

NoClassDefFoundError NullPointerException ArrayAccessException

ExceptionInInitializerError ArrayIndexOutOfBoundsException

IllegalArrayAccessException NoSuchClassException

Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

82. QID - 2.1250 : Working with Inheritance

Which statements, when inserted at line 1, will cause an exception at run time?

```
class B {}
class B1 extends B {}
class B2 extends B {}
public class ExtendsTest{
  public static void main(String args[]){
     B b = new B();
     B1 b1 = new B1();
     B2 b2 = new B2();
     // insert statement here
  }
}
```

Correct Option is : C

~~**A.**~~ `b = b1;`

> There won't be a problem anytime because B1 is a B

~~**B.**~~ `b2 = b;`

> It fails at Compile time as an object referenced by b may not be a B2, so an explicit cast will be needed.

C. `b1 = (B1) b;`

> It will pass at compile time but fail at run time as the actual object referenced by b is not a B1.

~~**D.**~~ `b2 = (B2) b1;`

It will not compile because b1 can never point to an object of class B2.

E. ~~`b1 = (B) b1;`~~

This won't compile. Another cast is needed. i.e. `b1 = (B1) (B) b1;`

Back to Question without Answer

83. QID - 2.976 : Java Basics

Consider the following two java files:

```
//in file SM.java
package x.y;
public class SM{
    public static void foo(){ };
}


//in file TestClass.java
//insert import statement here //1
public class TestClass{
   public static void main(String[] args){
      foo();
   }
}
```

What should be inserted at //1 so that TestClass will compile and run?

Correct Options are : C E

~~**A.**~~ `import static x.y.*;`

> x.y.* means all the classes in package x.y. Classes cannot be imported using "import static". You must do import x.y.* for importing class.
> Further, importing a class will not give you a direct access to the members of the class. You will need to do SM.foo(), if you import SM.

~~**B.**~~ `import static x.y.SM;`

> x.y.SM means you are trying to import class x.y.SM. A class cannot be imported statically.

C. `import static x.y.SM.foo;`

This is valid because this statement is importing the static member foo of class x.y.SM.

D. ~~`import static x.y.SM.foo();`~~

Even though foo is a method, you cannot put () in the import statement.

E. `import static x.y.SM.*;`

This is valid because this statement is importing all the static members of class x.y.SM.

Back to Question without Answer

84. QID - 2.1259 : Using Operators and Decision Constructs

Note: Although Wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of Wrapper classes.

What will be the output of the following program?

```
public class EqualTest{
   public static void main(String args[]){
      Integer i = new Integer(1) ;
      Long m = new Long(1);
      if( i.equals(m)) System.out.println("equal");    // 1
      else System.out.println("not equal");
   }
}
```

Correct Option is : B

~~**A.** equal~~

B. not equal

~~**C.** Compile time error at //1~~

~~**D.** Runtime error at //1~~

~~**E.** None of the above.~~

Explanation:

Signature of equals method is : `boolean equals(Object o);` So it can take any

object.
The equals methods of all wrapper classes first check if the two object are of same class or not. If not, they immediately return `false`. Hence it will print `not equal`.

Back to Question without Answer

85. QID - 2.931 : Working with Inheritance

Consider the following classes :

```
class A{
    public static void sM1() {  System.out.println("In base static"),
}
class B extends A{
Line 1 :   // public static void sM1() {  System.out.println("In sub
Line 2 :   // public  void sM1() {  System.out.println("In sub non-s
}
```

Which of the following statements are true?

Correct Options are : B C

~~**A.**~~ class B will not compile if line 1 is uncommented.

> static method sM1() can be shadowed by a static method sM1() in the subclass.

B. class B will not compile if line 2 is uncommented.

> static method cannot be overridden by a non-static method and vice versa

C. class B will not compile if line 1 and 2 are both uncommented.

~~**D.**~~ Only Option 2 is correct.

~~**E.**~~ Only Option 3 is correct.

Explanation:

Another concept (although not related to this question but about static methods) is that static methods are never overridden. They are HIDDEN or SHADOWED just like static or non-static fields. For example,

```
class A{
  int i = 10;
  public static void m1(){  }
  public void m2() { }
}
class B extends A{
  int i = 20;
  public static void m1() {  }
  public void m2() { }
}
```

Here, UNLIKE m2, m1() of B does not override m1() of A, it just shadows it, as proven by the following code:

```
A a  = new B();
System.out.println(a.i)  //will print 10 instead of 20
a.m1();  //will call A's m1
a.m2();  //will call B's m2 as m2() is not static and so overrides A
```

Back to Question without Answer

86. QID - 2.1233 : Using Loop Constructs

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        boolean flag = true;
        for(int i = 0; i < 3; i++){
            while(flag){
                c++;
                if(i>c || c>5) flag = false;
            }
        }
        System.out.println(c);
    }
}
```

Correct Option is : D

~~**A.** 3~~

~~**B.** 4~~

~~**C.** 5~~

D. 6

~~**E.** 7~~

Explanation:

In the first iteration of for loop, the while loop keeps running till c becomes 6. Now, for all next for loop iteration, the while loop never runs as the flag is false. So final value of c is 6.

Back to Question without Answer

87. QID - 2.1217 : Working with Methods - Access Modifiers

Consider the following classes in one file named A.java...

```
abstract class A{
   protected int m1(){ return 0; }
}
class B extends A{
   int m1(){ return 1; }
}
```

Which of the following statements are correct...

Correct Option is : B

~~**A.** The code will not compile as you cannot have more than 1 class in 1 file.~~

> You can. But only one class can be public.

B. The code will not compile because class B does not override the method m1() correctly.

> The overriding method cannot decrease the accessibility.

~~**C.** The code will not compile as A is an abstract class.~~

~~**D.** The code will not compile as A does not have any abstract method.~~

> You need not have any 'abstract' method to make a class abstract. Putting 'abstract' keyword is enough.

~~**E.** The code will compile fine.~~

Explanation:

The concept here is that an overriding method cannot make the overridden method more private.
The access hierarchy in increasing levels of accessibility is:
private->'no modifier'->protected->public (public is accessible to all and private is accessible to none except itself.)
Here, class B has no modifier for m1() so it is trying to reduce the accessibility of protected to default.
'protected' means the method will be accessible to all the classes in the same package and all the subclasses (even if the subclass is in a different package).
No modifier (which is the default level) means the method will be accessible only to all the classes in the same package. (i.e. not even to the subclass if the subclass is in a different package.)

Back to Question without Answer

88. QID - 2.946 : Creating and Using Arrays

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
    public static void main(String args[ ] ){
       int i = 1;
       int[] iArr = {1};
       incr(i) ;
       incr(iArr) ;
       System.out.println( "i = " + i + "  iArr[0] = " + iArr [ 0 ] );
    }
    public static void incr(int   n ) { n++ ; }
    public static void incr(int[ ] n ) { n [ 0 ]++ ; }
}
```

Correct Option is : B

~~**A.**~~ `The code will print i = 1 iArr[0] = 1;`

B. `The code will print i = 1 iArr[0] = 2;`

~~**C.**~~ `The code will print i = 2 iArr[0] = 1;`

~~**D.**~~ `The code will print i = 2 iArr[0] = 2;`

~~**E.**~~ The code will not compile.

There is no problem with the code.

Explanation:

Arrays are proper objects (i.e. `iArr instanceof Object` returns `true`) and Object references are passed by value (so effectively, it seems as though objects are being passed by reference).
So the value of reference of `iArr` is passed to the method `incr(int[] i);` This method changes the actual value of the int element at 0.

Back to Question without Answer

89. QID - 2.827 : Handling Exceptions

What will be the output when the following program is run?

```
package exceptions;
public class TestClass{
    public static void main(String[] args) {
        try{
            hello();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void hello() throws MyException{
        int[] dear = new int[7];
        dear[0] = 747;
        foo();
    }

    static void foo() throws MyException{
        throw new MyException("Exception from foo");
    }
}

class MyException extends Exception {
    public MyException(String msg){
        super(msg);
    }
}
```

(Assume that line numbers printed in the messages given below are correct.)

Correct Option is : C

~~**A.**~~ ~~Exception in thread "main"~~
~~java.lang.ArrayIndexOutOfBoundsException: 10~~

```
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

> You are creating an array of length 7. Since array numbering starts with 0, the first element would be array[0]. So ArrayIndexOutOfBoundsException will NOT be thrown.

B. ~~`Error in thread "main" java.lang.ArrayIndexOutOfBoundsException`~~

> java.lang.ArrayIndexOutOfBoundsException extends java.lang.RuntimeException, which in turn extends java.lang.Exception. Therefore, ArrayIndexOutOfBoundsException is an Exception and not an Error.

C. `exceptions.MyException: Exception from foo`

D. ~~`exceptions.MyException: Exception from foo`~~

```
    at exceptions.TestClass.foo(TestClass.java:29)
    at exceptions.TestClass.hello(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

> me.printStackTrace() would have produced this output.

Explanation:

Note that there are a few questions in the exam that test your knowledge about how exception messages are printed.

When you use `System.out.println(exception)`, a stack trace is not printed. Just the name of the exception class and the message is printed.
When you use `exception.printStackTrace()`, a complete chain of the names of the methods called, along with the line numbers, is printed from the point where the exception was thrown and up to the point where the exception was caught and `printStackTrace()` was called.

Back to Question without Answer

90. QID - 2.1181 : Working with Inheritance

What would be the result of attempting to compile and run the following code?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      B c = new C();
      System.out.println(c.max(10, 20));
   }
}
class A{
   int max(int x, int y)  { if (x>y) return x; else return y; }
}
class B extends A{
  int max(int x, int y)  {  return 2 * super.max(x, y) ; }
}
class C extends B{
  int max(int x, int y)  {  return super.max( 2*x, 2*y); }
}
```

Correct Option is : C

~~**A.**~~ The code will fail to compile.

~~**B.**~~ Runtime error.

C. The code will compile without errors and will print 80 when run.

~~**D.**~~ The code will compile without errors and will print 40 when run.

~~**E.**~~ The code will compile without errors and will print 20 when run.

Explanation:

When the program is run, the `main()` method will call the `max()` method in C with parameters 10 and 20 because the actual object referenced by 'c' is of class C. This method will call the `max()` method in B with the parameters 20 and 40. The `max()` method in B will in turn call the `max()` method in A with the parameters 20 and 40. The `max()` method in A will return 40 to the `max()` method in B. The `max()` method in B will return 80 to the `max()` method in C. And finally the `max()` of C will return 80 in `main()` which will be printed out.

Back to Question without Answer

Last Day Test (Unique)

Take this test when you are all done with your preparation and are ready for the actual exam. Questions in this test are completely unique in the sense that these are not included in "Practice Tests" or "Objectivewise Tests".

01. QID - 2.840

What will be the output of the following program when it is compiled and run with the command line:

```
java TestClass 1 2 3

public class TestClass {

    public static void main(String[] args) {
        System.out.println("Values : "+args[0]+args[1]);
    }
}
```

Select 1 option

A. Values : java TestClass

B. Values : TestClass 1

C. Values : 12

D. Values : 23

E. Values : 3

Check Answer

02. QID - 2.1366

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        while(int k = 5; k<7){
            System.out.println(k++);
        }
    }
}
```

Select 1 option

A. 5
6

B. 5
6
7

C. It will keep printing 5.

D. It will not compile.

E. It will throw an exception at run time.

Check Answer

03. QID - 2.878

What will the following code print?

```
int[] scores = { 1, 2, 3, 4, 5, 6};
System.arraycopy(scores, 2, scores, 3, 2);
for(int i :  scores) System.out.print(i);
```

Select 1 option

A. `123446`

B. `123356`

C. `1233456`

D. `123346`

E. `123336`

Check Answer

04. QID - 2.834

Consider the following

```
public class TestClass {
    public static void main(String[] args) {
        TestClass tc = new TestClass();
        tc.myMethod();
    }

    public void myMethod() {
        yourMethod();
    }

    public void yourMethod() {
        throw new Exception();
    }
}
```

What changes can be done to make the above code compile?

Select 1 option

A. Change declaration of main to :

```
public static void main(String[] args) throws Exception
```

B. Change declaration of myMethod to

```
public void myMethod throws Exception
```

C. Change declaration of yourMethod to

```
public void yourMethod throws Exception
```

D. Change declaration of main and yourMethod to :

`public static void main(String[] args) throws Exception` and `public void yourMethod throws Exception`

E. Change declaration of all the three method to include `throws Exception`.

Check Answer

05. QID - 2.889

What will the following code print when run?

```
class A{
    String value = "test";
    A(String val){
        this.value = val;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        new A("new test").print();
    }
}
```

Select 1 option

A. test

B. new test

C. It will not compile.

D. It will throw an exception at run time.

Check Answer

06. QID - 1.926

Consider the following class...

```
class TestClass{
    void probe(Object x) { System.out.println("In Object"); } //3

    void probe(Number x) { System.out.println("In Number"); } //2

    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(Long x) { System.out.println("In Long"); } //4

    public static void main(String[] args){
        double a = 10;
        new TestClass().probe(a);
    }
}
```

What will be printed?

Select 1 option

A. In Number

B. In Object

C. In Long

D. In Integer

E. It will not compile.

Check Answer

07. QID - 2.1357

Which of these combinations of switch expression types and case label value types are legal within a switch statement?

Select 1 option

A. switch expression of type int and case label value of type char.

B. switch expression of type float and case label value of type int.

C. switch expression of type byte and case label value of type float.

D. switch expression of type char and case label value of type byte.

E. switch expression of type boolean and case label value of type boolean.

Check Answer

08. QID - 2.1176

Which code fragments will print the last argument given on the command line to the standard output, and exit without any output and exceptions if no arguments are given?

```
1.
 public static void main(String args[ ]){
       if (args.length != 0)   System.out.println(args[args.length-1
 }

2.
public static void main(String args[ ]){
       try {      System.out.println(args[args.length-1]);        }
       catch (ArrayIndexOutOfBoundsException e) {    }
}

3.
 public static void main(String args[ ]){
     int i = args.length;
     if (i != 0) System.out.println(args[i-1]);
}

4.
public static void main(String args[ ]){
    int i = args.length-1;
   if (i > 0) System.out.println(args[i]);
}

5.
 public static void main(String args[ ]){
       try { System.out.println(args[args.length-1]); }
       catch (NullPointerException e) {}
 }
```

Select 3 options

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

09. QID - 2.1224

Consider the following code for the main() method:

```
public static void main(String[] args) throws Exception{
   int i = 1, j = 10;
   do {
      if (i++ > --j) continue;
   } while (i < 5);
   System.out.println("i=" + i + " j=" + j);
}
```

What will be the output when the above code is executed?

Select 1 option

A. i=6 j=6

B. i=5 j=6

C. i=5 j=5

D. i=6 j=5

E. None of these.

Check Answer

10. QID - 2.1253

Where, in a constructor, can you place a call to a super class's constructor ?

Select 1 option

A. Anywhere in the constructor's body.

B. As the first statement in the constructor.

C. Only as the first statement and it can be called just like any other method call i.e. ClassName(...).

D. You can't call super class's constructor in a base class as constructors are not inherited..

E. None of the above.

Check Answer

11. QID - 2.1339

What will the following code print?

```
int i = 1;
int j = i++;
if( (i==++j) | (i++ == j) ){
  i+=j;
}
System.out.println(i);
```

Select 1 option

A. 3

B. 4

C. 5

D. 2

E. It will not compile.

Check Answer

12. QID - 2.1314

Which statements concerning conversion are true?

Select 4 options

A. Conversion from char to long does not need a cast.

B. Conversion from byte to short does not need a cast.

C. Conversion from short to char needs a cast.

D. Conversion from int to float need a cast.

E. Conversion from byte, char or short to int, long or float does not need a cast.

Check Answer

13. QID - 2.1334

What will the following program print when run?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[] ){ A b = new B("good bye");
}
class A{
   A() { this("hello", " world");  }
   A(String s) { System.out.println(s); }
   A(String s1, String s2){ this(s1 + s2); }
}
class B extends A{
   B(){ super("good bye"); };
   B(String s){ super(s, " world "); }
   B(String s1, String s2){ this(s1 + s2 + " ! "); }
}
```

Select 1 option

A. It will print "good bye".

B. It will print "hello world".

C. It will print "good bye world".

D. It will print "good bye" followed by "hello world".

E. It will print "hello world" followed by "good bye".

Check Answer

14. QID - 2.1268

Which of these for statements are valid?

```
1. for (int i=5; i=0; i--) { }

2.  int j=5;
      for(int i=0, j+=5;  i<j ; i++) {  j--;  }

3. int i, j;
    for (j=10; i<j; j--) { i += 2; }

4. int i=10;
    for ( ; i>0 ; i--) { }

5. for (int i=0, j=10; i<j; i++, --j) {;}
```

Select 1 option

A. 1, 2

B. 3, 4

C. 1, 5

D. 4, 5

E. 5

Check Answer

15. QID - 2.862

Consider the following code appearing in the same file:

```
class Data {
    int x = 0, y = 0;
    public Data(int x, int y){
        this.x = x; this.y = y;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d = new Data(1, 1);
        //add code here
    }
}
```

Which of the following options when applied individually will change the Data object currently referred to by the variable d to contain 2, 2 as values for its data fields?

Select 2 options

A. Add the following two statements:

```
d.x = 2;
d.y = 2;
```

B. Add the following statement:

```
d = new Data(2, 2);
```

C. Add the following two statements:

```
d.x += 1;
d.y += 1;
```

D. Add the following statement:

```
d = d + 1;
```

Check Answer

16. QID - 2.864

Java Exceptions is a mechanism ..

Select 2 options

A. for dealing with unexpected user inputs.

B. that you can use to determine what to do when something unexpected happens.

C. for logging unexpected behavior.

D. to ensure that the program runs even if something unexpected happens.

E. that the VM uses to exit the program when something unexpected happens.

Check Answer

17. QID - 2.839

How can you fix the following code to make it compile:

```
import java.io.*;
class Great {
    public void doStuff() throws FileNotFoundException{
    }
}

class Amazing extends Great {
  public void doStuff() throws IOException, IllegalArgumentException
  }
}

public class TestClass {
    public static void main(String[] args) throws IOException{
        Great g = new Amazing();
        g.doStuff();
    }
}
```

Assume that changes suggested in a option are to be applied independent of other options.

Select 2 options

A. Change `doStuff` in Amazing to throw only `IllegalArgumentException`.

B. Change `doStuff` in `Great` to throw only `FileNotFoundException` as well as `IllegalArgumentException`.

C. Change `doStuff` in `Amazing` to throw only `IOException`.

D. Change `doStuff` in `Great` to throw only `IOException` instead of `FileNotFoundException`.

E. Replace `g.doStuff()` to `((Amazing) g).doStuff()`.

Check Answer

18. QID - 2.925

Given:

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<Double> al = new ArrayList<>();

        //INSERT CODE HERE
     }
}
```

What can be inserted in the above code so that it can compile without any error?

Select 3 options

A. `al.add(111);`

B. `System.out.println(al.indexOf(1.0));`

C. `System.out.println(al.contains("string"));`

D. `Double d = al.get(al.length);`

E. `al.notifyAll();`

Check Answer

19. QID - 2.908

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {

          boolean flag  = true;
          switch (flag){
              case true : System.out.println("true");
                  default: System.out.println("false");
          }

    }
}
```

Select 1 option

A. It will not compile.

B. false

C. true
false

D. Exception at run time.

Check Answer

20. QID - 2.871

What will the following code print when compiled and run?

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<Integer> al = new ArrayList<>(); //1
        al.add(111); //2
        System.out.println(al.get(al.size()));  //3
     }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at run time because of line //1

C. It will throw an exception at run time because of line //2

D. It will throw an exception at run time because of line //3

E. null.

Check Answer

21. QID - 2.853

Which line in the following code will cause the compilation to fail?

```
public class TestClass {

    public static void main(String[] args) throws Exception {
       work();                    //LINE 10
       int j = j1;                 //LINE 11
       int j1 = (double) x; //LINE 12
    }

    public static void work() throws Exception{
        System.out.println(x); //LINE 15
    }

    static double x;    //19
}
```

Select 1 option

A. Line 10

B. Line 11

C. Line 12

D. Line 15

E. Line 19

Check Answer

22. QID - 2.846

Consider the following code:

```
public class TestClass {

    public static void doStuff() throws Exception{
        System.out.println("Doing stuff...");
        if(Math.random()>0.4){
            throw new Exception("Too high!");
        }
        System.out.println("Done stuff.");
    }

    public static void main(String[] args) throws Exception {
        doStuff();
        System.out.println("Over");
    }
}
```

Which of the following are possible outputs when the above program is compiled and run? (Assume that Math.random() returns a double between 0.0 and 1.0 not including 1.0. Further assume that there is no mistake in the line numbers printed in the output.)

Select 2 options

A.

```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
 at TestClass.doStuff(TestClass.java:29)
 at TestClass.main(TestClass.java:41)
```

B.

```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
 at TestClass.doStuff(TestClass.java:29)
 at TestClass.main(TestClass.java:41)
Over
```

C.

```
Doing stuff...
Done stuff.
Over
```

D.

```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
 at TestClass.doStuff(TestClass.java:29)
 at TestClass.main(TestClass.java:41)
Done stuff.
```

Check Answer

23. QID - 2.1138

What will the following program print when run?

```
class Super{
  public String toString(){
     return "4";
  }
}
public class SubClass extends Super{
  public String toString(){
     return super.toString()+"3";
  }
  public static void main(String[] args){
    System.out.println( new SubClass() );
  }
}
```

Select 1 option

A. 43

B. 7

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

24. QID - 2.933

Which of the given options can be successfully inserted at line 1....

```
//line 1
   public class A{
   }
```

Select 3 options

A. `import java.lang.*;`

B. `package p.util;`

C. `public class MyClass{ }`

D. `abstract class MyClass{ }`

Check Answer

25. QID - 2.1011

Consider the following code:

```
public class TestClass{
   public void method(Object o){
      System.out.println("Object Version");
   }
   public void method(java.io.FileNotFoundException s){
      System.out.println("java.io.FileNotFoundException Version");
   }
   public void method(java.io.IOException s){
      System.out.println("IOException Version");
   }
   public static void main(String args[]){
      TestClass tc = new TestClass();
      tc.method(null);
   }
}
```

What would be the output when the above program is compiled and run? (Assume that FileNotFoundException is a subclass of IOException, which in turn is a subclass of Exception)

Select 1 option

A. It will print `Object Version`

B. It will print `java.io.IOException Version`

C. It will print `java.io.FileNotFoundException Version`

D. It will not compile.

E. It will throw an exception at runtime.

Check Answer

26. QID - 2.869

Which of the following are true about the enhanced for loop?

Select 3 options

A. It can iterate over an array or a Collection but not a Map.

B. Using an enhanced for loop prevents the code from going into an infinite loop.

C. Using an enhanced for loop on an array may cause infinite loop.

D. An enhanced for loop can iterate over a Map.

E. You cannot find out the number of the current iteration while iterating.

Check Answer

27. QID - 2.1024

Given the following code fragment, which of the following lines would be a part of the output?

```
outer:
   for ( int i = 0 ; i<3 ; i++ ){
      for ( int j = 0 ; j<2 ; j++ ){
         if ( i == j ){
            continue outer;
         }
         System.out.println( "i=" + i + " , j=" + j );
      }
   }
```

Select 2 options

A. i = 1, j = 0

B. i = 0, j = 1

C. i = 1, j = 2

D. i = 2, j = 1

E. i = 2, j = 2

Check Answer

28. QID - 2.899

Given:

```
interface I { }

class A implements I{
    public String toString(){ return "in a"; }
}

class B extends A{
    public String toString(){ return "in b"; }
}


public class TestClass {

    public static void main(String[] args) {
        B b = new B();
        A a = b;
        I i = a;

        System.out.println(i);
        System.out.println((B)a);
        System.out.println(b);

    }
}
```

What will be printed when the above code is compiled and run?

Select 1 option

A. in i
in a
in b

B. I
A
in b

C. in a
in a
in b

D. in a
in b
in b

E. in b
in b
in b

Check Answer

29. QID - 2.991

What can be the return type of method getSwitch so that this program compiles and runs without any problems?

```
public class TestClass{
   public static XXX getSwitch(int x){
      return x - 20/x + x*x;
   }
   public static void main(String args[]){
       switch( getSwitch(10) ){
          case 1 :
          case 2 :
          case 3 :
          default : break;
       }
   }
}
```

Select 1 option

A. int

B. float

C. long

D. double

E. char

F. byte

G. short

Check Answer

30. QID - 2.1290

Consider the following method:

```
static int mx(int s){
    for(int i=0; i<3; i++){
        s = s + i;
    }
    return s;
}
```

and the following code snippet:

```
int s = 5;
    s += s + mx(s) + ++s;
    System.out.println(s);
```

What will it print?

Select 1 option

A. 21

B. 22

C. 23

D. 24

E. 25

F. 26

Check Answer

31. QID - 2.1198

What will be the output of the following program ?

```
class CorbaComponent{
    String ior;
    CorbaComponent(){ startUp("IOR"); }
    void startUp(String s){ ior  =  s; }
    void print(){ System.out.println(ior); }
}

class OrderManager extends CorbaComponent{
   OrderManager(){  }
   void startUp(String s){  ior = getIORFromURL(s);   }
   String getIORFromURL(String s){  return "URL://"+s; }
}

public class Application{
   public static void main(String args[]){ start(new OrderManager()),
   static void start(CorbaComponent cc){ cc.print(); }
}
```

Select 1 option

A. It will throw an exception at run time.

B. It will print `IOR`

C. It will print `URL://IOR`

D. It will not compile.

E. None of the above.

Check Answer

32. QID - 2.951

Consider the following program...

```
class Super {  }
class Sub extends Super {  }
public class TestClass{
   public static void main(String[] args){
      Super s1 = new Super(); //1
      Sub s2 = new Sub();     //2
      s1 = (Super) s2;        //3
   }
}
```

Which of the following statements are correct?

Select 1 option

A. It will compile and run without any problems.

B. It will compile but WILL throw ClassCastException at runtime.

C. It will compile but MAY throw ClassCastException at runtime.

D. It will not compile.

E. None of the above.

Check Answer

33. QID - 2.857

Consider the following code:

```
public class TestClass {

    //define tester method here

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        while(tc.tester()){
            System.out.println("running...");
        }
    }
}
```

Which of the following options would be a valid implementation of tester() method?

Select 2 options

A.
```
public boolean tester(){
    return false;
}
```

B.
```
public Boolean tester(){
    return false;
}
```

C.
```
public tester(){
    return false;
}
```

D.
```
public int tester(){
    return 0;
```

```
}
```

E.

```
public String tester(){
    return "false";
}
```

Check Answer

34. QID - 2.929

Consider the following lines of code:

```
boolean greenLight = true;
boolean pedestrian = false;
boolean rightTurn = true;
boolean otherLane = false;
```

You can go ahead only if the following expression evaluates to 'true' :

```
(( (rightTurn && !pedestrian || otherLane) || ( ? && !pedestrian &&
greenLight ) )  == true )
```

What variables can you put in place of '?' so that you can go ahead?

Select 1 option

A. rightTurn

B. otherLane

C. Any variable would do.

D. None of the variable would allow to go.

Check Answer

35. QID - 2.1194

Consider the following lines of code:

```
System.out.println(null + true); //1
System.out.println(true + null); //2
System.out.println(null + null); //3
```

Which of the following statements are correct?

Select 1 option

A. None of the 3 lines will compile.

B. All the 3 line will compile and print `nulltrue`, `truenull` and `nullnull` respectively.

C. Line 1 and 2 won't compile but line 3 will print `nullnull`.

D. Line 3 won't compile but line 1 and 2 will print `nulltrue` and `truenull` respectively.

E. None of the above.

Check Answer

36. QID - 2.1078

The following code snippet will print true.

```
String str1 = "one";
String str2 = "two";
System.out.println( str1.equals(str1=str2) );
```

Select 1 option

A. True

B. False

Check Answer

37. QID - 2.918

Given:

```
int a = 1 + 2 + 3 * 4;
int b = 2 * 3 + 4;

int total = a + b;
```

What will be the value of total?

Select 1 option

A. 34

B. 38

C. 29

D. 25

Check Answer

38. QID - 2.953

Which of the following are also called as "short circuiting logical operators"?

Select 2 options

A. &

B. ||

C. &&

D. |

E. ^

Check Answer

39. QID - 2.1371

Given the following declarations:

```
int a = 5, b = 7, k = 0;
Integer m = null;
```

and the following statements:

```
k = new Integer(a) + new Integer(b);  //1
k = new Integer(a) + b; //2
k = a + new Integer(b); //3
m = new Integer(a) + new Integer(b); //4
```

Executed independent of each other, what will be the value of k (for //1, //2, and //3) and m (for //4) after execution of each of these statements?

Select 1 option

A. 12
will not compile
will not compile
12

B. will not compile
will not compile
will not compile
12

C. 12
12
12
12

D. will not compile
will not compile
will not compile
will not compile

E. 12
12
12
will not compile

Check Answer

40. QID - 2.1175

Which of the following statements will evaluate to true?

Select 1 option

A. `"String".replace('g','G') == "String".replace('g','G')`

B. `"String".replace('g','g') == new String("String").replace('g','g')`

C. `"String".replace('g','G')=="StrinG"`

D. `"String".replace('g','g')=="String"`

E. None of these.

Check Answer

41. QID - 2.1169

Which of the following can be used as a constructor for the class given below?

```
public class TestClass{
   // lots of code ...
}
```

Select 2 options

A. `public void TestClass() {...}`

B. `public TestClass() {...}`

C. `public static TestClass() {...}`

D. `public final TestClass() {...}`

E. `public TestClass(int x) { ...}`

Check Answer

42. QID - 2.875

What will the following code print when compiled and run:

```
class Data {

    int intVal = 0;
    String strVal = "default";
    public Data(int k){
        this.intVal = k;
    }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d1 = new Data(10);
        d1.strVal = "D1";
        Data d2 = d1;
        d2.intVal = 20;
        System.out.println("d2 val = "+d2.strVal);
    }
}
```

Select 1 option

A. d2 val =

B. d2 val = default

C. d2 val = D1

D. Exception at run time.

Check Answer

43. QID - 2.1090

What is the effect of compiling and running the code shown in exhibit?

```
public class TestClass{
   public static void main (String args []){
      int sum = 0;
      for (int i = 0, j = 10; sum > 20; ++i, --j)       // 1
      {
         sum = sum+ i + j;
      }
      System.out.println("Sum = " + sum);
   }
}
```

Select 1 option

A. Compile time error at line 1.

B. It will print Sum = 0

C. It will print Sum = 20

D. Runtime error.

E. None of the above.

Check Answer

44. QID - 2.855

What will be printed when the following code is compiled and run?

```
public class LoadTest{

    public static void main(String[] args) throws Exception {
         LoadTest t = new LoadTest();
         int i = t.getLoad();
         double d = t.getLoad();
         System.out.println( i + d );
    }

    public int getLoad() {
        return 1;
    }

    public double getLoad(){
        return 3.0;
    }

}
```

Select 1 option

A. 13.0

B. 4.0

C. 4

D. The code will not compile.

Check Answer

45. QID - 2.844

Consider the following code:

```
import java.util.ArrayList;

public class Student{

    ArrayList<Integer> scores;
    private double average;

    public ArrayList<Integer> getScores(){ return scores; }

    public double getAverage(){ return average; }

    private void computeAverage(){
        //valid code to compute average
        average =//update average value
    }

    public Student(){
        computeAverage();
    }
}
```

What can be done to improve the encapsulation of this class?

Select 2 options

A. Make the class private.

B. Make the `scores` instance field private.

C. Make `getScores()` protected.

D. Make `computeAverage()` public.

E. Change getScores to return a copy of the scores list:

```
public ArrayList<Integer> getScores(){
   return new ArrayList(scores);
}
```

Check Answer

46. QID - 2.856

Consider the following code:

```
public class MyClass {

  protected int value = 10;

}
```

Which of the following statements are correct regarding the field value?

Select 1 option

A. It cannot be accessed from any other class.

B. It can be read but cannot be modified from any other class.

C. It can be modified but only from a subclass of MyClass.

D. It can be read and modified from any class within the same package or from any subclass of MyClass.

Check Answer

47. QID - 2.886

Given:

```
class StaticTest{

    void m1(){
        StaticTest.m2();  // 1
        m4();             // 2
        StaticTest.m3();  // 3
    }

    static void m2(){ }  // 4

    void m3(){
        m1();             // 5
        m2();             // 6
        StaticTest.m1(); // 7
    }

    static void m4(){ }
}
```

Which of the lines will fail to compile?

Select 2 options

A. 1

B. 2

C. 3

D. 4

E. 5

F. 6

G. 7

Check Answer

48. QID - 2.841

You have a method that currently does not handle any exception thrown from the code contained in its method body. You are now changing this method to call another method that throws `IOException`.

What changes, independent of each other, can you make to your method so that it will compile?

Select 2 options

A. Set the exception to `null` and don't rethrow it.

B. Declare `IOException` in the throws clause of your method.

C. Wrap the call to another method within a try-catch block that catches `RuntimeException`.

D. Wrap the call to another method within a try-catch block that catches `Exception`.

Check Answer

49. QID - 2.911

Given the complete contents of TestClass.java file:

```
package x;
public class TestClass {
    ArrayList<String> al;
    public void init(){
        al = new ArrayList<>();
        al.add("Name 1");
        al.add("Name 2");
    }
    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.init();
        System.out.println("Size = "+tc.al.size());
    }
}
```

Which import statement should be added to make it compile?

Select 1 option

A. `import java.lang.*;`

B. `import java.lang.ArrayList;`

C. `import java.util.ArrayList;`

D. `import java.collections.ArrayList;`

E. No import is necessary.

Check Answer

50. QID - 2.882

What will the following program print when run without any command line argument?

```
public class TestClass {
    public static void main(String[] args)  {

        boolean hasParams = (args == null ? false : true);
        if(hasParams){
            System.out.println("has params");
        }{
            System.out.println("no params");
        }
    }
}
```

Select 1 option

A. has params

B. has params
no params

C. no params

D. It will not compile.

Check Answer

51. QID - 2.828

Consider the following code appearing in a file named TestClass.java:

```
class Test{ }  // 1

public class TestClass {

   public int main(String[] args)  { // 2

        double x=10, double y;  // 3

        System.out.println[]; // 4

         for(int k =0; k<x; k++){ }

        return 0;
   }
}
```

Which of the lines are invalid?

Select 1 option

A. // 1 and // 4

B. // 3 and // 4

C. // 2 and // 4

D. // 2 and // 3

Check Answer

52. QID - 2.1185

What will the following code print?

```
String s = "blooper";
StringBuilder sb = new StringBuilder(s);
sb.append(s.substring(4)).delete(3, 5);
System.out.println(sb);
```

Select 1 option

A. `blorbloo`

B. `bloper`

C. `bloerper`

D. `blooperper`

E. `bloo`

Check Answer

53. QID - 2.1144

Consider the following interface definition:

```
public interface ConstTest{
 public int A = 1; //1
 int B = 1;          //2
 static int C = 1;  //3
 final int D = 1;   //4
 public static int E = 1; //5
 public final int F = 1;  //6
 static final int G = 1;    //7
 public static final int H = 1; //8
}
```

Which line(s) will cause a compilation error?

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. 5

F. 6

G. 7

H. 8

I. None of them will cause any error.

Check Answer

54. QID - 2.985

Which of the following declarations are valid?

Select 3 options

A. `float f1 = 1.0;`

B. `float f = 43e1;`

C. `float f = -1;`

D. `float f = 0x0123;`

E. `float f = 4;`

Check Answer

55. QID - 2.1372

Identify the exceptions that will be received when the following code snippets are executed.

1.
```
int factorial(int n){
       if(n==1) return 1;
      else return n*factorial(n-1);
   }
```
Assume that it is called with a very large integer.

2.
```
void printMe(Object[] oa){
       for(int i=0; i<=oa.length; i++)
       System.out.println(oa[i]);
    }
```
Assume that it is called as such: `printMe(null);`

3.
```
Object m1(){
       return new Object();
    }
    void m2(){
       String s = (String) m1();
    }
```

Select 1 option

A. ClassCastException
ArrayIndexOutOfBoundsException
StackOverflowError

B. ClassCastException
ArrayIndexOutOfBoundsException
SecurityException

C. No Exception Will Be Thrown
SecurityException
Will Not Compile

D. StackOverflowError
NullPointerException
No Exception Will Be Thrown

E. StackOverflowError
ArrayIndexOutOfBoundsException
ClassCastException

F. StackOverflowError
NullPointerException
NullPointerException

G. SecurityException
NullPointerException
No Exception Will Be Thrown

H. StackOverflowError
NullPointerException
ClassCastException

Check Answer

56. QID - 2.1160

Given that `OurClass` is a `MyClass` and `OurClass` has a `YourClass` object. Which of the following options are correct?

(Assume and `OurClass, MyClass,` and `YourClass` are valid java classes.)

Select 2 options

A. MyClass contains a reference to OurClass

B. OurClass contains a reference to MyClass

C. MyClass contains a reference to YourClass

D. OurClass contains a reference to YourClass

E. OurClass inherits from MyClass

Check Answer

57. QID - 2.1167

Given that SomeException is a checked exception, consider the following code:

```
//in file A.java
public class A{
   protected void m() throws SomeException{}
}

//in file B.java
public class B extends A{
   public void m(){ }
}

//in file TestClass.java
public class TestClass{
   public static void main(String[] args){
      // insert code here. // 1
   }
}
```

Which of the following options can be inserted at //1?

Select 1 option

A.
```
B b =  new A();
b.m();
```

B.
```
A a = new B();
a.m();
```

C.
```
A a = new B();
( ( B) a ).m();
```

D.
```
Object o = new B();
```

```
o.m();
```

E. None of these.

Check Answer

58. QID - 2.1200

What can be done to get the following code to compile and run? (Assume that the options are independent of each other.)

```
public float parseFloat( String s ){
     float f = 0.0f;       // 1
     try{
          f = Float.valueOf( s ).floatValue();     // 2
          return f ;       // 3
     }
     catch(NumberFormatException nfe){
        f = Float.NaN ;     // 4
       return f;      // 5
     }
     finally {
         return f;      // 6
     }
     return f ;     // 7
 }
```

Select 4 options

A. Remove line 3, 6

B. Remove line 5

C. Remove line 5, 6

D. Remove line 7

E. Remove line 3, 7

Check Answer

59. QID - 2.1352

Using a `continue` in a while loop causes the loop to break the current iteration and start the next iteration of the loop.

Select 1 option

A. True

B. False

Check Answer

60. QID - 2.1075

Given the following source code, which of the lines that are commented out may be reinserted without introducing errors?

```
abstract class Bang{
 //abstract void f();  //(0)
   final    void g(){}
 //final    void h(){} //(1)
   protected static int i;
   private int j;
}

final class BigBang extends Bang{
 //BigBang(int n) { m = n; } //(2)
   public static void main(String args[]){
      Bang mc = new BigBang();
   }
   void h(){}
 //void k(){ i++; } //(3)
 //void l(){ j++; } //(4)
   int m;
}
```

Select 1 option

A. `final void h( ) { } //(1)`

B. `BigBang(int n) { m = n; }  //(2)`

C. `void k( ) { i++ }   //(3)`

D. `void l( ) { j++ }  //(4)`

E. `abstract void f( ) ;  //(0)`

Check Answer

61. QID - 2.1128

What will the following code print?

```
int i = 0;
int j = 1;
if( (i++ == 0) & (j++ == 2) ){
   i = 12;
}
System.out.println(i+" "+j);
```

Select 1 option

A. 1 2

B. 2 3

C. 12 2

D. 12 1

E. It will not compile.

Check Answer

62. QID - 2.1230

What will the following code print when run?

```
public class TestClass{
  public static long main(String[] args){
     System.out.println("Hello");
     return 10L;
  }
}
```

Select 1 option

A. Hello

B. It will print nothing.

C. It will not compile

D. It will throw a Throwable at runtime.

E. None of the above.

Check Answer

63. QID - 2.1000

What will the following code print when run?

```
public class Test {

    static String s = "";

    public static void m0(int a, int b) {
        s += a;
        m2();
        m1(b);
    }

    public static void m1(int i) {
        s += i;
    }

    public static void m2() {
        throw new NullPointerException("aa");
    }

    public static void m() {
        m0(1, 2);
        m1(3);
    }

    public static void main(String args[]) {
        try {
            m();
        } catch (Exception e) {
        }
        System.out.println(s);
    }
}
```

Select 1 option

A. 1

B. 12

C. 123

D. 2

E. It will throw exception at runtime.

Check Answer

64. QID - 2.1051

Assuming that a valid integer will be passed in the command line as first argument, which statements regarding the following code are correct?

```
public class TestClass{
   public static void main(String args[]){
      int x = Integer.parseInt(args[0]);
      switch(x){
         case x < 5 :   System.out.println("BIG"); break;
         case x > 5 :   System.out.println("SMALL");
         default :    System.out.println("CORRECT"); break;
      }
   }
}
```

Select 1 option

A. `BIG` will never be followed by `SMALL`.

B. `SMALL` will never follow anything else.

C. `SMALL` will always be followed by `CORRECT`.

D. It will not compile.

E. It will throw an exception at runtime.

Check Answer

65. QID - 2.1295

Given the following class definitions and declaration:

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}

D d = new D();
```

the expression `(d instanceof A)` will return `true`.

Select 1 option

A. True

B. False

Check Answer

66. QID - 2.1291

What will be the result of attempting to compile and run class B?

```
class A{
   final int fi = 10;
}
public class B extends A{
   int fi = 15;
   public static void main(String[] args){
       B b = new B();
       b.fi = 20;
       System.out.println(b.fi);
       System.out.println(  (  (A) b  ).fi  );
   }
}
```

Select 1 option

A. It will not compile.

B. It will print 10 and then 10

C. It will print 20 and then 20

D. It will print 10 and then 20

E. It will print 20 and then 10

Check Answer

67. QID - 2.1182

What will be the output of the following class:

```
public class TestClass{
  public void testRefs(String str, StringBuilder sb){
    str = str + sb.toString();
    sb.append(str);
    str = null;
    sb = null;
  }
  public static void main(String[] args){
    String s = "aaa";
    StringBuilder sb = new StringBuilder("bbb");
    new TestClass().testRefs(s, sb);
    System.out.println("s="+s+" sb="+sb);
  }
}
```

Select 1 option

A. `s=aaa sb=bbb`

B. `s=null sb=null`

C. `s=aaa sb=null`

D. `s=null sb=bbbaaa`

E. `s=aaa sb=bbbaaabbb`

Check Answer

68. QID - 2.1166

Which line of code will not be acceptable to the compiler?

```
public class XBox{
  volatile int root = 20; //1
  private XBox() //2
  {
    volatile int i = 30; //3
  }
  private void XBox() //4
  {
    int local = 30;
  }
}
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. The code will compile fine.

Check Answer

69. QID - 2.1257

Which of the following statements concerning the switch construct are true?

Select 3 options

A. A character literal can be used as a value for a case label.

B. A 'long' cannot be used as a switch variable.

C. An empty switch block is a valid construct.

D. A switch block must have a default label.

E. If present, the default label must be the last of all the labels.

Check Answer

70. QID - 2.1158

What will the following program print?

```
public class InitTest{
   public InitTest(){
      s1 = sM1("1");
   }
   static String s1 = sM1("a");
   String s3 = sM1("2");{
      s1 = sM1("3");
   }
   static{
      s1 = sM1("b");
   }
   static String s2 = sM1("c");
   String s4 = sM1("4");
    public static void main(String args[]){
        InitTest it = new InitTest();
    }
    private static String sM1(String s){
       System.out.println(s);  return s;
    }
}
```

Select 1 option

A. The program will not compile.

B. It will print : `a b c 2 3 4 1`

C. It will print : `2 3 4 1 a b c`

D. It will print : `1 a 2 3 b c 4`

E. It will print : `1 a b c 2 3 4`

Check Answer

71. QID - 2.1161

What will be printed when the following program is compiled and run?

```
class Super{
  public int getNumber( int a){
     return 2;
  }
}
public class SubClass extends Super{
  public int getNumber( int a, char ch){
     return 4;
  }
  public static void main(String[] args){
    System.out.println( new SubClass().getNumber(4) );
  }
}
```

What will be printed?

Select 1 option

A. 4

B. 2

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

72. QID - 2.1206

If `a.equals(b)` returns `true`, `b instanceof ClassOfA` must always be `true`.

(Assume that `ClassOfA` is the name of the class of the variable `a`.)

Select 1 option

A. True

B. False

Check Answer

73. QID - 2.1096

Note: This question may be considered too advanced for this exam. For what command line arguments will the following program print true?

```
class TestClass{

    public static void main(String[] args){
        Integer i = Integer.parseInt(args[0]);
        Integer j = i;
        i--;
        i++;
        System.out.println((i==j));

    }
}
```

Select 3 options

A. 0

B. -1

C. 127

D. -256

E. 256

F. For all the values between 0 and 255 (both included).

Check Answer

74. QID - 2.1241

Consider :

```
class A {}
class B extends A {}
class C extends B {}
```

Which of these boolean expressions correctly identifies when an object 'o' actually refers to an object of class B and not of C?

Select 2 options

A. `(o instanceof B) && (!(o instanceof A))`

B. `!((o instanceof A) || (o instanceof B))`

C. `(o instanceof B) && (!(o instanceof C))`

D. `! ( !(o instanceof B) || (o instanceof C))`

E. `(o instanceof B) && !((o instanceof A) || (o instanceof C))`

Check Answer

75. QID - 2.1207

Which of the following are valid code fragments:

Select 2 options

A. `new Object[]{ "aaa", new Object(), new ArrayList(), 10};`

B. `new Object[]{ "aaa", new Object(), new ArrayList(), {} };`

C. `new Object[]{ "aaa", new Object(), new ArrayList(), new String[] {""} };`

D. `new Object[1]{ new Object() };`

Check Answer

76. QID - 2.874

Which of the following are benefits of an array over an `ArrayList` ?

Select 2 options

A. It consumes less memory.

B. Accessing an element in an array is faster than in `ArrayList`.

C. You do not have to worry about thread safety.

D. It implements `Collection` interface and can thus be passed where ever a `Collection` is required.

Check Answer

77. QID - 2.1116

Which of the following are true about the "default" constructor?

Select 2 options

A. It is provided by the compiler only if the class does not define any constructor.

B. It initializes the instance members of the class.

C. It calls the default 'no-args' constructor of the super class.

D. It initializes instance as well as class fields of the class.

E. It is provided by the compiler if the class does not define a 'no- args' constructor.

Check Answer

78. QID - 2.1247

Which of these assignments are valid?

Select 3 options

A. `short s = 12 ;`

B. `long g = 012 ;`

C. `int i = (int) false;`

D. `float f = -123;`

E. `float d = 0 * 1.5;`

Check Answer

79. QID - 2.879

What will the following code print?

```
int[] scores1 = { 1, 2, 3, 4, 5, 6};
int[] scores2 = { 0, 0, 0, 0, 0, 0};
System.arraycopy(scores2, 2, scores1, 3, 2);
for(int i :  scores2) System.out.print(i);
```

Select 1 option

A. `123006`

B. `000000`

C. `000450`

D. It throw an exception at run time.

Check Answer

80. QID - 2.888

What will the following code print when run?

```
class A {
}

class AA extends A {
}


public class TestClass {
    public static void main(String[] args) throws Exception {
        A a = new A();
        AA aa = new AA();
        a = aa;
        System.out.println("a = "+a.getClass());
        System.out.println("aa = "+aa.getClass());
    }
}
```

Select 1 option

A. It will not compile.

B. It will throw ClassCastException at runtime.

C. a = class AA
aa = class AA

D. a = class A
aa = class AA

Check Answer

81. QID - 2.900

What, if anything, is wrong with the following code?

```
interface T1{
}
interface T2{
   int VALUE = 10;
   void m1();
}

interface T3 extends T1, T2{
   public void m1();
   public void m1(int x);
}
```

Select 1 option

A. T3 cannot implement both T1 and T2 because it leads to ambiguity.

B. There is nothing wrong with the code.

C. The code will work fine only if VALUE is removed from T2 interface.

D. The code will work fine only if m1() is removed from either T2 and T3.

E. None of the above.

Check Answer

82. QID - 2.832

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showOne(100);
        System.out.println(ct.myValue);
        ct.showTwo(200);
        System.out.println(ct.myValue);
    }
}
```

Select 1 option

A. 0 followed by 100.

B. 100 followed by 100.

C. 0 followed by 200.

D. 100 followed by 200.

Check Answer

83. QID - 2.919

After which line will the object created at line XXX be eligible for garbage collection?

```
public Object getObject(Object a) //0
   {

Object b = new Object();  //XXX

Object c, d = new Object(); //1
c = b; //2
b = a = null; //3
return c; //4
}
```

Select 1 option

A. //2

B. //3

C. //4

D. Never in this method.

E. Cannot be determined.

Check Answer

84. QID - 2.909

Which of the following declaration are valid:

```
1.  bool b = null;

2. boolean b = 1;

3. boolean b = true|false;

4 bool b = (10<11);

5. boolean b = true||false;
```

Select 1 option

A. 1 and 4

B. 2, 3, and 5

C. 2 and 3

D. 3 and 5

E. 5

Check Answer

85. QID - 2.896

Which of the following classes have a default constructor?

```
class A{  }
class B {  B(){ } }
class C{  C(String s){ } }
```

Select 1 option

A. A

B. A and B

C. B

D. C

E. B and C

Check Answer

86. QID - 2.881

Java's Exception mechanism helps in which of the following ways?

Select 2 options

A. It allows creation of new exceptions that are custom to a particular application domain.

B. It improves code because error handling code is clearly separated from the main program logic.

C. It enhances the security of the application by reporting errors in the logs.

D. It improves the code because the exception is handled right at the place where it occured.

E. It provides a vast set of standard exceptions that covers all possible exceptions.

Check Answer

87. QID - 2.883

When is the Object created at line //1 eligible for garbage collection?

```
public class TestClass{
  public Object getObject(){
     Object obj = new String("aaaaa");    //1
     Object objArr[] = new Object[1]; //2
     objArr[0] = obj; //3
     obj = null;      //4
     objArr[0] = null;//5
     return obj;      //6
  }
}
```

Select 1 option

A. Just after line 2.

B. Just after line 3.

C. Just after line 4.

D. Just after line 5.

E. Just after line 6.

Check Answer

88. QID - 2.861

You want to find out whether two strings are equal or not, in terms of the actual characters within the strings. What is the best way to do this?

Select 1 option

A. use String's equals method.

B. use String's `equalsIgnoreCase` method.

C. Use == operator.

D. Use String's match method.

Check Answer

89. QID - 2.876

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){        this.side = length;    }

    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

    double getArea() {   return area;   }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Select 2 options

A. Make `setSide()` method private.

B. Make `getArea()` method private.

C. Make `side` and `area` fields private.

D. Make the side field private and remove the area field.

E. Change getArea method to:

```
public double getArea(){ return side*side; }
```

F. Add a `setArea()` method.

Check Answer

90. QID - 2.1365

Which of the following standard java exception classes extend `java.lang.RuntimeException`?

Select 4 options

A. `java.lang.SecurityException`

B. `java.lang.ClassCastException`

C. `java.lang.NullPointerException`

D. `java.lang.CloneNotSupportedException`

E. `java.lang.IndexOutOfBoundsException`

Check Answer

Last Day Test (Unique) (Answered)

Take this test when you are all done with your preparation and are ready for the actual exam. Questions in this test are completely unique in the sense that these are not included in "Practice Tests" or "Objectivewise Tests".

01. QID - 2.840 : Java Basics

What will be the output of the following program when it is compiled and run with the command line:

```
java TestClass 1 2 3

public class TestClass {

    public static void main(String[] args) {
        System.out.println("Values : "+args[0]+args[1]);
    }
}
```

Correct Option is : C

~~**A.**~~ Values : java TestClass

~~**B.**~~ Values : TestClass 1

C. Values : 12

~~**D.**~~ Values : 23

~~**E.**~~ Values : 3

Explanation:

In Java, command line arguments are passed into the program using the `String[]` parameter to the main method. The String array contains actual parameters and does not include java and the name of the class.

Therefore, in this case, args will point to an array of Strings with 3 elements - "1", "2", and "3". The program prints out only `args[0]` and `args[1]`, which is 1 and 2.

Back to Question without Answer

02. QID - 2.1366 : Using Loop Constructs

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        while(int k = 5; k<7){
            System.out.println(k++);
        }
    }
}
```

Correct Option is : D

~~**A.**~~ 5
6

~~**B.**~~ 5
6
7

~~**C.**~~ It will keep printing 5.

D. It will not compile.

In Java, a while or do/while construct takes an expression that returns a boolean. But unlike a for loop, you cannot put instantiation and increment sections in the while condition.

Therefore, `for(int k=5;k<7;)` is valid but `while(int k=5;k<7;)` is not.

~~**E.**~~ It will throw an exception at run time.

Back to Question without Answer

03. QID - 2.878 : Creating and Using Arrays

What will the following code print?

```
int[] scores = { 1, 2, 3, 4, 5, 6};
System.arraycopy(scores, 2, scores, 3, 2);
for(int i :  scores) System.out.print(i);
```

Correct Option is : D

~~**A.**~~ `123446`

~~**B.**~~ `123356`

~~**C.**~~ `1233456`

D. `123346`

~~**E.**~~ `123336`

> Note that if the src and dest arguments refer to the same array object, then the copying is performed as if the components at positions `srcPos` through `srcPos+length-1` were first copied to a temporary array with length components and then the contents of the temporary array were copied into positions `destPos` through `destPos+length-1` of the destination array.

Explanation:

The arraycopy method basically copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array.

The last parameter is the number of elements that you want to copy.

There are questions in the exam on `System.arraycopy` so you should go through the following JavaDoc description for this method:

```
public static void arraycopy(Object src,
             int srcPos,
             Object dest,
             int destPos,
             int length)
```

Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. A subsequence of array components is copied from the source array referenced by src to the destination array referenced by dest. The number of components copied is equal to the length argument. The components at positions srcPos through srcPos+length-1 in the source array are copied into positions destPos through destPos+length-1, respectively, of the destination array. If the src and dest arguments refer to the same array object, then the copying is performed as if the components at positions srcPos through srcPos+length-1 were first copied to a temporary array with length components and then the contents of the temporary array were copied into positions destPos through destPos+length-1 of the destination array.

If dest is null, then a NullPointerException is thrown.

If src is null, then a NullPointerException is thrown and the destination array is not modified.

Otherwise, if any of the following is true, an ArrayStoreException is thrown and the destination is not modified:

The src argument refers to an object that is not an array.
The dest argument refers to an object that is not an array.
The src argument and dest argument refer to arrays whose component types are

different primitive types.
The src argument refers to an array with a primitive component type and the dest argument refers to an array with a reference component type.
The src argument refers to an array with a reference component type and the dest argument refers to an array with a primitive component type.
Otherwise, if any of the following is true, an IndexOutOfBoundsException is thrown and the destination is not modified:

The srcPos argument is negative.
The destPos argument is negative.
The length argument is negative.
srcPos+length is greater than src.length, the length of the source array.
destPos+length is greater than dest.length, the length of the destination array.
Otherwise, if any actual component of the source array from position srcPos through srcPos+length-1 cannot be converted to the component type of the destination array by assignment conversion, an ArrayStoreException is thrown. In this case, let k be the smallest nonnegative integer less than length such that src[srcPos+k] cannot be converted to the component type of the destination array; when the exception is thrown, source array components from positions srcPos through srcPos+k-1 will already have been copied to destination array positions destPos through destPos+k-1 and no other positions of the destination array will have been modified. (Because of the restrictions already itemized, this paragraph effectively applies only to the situation where both arrays have component types that are reference types.)

Parameters:
src - the source array.
srcPos - starting position in the source array.
dest - the destination array.
destPos - starting position in the destination data.
length - the number of array elements to be copied.
Throws:
IndexOutOfBoundsException - if copying would cause access of data outside array bounds.

ArrayStoreException - if an element in the src array could not be stored into the dest array because of a type mismatch.
NullPointerException - if either src or dest is null.

Back to Question without Answer

04. QID - 2.834 : Handling Exceptions

Consider the following

```
public class TestClass {
    public static void main(String[] args) {
        TestClass tc = new TestClass();
        tc.myMethod();
    }

    public void myMethod() {
        yourMethod();
    }

    public void yourMethod() {
        throw new Exception();
    }
}
```

What changes can be done to make the above code compile?

Correct Option is : E

~~A.~~ Change declaration of main to :

```
public static void main(String[] args) throws Exception
```

~~B.~~ Change declaration of myMethod to

```
public void myMethod throws Exception
```

~~C.~~ Change declaration of yourMethod to

```
public void yourMethod throws Exception
```

~~D.~~ Change declaration of main and yourMethod to :

`public static void main(String[] args) throws Exception` and
`public void yourMethod throws Exception`

E. Change declaration of all the three method to include `throws Exception`.

Explanation:

`java.lang.Exception` is a checked Exception. Which means, the method that throws this exception must declare it in the throws clause. Hence, `yourMethod` must declare `throws Exception` in its throws clause.

Now, since the call to `yourMethod` in `myMethod` can also potentially throw an exception, `myMethod` must also declare it in its throws clause. By the same logic, main method should also declare it in its throws clause.

Another alternative is to catch this exception in myMethod:

```
public void myMethod(){

   try{
     yourMethod();
   }
   catch(Exception e){  // since you are catching the exception
thrown by yourMethod, there is no need to declare it in the throws
clause of myMethod.
      e.printStackTrace);
   }
}
```

Further, since a call to `myMethod` cannot throw Exception anymore, main method does not need to declare it either.

Yet another alternative is to catch the exception in the main method:

```
  public static void main(String[] args) {
```

```
        TestClass tc = new TestClass();
        try{
          tc.myMethod();
        }
       catch(Exception e){  // since you are catching the exception
thrown by myMethod, there is no need to declare it in the throws
clause of main.
         e.printStackTrace);
       }
    }

    public void myMethod()  throws Exception{ //Notice the throws
clause here.
        yourMethod();
    }
```

[Back to Question without Answer]

05. QID - 2.889 : Working with Methods

What will the following code print when run?

```
class A{
    String value = "test";
    A(String val){
        this.value = val;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        new A("new test").print();
    }
}
```

Correct Option is : C

~~**A.** test~~

~~**B.** new test~~

C. It will not compile.

There is no method named `print()` defined in class A. Further, there is no such method in class Object either.
To print the contents of an object you can use `toString()` method that returns a String:

```
System.out.println(a.toString());
```

However, for this to print a meaningful value, class A should override the Object class's `toString()` method to return a meaningful String.

~~D.~~ It will throw an exception at run time.

Back to Question without Answer

06. QID - 1.926 : Overloading methods

Consider the following class...

```
class TestClass{
    void probe(Object x) { System.out.println("In Object"); } //3

    void probe(Number x) { System.out.println("In Number"); } //2

    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(Long x) { System.out.println("In Long"); } //4

    public static void main(String[] args){
        double a = 10;
        new TestClass().probe(a);
    }
}
```

What will be printed?

Correct Option is : A

A. In Number

~~**B.**~~ In Object

~~**C.**~~ In Long

~~**D.**~~ In Integer

~~**E.**~~ It will not compile.

Explanation:

Here, we have four overloaded probe methods but there is no probe method that takes a `double` parameter. However, a `double` will be boxed into a `Double` and class `Double` extends `Number`. Therefore, a `Double` can be passed to the method that takes `Number`. A Double can also be passed to a method that takes `Object`, but `Number` is more specific than Object therefore `probe(Number )` will be called.

Back to Question without Answer

07. QID - 2.1357 : Using Operators and Decision Constructs

Which of these combinations of switch expression types and case label value types are legal within a switch statement?

Correct Option is : A

A. switch expression of type int and case label value of type char.

~~**B.**~~ switch expression of type float and case label value of type int.

~~**C.**~~ switch expression of type byte and case label value of type float.

~~**D.**~~ switch expression of type char and case label value of type byte.

This will not work in all cases because a byte may have negative values which cannot be assigned to a char. For example, char ch = -1; does not compile. Therefore, the following does not compile either:

```
    char ch = 'x';
       switch(ch){
          case -1 :         System.out.println("-1"); break; //
This will not compile : "possible loss of precision"
          default:         System.out.println("default");
    }
```

~~**E.**~~ switch expression of type boolean and case label value of type boolean.

Explanation:

Following are primitive types that can be used for a switch() clause : byte, char, short

or int. Java 7 now allows Strings as well. Note that you cannot use boolean, long, float or double. Every case constant expression in a switch block must be assignable to the type of switch expression. Meaning :

```
byte by = 10;
switch(by){
    case: 200 :                 //some code;
    case: 300 :                 //some code;
}
```

This will not compile since 300 is not assignable to 'by' which can only hold values from -128 to 127.

Back to Question without Answer

08. QID - 2.1176 : Using Operators and Decision Constructs

Which code fragments will print the last argument given on the command line to the standard output, and exit without any output and exceptions if no arguments are given?

```
1.
 public static void main(String args[ ]){
       if (args.length != 0)   System.out.println(args[args.length-1
 }

2.
public static void main(String args[ ]){
       try {      System.out.println(args[args.length-1]);        }
       catch (ArrayIndexOutOfBoundsException e) {    }
}

3.
 public static void main(String args[ ]){
     int i = args.length;
     if (i != 0) System.out.println(args[i-1]);
}

4.
public static void main(String args[ ]){
    int i = args.length-1;
   if (i > 0) System.out.println(args[i]);
}

5.
 public static void main(String args[ ]){
       try { System.out.println(args[args.length-1]); }
       catch (NullPointerException e) {}
 }
```

Correct Options are : A B C

A. 1

B. 2

C. 3

~~**D.** 4~~

if there is only one argument, i will be 1-1 = 0. And the if condition will fail.

~~**E.** 5~~

Explanation:

If no argument is given, a String array of length Zero is received in the main method. So, there is no NullPointerException on accessing args even if no argument is given. Indexing in java starts from zero. So the last element will be at args.length-1.

Back to Question without Answer

09. QID - 2.1224 : Using Loop Constructs

Consider the following code for the main() method:

```
public static void main(String[] args) throws Exception{
   int i = 1, j = 10;
   do {
      if (i++ > --j) continue;
   } while (i < 5);
   System.out.println("i=" + i + " j=" + j);
}
```

What will be the output when the above code is executed?

Correct Option is : B

~~**A.** i=6 j=6~~

B. i=5 j=6

~~**C.** i=5 j=5~~

~~**D.** i=6 j=5~~

~~**E.** None of these.~~

Explanation:

To understand the flow, let us put a print statement in the code:

```
  int i = 1, j = 10;
```

```
int k =1;
do {
   System.out.println("Iteration "+k+": i=" + i + " j=" + j);
   k++;
   if (i++ > --j) continue;
} while (i < 5);
System.out.println("i=" + i + " j=" + j);
```

It generates the following output:

```
Iteration 1: i=1 j=10
Iteration 2: i=2 j=9
Iteration 3: i=3 j=8
Iteration 4: i=4 j=7
i=5 j=6
```

In the iteration 1, the if comparison goes like this:
`if (1++ > --10 ) continue; => if( 1 > 9 )` . The values of i and j after the if statement are 2 and 9
In the iteration 2, the if comparison goes like this:
`if (2++ > --9 ) continue; => if( 2 > 8 )` . The values of i and j after the if statement are 3 and 8
In the iteration 3, the if comparison goes like this:
`if (3++ > --8 ) continue; => if( 3 > 7 )` . The values of i and j after the if statement are 4 and 7
In the iteration 4, the if comparison goes like this:
`if (4++ > --7 ) continue; => if( 4 > 6 )` . The values of i and j after the if statement are 5 and 6

Now, i is not < 5 so the `while(i<5)` check fails and the loop terminates. So the final values are 5 and 6.

Back to Question without Answer

10. QID - 2.1253 : Working with Inheritance

Where, in a constructor, can you place a call to a super class's constructor ?

Correct Option is : B

~~**A.**~~ Anywhere in the constructor's body.

B. As the first statement in the constructor.

~~**C.**~~ Only as the first statement and it can be called just like any other method call i.e. ClassName(...).

> No. You have to do super(...) instead of ClassName(...)

~~**D.**~~ You can't call super class's constructor in a base class as constructors are not inherited..

> That constructors are not inherited is true but you can call them using super(...). You can call the super class's constructor only from a constructor and only as the first statement.

~~**E.**~~ None of the above.

Back to Question without Answer

11. QID - 2.1339 : Using Operators and Decision Constructs

What will the following code print?

```
int i = 1;
int j = i++;
if( (i==++j) | (i++ == j) ){
  i+=j;
}
System.out.println(i);
```

Correct Option is : C

~~**A.**~~ 3

~~**B.**~~ 4

C. 5

~~**D.**~~ 2

~~**E.**~~ It will not compile.

Explanation:

This question is based on 2 concepts:

1. `i == ++j` is not same as `i == j++;`
In the case of `i == ++j`, j is first incremented and then compared with i. While in the case of `i == j++;`, j is first compared with i and then incremented.

2. The | operator, when applied for boolean operands, ensures that both the sides are evaluated. This is opposed to || which does not evaluate the Right Hand Side if the result can be known by just evaluating the Left Hand Side.

Now, let us see the values of i and j at each step:

```
int i = 1;
int j = i++; // j is assigned 1 and i is incremented to 2
if( (i==++j) | (i++ == j) )     // increment j (so j becomes 2) and
compare with i => return true.
        //since it is |, evaluate next condition: compare i with 2
and increment i => i becomes 3.{
    i+=j; //i = 3+2 = 5
}
System.out.println(i); //prints 5
```

Back to Question without Answer

12. QID - 2.1314 : Working with Java Data Types - Variables and Objects

Which statements concerning conversion are true?

Correct Options are : A B C E

A. Conversion from char to long does not need a cast.

B. Conversion from byte to short does not need a cast.

C. Conversion from short to char needs a cast.

> The reverse is also true. Because their ranges are not compatible.

~~**D.**~~ Conversion from int to float need a cast.

> No. Because a float can hold any value of int. Note that opposite is not true because of loss of precision.

E. Conversion from byte, char or short to int, long or float does not need a cast.

> Because int, long or float are bigger that byte char or short.

Back to Question without Answer

13. QID - 2.1334 : Working with Inheritance

What will the following program print when run?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[] ){ A b = new B("good bye");
}
class A{
   A() { this("hello", " world");  }
   A(String s) { System.out.println(s); }
   A(String s1, String s2){ this(s1 + s2); }
}
class B extends A{
   B(){ super("good bye"); };
   B(String s){ super(s, " world "); }
   B(String s1, String s2){ this(s1 + s2 + " ! "); }
}
```

Correct Option is : C

~~**A.**~~ It will print "good bye".

~~**B.**~~ It will print "hello world".

C. It will print "good bye world".

~~**D.**~~ It will print "good bye" followed by "hello world".

~~**E.**~~ It will print "hello world" followed by "good bye".

Explanation:

`new B("good bye");` will call class B's one args constructor which in turn calls `super(s, " world ");` (i.e. class A's two args constructor) which in turn calls `this(s1 + s2);` (i.e. class A's one arg constructor with parameter "good bye world") which prints it.

Back to Question without Answer

14. QID - 2.1268 : Using Loop Constructs

Which of these for statements are valid?

```
1. for (int i=5; i=0; i--) { }

2.  int j=5;
      for(int i=0, j+=5;  i<j ; i++) {  j--;  }

3. int i, j;
    for (j=10; i<j; j--) { i += 2; }

4. int i=10;
    for ( ; i>0 ; i--) { }

5. for (int i=0, j=10; i<j; i++, --j) {;}
```

Correct Option is : D

~~**A.**~~ 1, 2

~~**B.**~~ 3, 4

~~**C.**~~ 1, 5

1 is not valid.

D. 4, 5

~~**E.**~~ 5

Explanation:

No 1.
uses '=' instead of '==' for condition which is invalid. The loop condition must be of type boolean.

No 2.
uses 'j +=5'. Now, this statement is preceded by 'int i=0,' and that means we are trying to declare variable j. But it is already declared before the for loop. If we remove the int in the initialization part and declare i before the loop then it will work. But if we remove the statement int j = 5; it will not work because compiler will try to do j = j+5 and you can't use the variable before it is initialized. Although the compiler gives a message 'Invalid declaration' for j += 5 but it really means the above mentioned thing.

No 3. i is uninitialized.

No 4. is valid. It contains empty initialization part.

No 5.
This is perfectly valid. You can have any number of comma separated statements in initialization and incrementation part. The condition part must contain a single expression that returns a boolean.
All a for loop needs is two semi colons :-
for(; ;) {} This is a valid for loop that never ends. A more concise form for the same is : for(; ;);

Back to Question without Answer

15. QID - 2.862 : Working with Methods

Consider the following code appearing in the same file:

```
class Data {
    int x = 0, y = 0;
    public Data(int x, int y){
        this.x = x; this.y = y;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d = new Data(1, 1);
        //add code here
    }
}
```

Which of the following options when applied individually will change the Data object currently referred to by the variable d to contain 2, 2 as values for its data fields?

Correct Options are : A C

A. Add the following two statements:

```
d.x = 2;
d.y = 2;
```

~~**B.**~~ Add the following statement:

```
d = new Data(2, 2);
```

This will create a new Data object and will not change the original Data object referred to be d.

C. Add the following two statements:

```
d.x += 1;
d.y += 1;
```

~~D.~~ Add the following statement:

```
d = d + 1;
```

This will not compile because Java does not allow operator overloading for user defined objects.

Back to Question without Answer

16. QID - 2.864 : Handling Exceptions

Java Exceptions is a mechanism ..

Correct Options are : B C

~~**A.**~~ for dealing with unexpected user inputs.

B. that you can use to determine what to do when something unexpected happens.

> Exceptions provide the means to separate the details of what to do when something out of the ordinary happens from the main logic of a program. Once you get an exception, you can catch it and in the catch block you can determine what actions should be taken based on the situation. Thus, the actions that you have to take under exceptional circumstances are isolated from the main flow of the program and improves clarity of the code.

C. for logging unexpected behavior.

> once you catch an exception, you can log the details.

~~**D.**~~ to ensure that the program runs even if something unexpected happens.

> While it is possible to keep the program "running", in case of an exception, that is not what exceptions mechanism is meant for. Exceptions provide the means to separate the details of what to do when something out of the ordinary happens from the main logic of a program.

~~**E.**~~ that the VM uses to exit the program when something unexpected happens.

Explanation:

The actual exam contains several questions on exceptions that contain vague statements. It is not possible to determine what exactly is meant by a particular option and so our answers are based on our interpretation of the options. To answer such questions, we recommend you to go through the following trail that explains the exceptions from Oracle's perspective:
http://docs.oracle.com/javase/tutorial/essential/exceptions/

Back to Question without Answer

17. QID - 2.839 : Working with Inheritance

How can you fix the following code to make it compile:

```
import java.io.*;
class Great {
    public void doStuff() throws FileNotFoundException{
    }
}

class Amazing extends Great {
  public void doStuff() throws IOException, IllegalArgumentException
  }
}

public class TestClass {
    public static void main(String[] args) throws IOException{
        Great g = new Amazing();
        g.doStuff();
    }
}
```

Assume that changes suggested in a option are to be applied independent of other options.

Correct Options are : A D

A. Change `doStuff` in Amazing to throw only `IllegalArgumentException`.

> `IllegalArgumentException` extends from `RuntimeException`. So you don't have to worry about it at least at compile time. You may or may not declare it in the throws clause. The caller doesn't have to catch it anyway.
> The overriding method in the subclass is free to not throw any checked exception at all even if the overridden method throws a checked exception. No exception is a valid subset of exceptions thrown by the overridden method.

~~**B.**~~ Change `doStuff` in `Great` to throw only `FileNotFoundException` as well as `IllegalArgumentException`.

~~**C.**~~ Change `doStuff` in `Amazing` to throw only `IOException`.

D. Change `doStuff` in `Great` to throw only `IOException` instead of `FileNotFoundException`.

~~**E.**~~ Replace `g.doStuff()` to `((Amazing) g).doStuff()`.

Explanation:

The rule is that an overriding method cannot throw an exception that is a super class of the exception thrown by the overridden method.

Now, `FileNotFoundException` is a subclass of `IOException`. Therefore, Amazing's `doStuff()` cannot throw `IOException` if the base class's `doStuff` throws only `FileNotFoundException`.

Think of it this way:

```
FileNotFoundException fne = new IOException(); // Will this work?
No, because an IOException is NOT a FileNotFoundException.
IOException ioe = new FileNotFoundException(); // Will this work?
Yes, because a FileNotFoundException is an IOException.
```

Therefore, overriding method must not throw an exception that cannot be assigned to a variable whose class is the class of the overridden method's exception.

Back to Question without Answer

18. QID - 2.925 : Creating and Using Arrays

Given:

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<Double> al = new ArrayList<>();

        //INSERT CODE HERE
     }
}
```

What can be inserted in the above code so that it can compile without any error?

Correct Options are : B C E

~~**A.**~~ `al.add(111);`

> You cannot box an int into a Double object.

B. `System.out.println(al.indexOf(1.0));`

> 1.0 is a double and so it can be boxed into a Double object.

C. `System.out.println(al.contains("string"));`

~~**D.**~~ `Double d = al.get(al.length);`

> ArrayList does not have a field named length. It does have a method named size() though. So you can do:
> `Double d = al.get(al.size());` It will compile but will throw IndexOutOfBoundsException at run time.

E. `al.notifyAll();`

Since notifyAll() method is a part of Object class, it can be called on any object. Though it will throw an IllegalMonitorStateException at runtime because this method must be called if you have acquired a lock on that object.

Please note that although not specifically mentioned in the objectives, some candidates have reported getting an option referring to this method. That is why we have included it.

Back to Question without Answer

19. QID - 2.908 : Using Operators and Decision Constructs

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {

          boolean flag  = true;
          switch (flag){
              case true : System.out.println("true");
                  default: System.out.println("false");
          }

    }
}
```

Correct Option is : A

A. It will not compile.

> A boolean cannot be used for a switch statement. It needs an integral type, an enum, or a String.

~~**B.**~~ false

~~**C.**~~ true
false

~~**D.**~~ Exception at run time.

Back to Question without Answer

20. QID - 2.871 : Creating and Using Arrays

What will the following code print when compiled and run?

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<Integer> al = new ArrayList<>(); //1
        al.add(111); //2
        System.out.println(al.get(al.size()));  //3
     }
}
```

Correct Option is : D

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an exception at run time because of line //1

~~**C.**~~ It will throw an exception at run time because of line //2

> Although 111 is a primitive, it will automatically be boxed into an Integer object. So there will be no exception because of this.

D. It will throw an exception at run time because of line //3

> It will throw an `IndexOutOfBoundsException` at run time because of this line. The `size()` method of `ArrayList` returns the number of elements. Here, it returns 1. Since numbering in `ArrayList` starts with 0. al.get(1) will cause an `IndexOutOfBoundsException` to be thrown because only 0 is a valid index for a list of size 1.

~~E.~~ null.

Back to Question without Answer

21. QID - 2.853 : Using Operators and Decision Constructs

Which line in the following code will cause the compilation to fail?

```
public class TestClass {

    public static void main(String[] args) throws Exception {
       work();                  //LINE 10
       int j = j1;               //LINE 11
       int j1 = (double) x; //LINE 12
    }

    public static void work() throws Exception{
        System.out.println(x); //LINE 15
    }

    static double x;    //19
}
```

Correct Option is : B

~~**A.** Line 10~~

B. Line 11

j1 has not been declared at this point. You cannot use a variable before it is declared.
Note that static and instance fields are always defined before everything else. So even though x is declared after the work() method, it will be initialized before the method is actually executed.

~~**C.** Line 12~~

~~**D.** Line 15~~

E. Line 19

Back to Question without Answer

22. QID - 2.846 : Handling Exceptions

Consider the following code:

```
public class TestClass {

    public static void doStuff() throws Exception{
        System.out.println("Doing stuff...");
        if(Math.random()>0.4){
            throw new Exception("Too high!");
        }
        System.out.println("Done stuff.");
    }

    public static void main(String[] args) throws Exception {
        doStuff();
        System.out.println("Over");
    }
}
```

Which of the following are possible outputs when the above program is compiled and run? (Assume that Math.random() returns a double between 0.0 and 1.0 not including 1.0. Further assume that there is no mistake in the line numbers printed in the output.)

Correct Options are : A C

A.

```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
 at TestClass.doStuff(TestClass.java:29)
 at TestClass.main(TestClass.java:41)
```

B.

```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
 at TestClass.doStuff(TestClass.java:29)
 at TestClass.main(TestClass.java:41)
Over
```

> If `doStuff()` throws an exception, the code after the call to `doStuff()` in main will not be executed and therefore, "Over" will not be printed.

C.

```
Doing stuff...
Done stuff.
Over
```

~~D.~~

```
Doing stuff...
Exception in thread "main" java.lang.Exception: Too high!
 at TestClass.doStuff(TestClass.java:29)
 at TestClass.main(TestClass.java:41)
Done stuff.
```

> Once an exception is thrown in a method, the code after that exception will not be executed. Therefore, if `doStuff()` throws an exception, "Done stuff." will not be printed.

Explanation:

There are only two possibilities:
1. If `Math.random()` generates a number more than 0.4, the if part will throw an exception. In this case, the remain code of `doStuff` will not be called and `main()` will receive an exception due to the call to `doStuff`. Since `doStuff()` is not within a try/catch block, the exception will propagate up and the remaining code in `main()` will not be executed either.

Since the exception is not caught anywhere in the code, it will finally reach the JVM's thread that has called the main method. This thread catches the exception and prints out the stack trace.

2. If `Math.random()` generates a number not more than 0.4, if part will not be executed and "`Done stuff.`" will be printed. After the call returns in `main()`, "`Over`" will be printed as well.

Back to Question without Answer

23. QID - 2.1138 : Working with Inheritance

What will the following program print when run?

```
class Super{
  public String toString(){
     return "4";
  }
}
public class SubClass extends Super{
  public String toString(){
     return super.toString()+"3";
  }
  public static void main(String[] args){
    System.out.println( new SubClass() );
  }
}
```

Correct Option is : A

A. 43

~~**B.**~~ 7

~~**C.**~~ It will not compile.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

This is quite simple, `toString()` is called on the Object of class SubClass.
Subclass's `toString()` calls super class's `toString()` which returns String 4 (not an integer 4!). It then appends "3" to it.
So the final value is "43".

Back to Question without Answer

24. QID - 2.933 : Java Basics

Which of the given options can be successfully inserted at line 1....

```
//line 1
   public class A{
   }
```

Correct Options are : A B D

A. `import java.lang.*;`

> Although this package is automatically imported, it is not an error to import it explicitly.

B. `package p.util;`

> It is a perfectly valid package statement.

C. `public class MyClass{ }`

> There can be only 1 "public" class within package scope in a file. You can have additional inner classes that are public though.

D. `abstract class MyClass{ }`

> You can have more than one classes in a file but at most one of them can be public.

Explanation:

To make a class abstract, you only need to mark it abstract as shown in Option 4. You don't necessarily need to put an abstract method in a class.

Back to Question without Answer

25. QID - 2.1011 : Overloading methods

Consider the following code:

```
public class TestClass{
   public void method(Object o){
      System.out.println("Object Version");
   }
   public void method(java.io.FileNotFoundException s){
      System.out.println("java.io.FileNotFoundException Version");
   }
   public void method(java.io.IOException s){
      System.out.println("IOException Version");
   }
   public static void main(String args[]){
      TestClass tc = new TestClass();
      tc.method(null);
   }
}
```

What would be the output when the above program is compiled and run? (Assume that FileNotFoundException is a subclass of IOException, which in turn is a subclass of Exception)

Correct Option is : C

~~**A.**~~ It will print `Object Version`

~~**B.**~~ It will print `java.io.IOException Version`

C. It will print `java.io.FileNotFoundException Version`

~~**D.**~~ It will not compile.

~~E.~~ It will throw an exception at runtime.

Explanation:

The reason is quite simple, the most specific method depending upon the argument is called. Here, null can be passed to all the 3 methods but `FileNotFoundException` class is the subclass of `IOException` which in turn is the subclass of `Object`. So, `FileNotFoundException` class is the most specific class. So, this method is called. Had there been two most specific methods, it would not even compile as the compiler will not be able to determine which method to call. For example:

```
public class TestClass{
   public void method(Object o){
      System.out.println("Object Version");
   }
   public void method(String s){
      System.out.println("String Version");
   }
   public void method(StringBuffer s){
      System.out.println("StringBuffer Version");
   }
   public static void main(String args[]){
      TestClass tc = new TestClass();
      tc.method(null);
   }
}
```

Here, `null` can be passed as both `StringBuffer` and `String` and none is more specific than the other. So, it will not compile.

Back to Question without Answer

26. QID - 2.869 : Using Loop Constructs

Which of the following are true about the enhanced for loop?

Correct Options are : A B E

A. It can iterate over an array or a Collection but not a Map.

> The enhanced for loop needs either an array or any object that implements Collection interface. Map does not implement Collection interface, though you can use Map.getKeySet() or Map.values() to get a Collection and then iterate over that Collection.

B. Using an enhanced for loop prevents the code from going into an infinite loop.

> Since there is no explicity condition check written in the code, it provides less opportunity to write code that causes infinite loop.

~~**C.**~~ Using an enhanced for loop on an array may cause infinite loop.

~~**D.**~~ An enhanced for loop can iterate over a Map.

E. You cannot find out the number of the current iteration while iterating.

> Unlike in a regular for loop, there is no iteration variable available in an enhanced for loop, so it is not possible to determine the number of the iteration just by using the enhanced for loop. You will need to do something like:
>
> ```
> int i = 0;
> for(Object obj : collectionOrArray){
>   System.out.println("Iteration number = "+i+" Object = "+obj);
> ```

```
    i++;
}
```

Back to Question without Answer

27. QID - 2.1024 : Using Loop Constructs

Given the following code fragment, which of the following lines would be a part of the output?

```
outer:
   for ( int i = 0 ; i<3 ; i++ ){
      for ( int j = 0 ; j<2 ; j++ ){
         if ( i == j ){
            continue outer;
         }
         System.out.println( "i=" + i + " , j=" + j );
      }
   }
```

Correct Options are : A D

A. i = 1, j = 0

~~B.~~ i = 0, j = 1

~~C.~~ i = 1, j = 2

D. i = 2, j = 1

~~E.~~ i = 2, j = 2

Explanation:

The given code prints:

```
i=1, j=0
```

```
i=2, j=0
i=2, j=1
```

The variable i iterates through the values 0, 1 and 2 in the outer loop, while j varies from 0 to 1 in the inner loop.

If the values of i and j are equal, the continue statement is executed and printing is skipped and next iteration of outer 'for' loop starts.

Back to Question without Answer

28. QID - 2.899 : Working with Inheritance

Given:

```
interface I { }

class A implements I{
    public String toString(){ return "in a"; }
}

class B extends A{
    public String toString(){ return "in b"; }
}


public class TestClass {

    public static void main(String[] args) {
        B b = new B();
        A a = b;
        I i = a;

        System.out.println(i);
        System.out.println((B)a);
        System.out.println(b);

    }
}
```

What will be printed when the above code is compiled and run?

Correct Option is : E

~~A.~~ in i
in a
in b

~~**B.** I~~
A
in b

~~**C.** in a~~
in a
in b

~~**D.** in a~~
in b
in b

E. in b
in b
in b

There is only one object created in this code and the class of that object is B. Therefore, B's toString will be called no matter what reference you use. Therefore, it is print "in b" for all the cases.

Explanation:

If you answered this question incorrectly, you need to understand the concept of polymorphism. We suggest you to go through any book to understand it thoroughly because there are several questions in the exam on similar pattern.

In a nutshell, polymorphism means that it is always the class of the object (and not the class of the reference variable that a variable points to) that determines which method will be called at run time. The concept of polymorphism doesn't apply to private methods or static methods because these methods are never inherited.

Back to Question without Answer

29. QID - 2.991 : Using Operators and Decision Constructs

What can be the return type of method getSwitch so that this program compiles and runs without any problems?

```
public class TestClass{
   public static XXX getSwitch(int x){
      return x - 20/x + x*x;
   }
   public static void main(String args[]){
       switch( getSwitch(10) ){
          case 1 :
          case 2 :
          case 3 :
          default : break;
       }
   }
}
```

Correct Option is : A

A. int

~~**B.**~~ float

~~**C.**~~ long

~~**D.**~~ double

~~**E.**~~ char

~~**F.** byte~~

~~**G.** short~~

Explanation:

If you just consider the method `getSwitch`, any of int long float or double will do. But the return value is used in the switch statement later on. A switch condition cannot accept float, long, double, or boolean. So only `int` is valid.

The return type cannot be byte, short, or char because the expression `x - 20/x + x*x;` returns an int.

Back to Question without Answer

30. QID - 2.1290 : Using Operators and Decision Constructs

Consider the following method:

```
static int mx(int s){
    for(int i=0; i<3; i++){
        s = s + i;
    }
    return s;
}
```

and the following code snippet:

```
int s = 5;
    s += s + mx(s) + ++s;
    System.out.println(s);
```

What will it print?

Correct Option is : D

~~**A.**~~ 21

~~**B.**~~ 22

~~**C.**~~ 23

D. 24

s += (expression) will be converted to s = s + expression. So the given expression will become:

```
s = s + s + mx(s) + ++s;
```

```
s = 5 + 5 + mx(5) + 6;
s = 5 + 5+ 8 + 6;
s = 24;
```

~~**E.**~~ 25

~~**F.**~~ 26

Back to Question without Answer

31. QID - 2.1198 : Working with Inheritance

What will be the output of the following program ?

```
class CorbaComponent{
    String ior;
    CorbaComponent(){ startUp("IOR"); }
    void startUp(String s){ ior  =  s; }
    void print(){ System.out.println(ior); }
}

class OrderManager extends CorbaComponent{
   OrderManager(){  }
   void startUp(String s){  ior = getIORFromURL(s);   }
   String getIORFromURL(String s){  return "URL://"+s; }
}

public class Application{
   public static void main(String args[]){ start(new OrderManager()),
   static void start(CorbaComponent cc){ cc.print(); }
}
```

Correct Option is : C

~~**A.**~~ It will throw an exception at run time.

~~**B.**~~ It will print `IOR`

C. It will print `URL://IOR`

~~**D.**~~ It will not compile.

~~**E.**~~ None of the above.

Explanation:

Note that method `startUp(String s)` is overridden by the base class.

When an object of clas `OrderManager` is constructed, the default no args constructor of `CorbaComponent` is called. This constructor calls the `startUp(String s)` with "IOR" as parameter. Now there are two eligible methods which can be called, the `CorbaComponent` one and `OrderManager` one.

The method selection is done on the basis of the ACTUAL OBJECT Class (which is `OrderManager` here). So `OrderManager`'s `startUp(...)` is called which sets the `ior` variable to `URL://IOR`.

Unlike method selection, variable selection is done on the basis of class of the variable and not on the actual class of object that it is referring to.

Back to Question without Answer

32. QID - 2.951 : Working with Inheritance

Consider the following program...

```
class Super {  }
class Sub extends Super {  }
public class TestClass{
   public static void main(String[] args){
      Super s1 = new Super(); //1
      Sub s2 = new Sub();     //2
      s1 = (Super) s2;        //3
   }
}
```

Which of the following statements are correct?

Correct Option is : A

A. It will compile and run without any problems.

~~**B.**~~ It will compile but WILL throw ClassCastException at runtime.

~~**C.**~~ It will compile but MAY throw ClassCastException at runtime.

~~**D.**~~ It will not compile.

~~**E.**~~ None of the above.

Explanation:

Note that `s2` is a variable of class `Sub`, which is a subclass of `Super`. `s1` is a variable

of class `Super`. A subclass can ALWAYS be assigned to a super class variable without any cast. It will always compile and run without any exception.

For example, a `Dog` "IS A" `Animal`, so you don't need to cast it.
But an `Animal` may not always be a `Dog`. So you need to cast it to make it compile and during the runtime the actual object referenced by `animal` should be a `Dog` otherwise it will throw a `ClassCastException`.

Back to Question without Answer

33. QID - 2.857 : Using Operators and Decision Constructs

Consider the following code:

```
public class TestClass {

    //define tester method here

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        while(tc.tester()){
            System.out.println("running...");
        }
    }
}
```

Which of the following options would be a valid implementation of tester() method?

Correct Options are : A B

A.
```
public boolean tester(){
    return false;
}
```

B.
```
public Boolean tester(){
    return false;
}
```

C.
```
public tester(){
    return false;
}
```

return type is missing.

~~D.~~

```
public int tester(){
    return 0;
}
```

It is a valid method but it will not work for while(tester()) because a while condition expects a boolean or Boolean value.

~~E.~~

```
public String tester(){
    return "false";
}
```

A string cannot be used in while condition. So it has the same problem as above.

Back to Question without Answer

34. QID - 2.929 : Using Operators and Decision Constructs

Consider the following lines of code:

```
boolean greenLight = true;
boolean pedestrian = false;
boolean rightTurn = true;
boolean otherLane = false;
```

You can go ahead only if the following expression evaluates to 'true' :

```
(( (rightTurn && !pedestrian || otherLane) || ( ? && !pedestrian &&
greenLight ) )  == true )
```

What variables can you put in place of '?' so that you can go ahead?

Correct Option is : C

~~**A.**~~ rightTurn

~~**B.**~~ otherLane

C. Any variable would do.

since the part before second || is true, the next part is not even evaluated.

~~**D.**~~ None of the variable would allow to go.

Explanation:

Observe that `(rightTurn && !pedestrian || otherLane)` is `true`, therefore `( ? && !pedestrian && greenLight )` does not matter.

|| and && are short circuit operators. So, if the first part of the expression (i.e. part before ||) is true (or false for &&) the other part is not evaluated at all.

Note that this is not true for | and &. In that case, the whole expression will be evaluated even if the value of the expression can be known by just evaluating first part.

Back to Question without Answer

35. QID - 2.1194 : Using Operators and Decision Constructs

Consider the following lines of code:

```
System.out.println(null + true); //1
System.out.println(true + null); //2
System.out.println(null + null); //3
```

Which of the following statements are correct?

Correct Option is : A

A. None of the 3 lines will compile.

~~**B.**~~ All the 3 line will compile and print `nulltrue`, `truenull` and `nullnull` respectively.

~~**C.**~~ Line 1 and 2 won't compile but line 3 will print `nullnull`.

~~**D.**~~ Line 3 won't compile but line 1 and 2 will print `nulltrue` and `truenull` respectively.

~~**E.**~~ None of the above.

Explanation:

Note that none of the parameters is a String so conversion to String will not happen. The following are the error messages given by the compiler.

C:\works\nbtestproject\src\TestClass.java:46: operator + cannot be applied to

<nulltype>,boolean
   System.out.println(null + true); //1
C:\works\nbtestproject\src\TestClass.java:46: reference to println is ambiguous, both method println(char[]) in java.io.PrintStream and method println(java.lang.String) in java.io.PrintStream match
   System.out.println(null + true); //1
C:\works\nbtestproject\src\TestClass.java:47: operator + cannot be applied to boolean,<nulltype>
   System.out.println(true + null); //2
C:\works\nbtestproject\src\TestClass.java:47: reference to println is ambiguous, both method println(char[]) in java.io.PrintStream and method println(java.lang.String) in java.io.PrintStream match
   System.out.println(true + null); //2
C:\works\nbtestproject\src\TestClass.java:48: operator + cannot be applied to <nulltype>,<nulltype>
   System.out.println(null + null); //3
C:\works\nbtestproject\src\TestClass.java:48: reference to println is ambiguous, both method println(char[]) in java.io.PrintStream and method println(java.lang.String) in java.io.PrintStream match
   System.out.println(null + null); //3
6 errors

If one operand expression is of type String, then string conversion is performed on the other operand to produce a String at run time. The result is a reference to a newly created String object that is the concatenation of the two Strings. The characters of the left-hand operand precede the characters of the right-hand operand in the newly created string.

Any type may be converted to type String by string conversion.

A value x of primitive type T is first converted to a reference value as if by giving it as an argument to an appropriate class instance creation expression e.g. if T is boolean, then use new Boolean(x) .

toString() is defined by the primordial class Object; many classes override it, notably Boolean, Character, Integer, Long, Float, Double, and String.

Note that had there been a method like String getString() { return null; }, println(getString() + true) etc. would have compiled fine and would have printed "nulltrue".

Back to Question without Answer

36. QID - 2.1078 : Using Operators and Decision Constructs

The following code snippet will print true.

```
String str1 = "one";
String str2 = "two";
System.out.println( str1.equals(str1=str2) );
```

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

First the value of 'str1' is evaluated (i.e. one). Now, before the method is called, the operands are evaluated, so str1 becomes "two". so "one".equals("two") is false.

Back to Question without Answer

37. QID - 2.918 : Using Operators and Decision Constructs

Given:

```
int a = 1 + 2 + 3 * 4;
int b = 2 * 3 + 4;

int total = a + b;
```

What will be the value of total?

Correct Option is : D

~~**A.**~~ 34

~~**B.**~~ 38

~~**C.**~~ 29

D. 25

Multiplication has more precedence than addition. So this will be evaluated as:

```
int a = 1 + 2 + (3 * 4);
3+12
15

int b = 2 * 3 + 4;
6+4
10
```

So, `total = 25`

Explanation:

You may get a few very simple questions about operator preference. Simple school math trick of BODMAS can be used to evaluate the expressions.

B Brackets first
O Orders (i.e. Powers and Square Roots, etc.)
DM Division and Multiplication (left-to-right)
AS Addition and Subtraction (left-to-right)

Back to Question without Answer

38. QID - 2.953 : Using Operators and Decision Constructs

Which of the following are also called as "short circuiting logical operators"?

Correct Options are : B C

~~**A.** &~~

B. ||

C. &&

~~**D.** |~~

~~**E.** ^~~

Explanation:

|| and && are called short circuiting operators because if, while evaluating a logical expression, at any stage, the value of the whole expression can be determined without evaluating the rest of the expression, then the remaining sub-expressions are not evaluated.

Consider this:

```
boolean bool  = true; int k = 10;
if( bool == false && ( (k = 3) == 5 )  ) { .....}
System.out.println(k);              // this will print 10.
```

Because the value of the whole expression can be determined just by looking at `bool == false`.

So `k = 3` is never executed. The big expression was short circuited by &&.

Had the expression been `if( bool == false & ( (k = 3) == 5 ) ) { .....} /* notice single & instead of && */`
then it would have printed 3 because `k = 3` will be executed. Even though the value of the expression is known immediately after evaluating `bool == false`, the rest of the expression is still evaluation. Thus, & is not a short circuiting operator.

Same thing happens with || and | as well.

Back to Question without Answer

39. QID - 2.1371 : Working with Java Data Types - Variables and Objects

Given the following declarations:

```
int a = 5, b = 7, k = 0;
Integer m = null;
```

and the following statements:

```
k = new Integer(a) + new Integer(b);  //1
k = new Integer(a) + b; //2
k = a + new Integer(b); //3
m = new Integer(a) + new Integer(b); //4
```

Executed independent of each other, what will be the value of k (for //1, //2, and //3) and m (for //4) after execution of each of these statements?

Correct Option is : C

~~**A.**~~ 12
will not compile
will not compile
12

~~**B.**~~ will not compile
will not compile
will not compile
12

C. 12
12
12
12

~~**D.**~~ will not compile
will not compile
will not compile
will not compile

~~**E.**~~ 12
12
12
will not compile

Explanation:

In all of these statements, auto-unboxing of integers will occur. For the last statement, after unboxing a and b, the value 12 will be boxed into an Integer object.

Back to Question without Answer

40. QID - 2.1175 : Working with Java Data Types - String, StringBuilder

Which of the following statements will evaluate to true?

Correct Option is : D

~~**A.**~~ `"String".replace('g','G') == "String".replace('g','G')`

> replace creates a new string object.

~~**B.**~~ `"String".replace('g','g') == new String("String").replace('g','g')`

~~**C.**~~ `"String".replace('g','G')=="StrinG"`

> replace creates a new string object.

D. `"String".replace('g','g')=="String"`

> replace returns the same object if there is no change.

~~**E.**~~ None of these.

Explanation:

There are 2 points to remember:
1. `replace()` method creates a new `String` object.
2. `replace()` method returns the same `String` object if both the parameters are same, i.e. if there is no change.

Back to Question without Answer

41. QID - 2.1169 : Constructors

Which of the following can be used as a constructor for the class given below?

```
public class TestClass{
   // lots of code ...
}
```

Correct Options are : B E

~~**A.**~~ `public void TestClass() {...}`

> There should be no return type. Not even void.

B. `public TestClass() {...}`

~~**C.**~~ `public static TestClass() {...}`

> Constructors cannot be static.

~~**D.**~~ `public final TestClass() {...}`

> Constructors cannot be final.

E. `public TestClass(int x) { ...}`

Explanation:

You can use only one of public protected and private.
Unlike methods, a constructor cannot be abstract, static, final, native, or synchronized. A constructor is not inherited, so there is no need to declare it final and an abstract constructor could never be implemented. A constructor is always invoked with respect

to an object, so it makes no sense for a constructor to be static. There is no practical need for a constructor to be synchronized, because it would lock the object under construction, which is normally not made available to other threads until all constructors for the object have completed their work. The lack of native constructors is an arbitrary language design choice that makes it easy for an implementation of the Java Virtual Machine to verify that superclass constructors are always properly invoked during object creation.

Back to Question without Answer

42. QID - 2.875 : Working with Methods

What will the following code print when compiled and run:

```
class Data {

    int intVal = 0;
    String strVal = "default";
    public Data(int k){
        this.intVal = k;
    }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d1 = new Data(10);
        d1.strVal = "D1";
        Data d2 = d1;
        d2.intVal = 20;
        System.out.println("d2 val = "+d2.strVal);
    }
}
```

Correct Option is : C

~~**A.**~~ d2 val =

~~**B.**~~ d2 val = default

C. d2 val = D1

~~**D.**~~ Exception at run time.

Explanation:

This is quite straight forward question. You are creating only one Data object. You are setting its strVal field to "D1". Next, you declare another Data variable d2 and assign to it the same Data object.

Thus, when you access strVal using d2, you will get D1.

The "throws Exception" part is not required and is there just to confuse you.

Back to Question without Answer

43. QID - 2.1090 : Using Loop Constructs

What is the effect of compiling and running the code shown in exhibit?

```
public class TestClass{
   public static void main (String args []){
      int sum = 0;
      for (int i = 0, j = 10; sum > 20; ++i, --j)       // 1
      {
         sum = sum+ i + j;
      }
      System.out.println("Sum = " + sum);
   }
}
```

Correct Option is : B

~~**A.**~~ Compile time error at line 1.

B. It will print Sum = 0

Note that the loop condition is sum >20 and not sum <20.

~~**C.**~~ It will print Sum = 20

Note that the loop condition is sum >20 and not sum <20.

~~**D.**~~ Runtime error.

~~**E.**~~ None of the above.

Explanation:

Read the questions carefully. This is very important. Some questions are easy but you need to read them carefully.

Back to Question without Answer

44. QID - 2.855 : Overloading methods

What will be printed when the following code is compiled and run?

```
public class LoadTest{

    public static void main(String[] args) throws Exception {
         LoadTest t = new LoadTest();
         int i = t.getLoad();
         double d = t.getLoad();
         System.out.println( i + d );
    }

    public int getLoad() {
        return 1;
    }

    public double getLoad(){
        return 3.0;
    }

}
```

Correct Option is : D

~~**A.** 13.0~~

~~**B.** 4.0~~

~~**C.** 4~~

D. The code will not compile.

You cannot have more than one method in a class with the same signature.
Method signature includes method name and the argument list but does not include return type.
Therefore, the two `getLoad()` methods have the same signature and will not compile.

This shows that method overloading cannot be done on the basis of the return types.

Back to Question without Answer

45. QID - 2.844 : Encapsulation

Consider the following code:

```
import java.util.ArrayList;

public class Student{

    ArrayList<Integer> scores;
    private double average;

    public ArrayList<Integer> getScores(){ return scores; }

    public double getAverage(){ return average; }

    private void computeAverage(){
        //valid code to compute average
        average =//update average value
    }

    public Student(){
        computeAverage();
    }
}
```

What can be done to improve the encapsulation of this class?

Correct Options are : B E

~~**A.**~~ Make the class private.

B. Make the `scores` instance field private.

An important aspect of encapsulation is that other classes should not be able to modify the state fields of a class directly. Therefore, the data members should be private and if the class want to allow access to these field, it should provide

appropriate setters and getters.

~~**C.**~~ Make `getScores()` protected.

~~**D.**~~ Make `computeAverage()` public.

E. Change getScores to return a copy of the scores list:

```
public ArrayList<Integer> getScores(){
   return new ArrayList(scores);
}
```

If you return the same scores list, the caller would be able to add or remove elements from it, thereby rendering the average incorrect.
This can be prevented by returning a copy of the list.

Back to Question without Answer

46. QID - 2.856 : Working with Methods - Access Modifiers

Consider the following code:

```
public class MyClass {

  protected int value = 10;

}
```

Which of the following statements are correct regarding the field value?

Correct Option is : D

~~**A.**~~ It cannot be accessed from any other class.

~~**B.**~~ It can be read but cannot be modified from any other class.

~~**C.**~~ It can be modified but only from a subclass of MyClass.

It can also be modified from any class defined in the same package.

D. It can be read and modified from any class within the same package or from any subclass of MyClass.

Back to Question without Answer

47. QID - 2.886 : Working with Methods

Given:

```
class StaticTest{

    void m1(){
        StaticTest.m2();  // 1
        m4();             // 2
        StaticTest.m3();  // 3
    }

    static void m2(){ }  // 4

    void m3(){
        m1();             // 5
        m2();             // 6
        StaticTest.m1(); // 7
    }

    static void m4(){ }
}
```

Which of the lines will fail to compile?

Correct Options are : C G

~~**A.**~~ 1

~~**B.**~~ 2

C. 3

~~**D.**~~ 4

~~**E.** 5~~

~~**F.** 6~~

G. 7

Explanation:

To call an instance method, you need to use the reference that points to an object of that class. When you call an instance method from another instance method, you don't need a reference because "this" is implicit.

You can call a static method from either a static or an instance method. No object reference is required. You can call it by using the name of the class or you can omit that as well.

At //3, you are trying to call an instance method from another instance method. Therefore, you need to either specify an object reference or you can rely on this if you omit it. However, you cannot do StaticTest.m3() because StaticTest is not a valid reference that points to an object of class StaticTest.

Same thing happens at //7.

Back to Question without Answer

48. QID - 2.841 : Handling Exceptions

You have a method that currently does not handle any exception thrown from the code contained in its method body. You are now changing this method to call another method that throws `IOException`.

What changes, independent of each other, can you make to your method so that it will compile?

Correct Options are : B D

~~**A.** Set the exception to `null` and don't rethrow it.~~

> This option doesn't make sense. To get the exception, you first need to catch it.

B. Declare `IOException` in the throws clause of your method.

~~**C.** Wrap the call to another method within a try-catch block that catches `RuntimeException`.~~

> `java.io.IOException` extends `Exception`. It cannot be caught by a catch block that catches `RuntimeException`.

D. Wrap the call to another method within a try-catch block that catches `Exception`.

> Since `IOException` is an `Exception`, you can catch it with a catch block that catches `Exception`.

Back to Question without Answer

49. QID - 2.911 : Creating and Using Arrays

Given the complete contents of TestClass.java file:

```
package x;
public class TestClass {
    ArrayList<String> al;
    public void init(){
        al = new ArrayList<>();
        al.add("Name 1");
        al.add("Name 2");
    }
    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.init();
        System.out.println("Size = "+tc.al.size());
    }
}
```

Which import statement should be added to make it compile?

Correct Option is : C

~~**A.**~~ `import java.lang.*;`

~~**B.**~~ `import java.lang.ArrayList;`

C. `import java.util.ArrayList;`

~~**D.**~~ `import java.collections.ArrayList;`

~~**E.**~~ No import is necessary.

Explanation:

Only `java.lang` package and the package in which the current class exists are automatically imported.
Class `ArrayList` is in `java.util` package, which is not imported automatically.

Note that classes in the default package (i.e. the package with no name) cannot be imported by classes in other (i.e. non default) packages. This is why you should not use the default package for creating classes.

Back to Question without Answer

50. QID - 2.882 : Using Operators and Decision Constructs

What will the following program print when run without any command line argument?

```
public class TestClass {
    public static void main(String[] args)  {

        boolean hasParams = (args == null ? false : true);
        if(hasParams){
            System.out.println("has params");
        }{
            System.out.println("no params");
        }
    }
}
```

Correct Option is : B

~~**A.**~~ has params

B. has params
no params

~~**C.**~~ no params

~~**D.**~~ It will not compile.

Explanation:

Remember that the args array is never `null`. If the program is run without any arguments, args points to a `String` array of length 0. Therefore, `hasParams` will be

`true` and it will print "`has params`".

Since there is no else, the subsequent code block will also be executed and it will print "`no params`". Note that it is not syntactically wrong to have section of code wrapped in `{ }`.

Back to Question without Answer

51. QID - 2.828 : Java Basics

Consider the following code appearing in a file named TestClass.java:

```
class Test{ }  // 1

public class TestClass {

   public int main(String[] args)  { // 2

        double x=10, double y;  // 3

        System.out.println[]; // 4

         for(int k =0; k<x; k++){ }

        return 0;
   }
}
```

Which of the lines are invalid?

Correct Option is : B

~~**A.**~~ // 1 and // 4

B. // 3 and // 4

~~**C.**~~ // 2 and // 4

~~**D.**~~ // 2 and // 3

Explanation:

// 1 is valid because it is a valid code that declares a class.

// 2 is a valid declaration of a method named main. Although, it is not a correct declaration for the standard main method that can be used to execute the class, but it is a valid method nevertheless.

// 3 is invalid syntax. It can be written as either `double x=10; double y;` or `double x=10, y;`
Note that even though x is a double and 10 is an int, it is valid because 10 will automatically be converted to a double. The reverse would not be valid i.e. `int x = 10.0;` will be invalid.
You need a cast for that: `int x = (int) 10.0;`

//4 is invalid because println is not a class name. So you cannot create an array of it. `println` is a method. So it should be written as `System.out.println();`

//5 is a valid declaration of a for loop.

Back to Question without Answer

52. QID - 2.1185 : Working with Java Data Types - String, StringBuilder

What will the following code print?

```
String s = "blooper";
StringBuilder sb = new StringBuilder(s);
sb.append(s.substring(4)).delete(3, 5);
System.out.println(sb);
```

Correct Option is : C

~~**A.**~~ `blorbloo`

~~**B.**~~ `bloper`

C. `bloerper`

```
s.substring(4) => "blooper".substring(4) => per
sb.append(s.substring(4)).delete(3, 5); =>
"blooperper".delete(3, 5) => bloerper
```

~~**D.**~~ `blooperper`

~~**E.**~~ `bloo`

Explanation:

Please read the following description of substring method of String and delete method of `StringBuilder` and `StringBuffer`:

`public String substring(int beginIndex)`

Returns a new string that is a substring of this string. The substring begins with the

character at the specified index and extends to the end of this string.
Examples:

```
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)
```

```
public StringBuilder delete(int start, int end)
```

Removes the characters in a substring of this sequence. The substring begins at the specified start and extends to the character at index end - 1 or to the end of the sequence if no such character exists. If start is equal to end, no changes are made.

Back to Question without Answer

53. QID - 2.1144 : Working with Inheritance

Consider the following interface definition:

```
public interface ConstTest{
 public int A = 1; //1
 int B = 1;          //2
 static int C = 1;  //3
 final int D = 1;   //4
 public static int E = 1; //5
 public final int F = 1;  //6
 static final int G = 1;    //7
 public static final int H = 1; //8
}
```

Which line(s) will cause a compilation error?

Correct Option is : I

~~**A.**~~ 1

~~**B.**~~ 2

~~**C.**~~ 3

~~**D.**~~ 4

~~**E.**~~ 5

~~**F.**~~ 6

~~**G.**~~ 7

~~**H.**~~ 8

I. None of them will cause any error.

Any field in an interface is implicitly public, static, and final, whether these keywords are specified or not.

Back to Question without Answer

54. QID - 2.985 : Using Operators and Decision Constructs

Which of the following declarations are valid?

Correct Options are : C D E

~~A.~~ `float f1 = 1.0;`

1.0 is a double.

~~B.~~ `float f = 43e1;`

43e1 is a double.

C. `float f = -1;`

D. `float f = 0x0123;`

E. `float f = 4;`

Explanation:

Although 1.0 and 43e1 can fit into a float, the implicit narrowing does not happen because implicit narrowing is permitted only among byte, char, short, and int.

Back to Question without Answer

55. QID - 2.1372 : Handling Exceptions

Identify the exceptions that will be received when the following code snippets are executed.

1.
```
int factorial(int n){
       if(n==1) return 1;
      else return n*factorial(n-1);
   }
```
Assume that it is called with a very large integer.

2.
```
void printMe(Object[] oa){
       for(int i=0; i<=oa.length; i++)
       System.out.println(oa[i]);
    }
```
Assume that it is called as such: `printMe(null);`

3.
```
Object m1(){
       return new Object();
    }
    void m2(){
       String s = (String) m1();
    }
```

Correct Option is : H

~~**A.**~~ ClassCastException
ArrayIndexOutOfBoundsException
StackOverflowError

~~**B.**~~ ClassCastException
ArrayIndexOutOfBoundsException
SecurityException

~~**C.**~~ No Exception Will Be Thrown
SecurityException
Will Not Compile

~~**D.**~~ StackOverflowError
NullPointerException
No Exception Will Be Thrown

~~**E.**~~ StackOverflowError
ArrayIndexOutOfBoundsException
ClassCastException

~~**F.**~~ StackOverflowError
NullPointerException
NullPointerException

~~**G.**~~ SecurityException
NullPointerException
No Exception Will Be Thrown

H. StackOverflowError
NullPointerException
ClassCastException

Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

56. QID - 2.1160 : Working with Java Data Types - Variables and Objects

Given that `OurClass` is a `MyClass` and `OurClass` has a `YourClass` object.
Which of the following options are correct?

(Assume and `OurClass, MyClass,` and `YourClass` are valid java classes.)

Correct Options are : D E

~~**A.**~~ MyClass contains a reference to OurClass

~~**B.**~~ OurClass contains a reference to MyClass

~~**C.**~~ MyClass contains a reference to YourClass

D. OurClass contains a reference to YourClass

E. OurClass inherits from MyClass

Explanation:

Visualize the hierarchy like this:
`OurClass` is a `MyClass` => `OurClass` extends (or inherits from) `MyClass`. Thus, option 5 is correct.
`OurClass` has a `YourClass` => `OurClass` refers to (or contains a reference to) `YourClass` object. Thus, option 4 is correct.

Back to Question without Answer

57. QID - 2.1167 : Handling Exceptions

Given that SomeException is a checked exception, consider the following code:

```
//in file A.java
public class A{
   protected void m() throws SomeException{}
}

//in file B.java
public class B extends A{
   public void m(){ }
}

//in file TestClass.java
public class TestClass{
   public static void main(String[] args){
      // insert code here. // 1
   }
}
```

Which of the following options can be inserted at //1?

Correct Option is : C

~~**A.**~~ `B b =  new A();`
`b.m();`

> `B b = new A();` is not valid because a superclass object can never be assigned to a base class reference.

~~**B.**~~ `A a = new B();`
`a.m();`

> A's m() declares 'throws SomeException', which is a checked exception, while the main() method doesn't. So a.m() must be wrapped in a try/catch block.

C.
```
A a = new B();
( ( B) a ).m();
```

> Due the explicit casting of 'a' to B, the compiler knows that 'a' will point to an object of class B (or its subclass), whose method m() does not throw an exception. So there is no need for a try catch block here.

D.
```
Object o = new B();
o.m();
```

> Object class does not have method m(). So o.m() will not compile. You can do
> `( (B) o ).m();`

E. None of these.

Back to Question without Answer

58. QID - 2.1200 : Handling Exceptions

What can be done to get the following code to compile and run? (Assume that the options are independent of each other.)

```
public float parseFloat( String s ){
     float f = 0.0f;       // 1
     try{
          f = Float.valueOf( s ).floatValue();     // 2
          return f ;       // 3
     }
     catch(NumberFormatException nfe){
        f = Float.NaN ;     // 4
       return f;      // 5
     }
     finally {
         return f;      // 6
     }
     return f ;     // 7
 }
```

Correct Options are : A C D E

A. Remove line 3, 6

B. Remove line 5

C. Remove line 5, 6

D. Remove line 7

E. Remove line 3, 7

Explanation:

Basically, an unreachable statement causes a compilation error (There is one exception: `if(false) { ... }` is valid.). As such, line 7 is unreachable because of the return statement in finally. This is because finally is always executed and there it returns a value, so there is no way line 7 can be executed!

When you remove the lines suggested by the options, all the lines of code are executed in one case or another. For example, in option 1, if you comment line 3 and 6, Line 7 will be executed if no exception is thrown in the try block.

We suggest you to try working out other scenarios yourself in a similar manner.

Back to Question without Answer

59. QID - 2.1352 : Using Loop Constructs

Using a `continue` in a while loop causes the loop to break the current iteration and start the next iteration of the loop.

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

A `continue` causes the next iteration of the loop to start without executing the remaining statements in the loop.

Back to Question without Answer

60. QID - 2.1075 : Constructors

Given the following source code, which of the lines that are commented out may be reinserted without introducing errors?

```
abstract class Bang{
 //abstract void f();  //(0)
   final    void g(){}
 //final    void h(){} //(1)
   protected static int i;
   private int j;
}

final class BigBang extends Bang{
 //BigBang(int n) { m = n; } //(2)
   public static void main(String args[]){
      Bang mc = new BigBang();
   }
   void h(){}
 //void k(){ i++; } //(3)
 //void l(){ j++; } //(4)
   int m;
}
```

Correct Option is : C

A. `final void h( ) { } //(1)`

It will fail because BigBang will try to override a final method.

B. `BigBang(int n) { m = n; }  //(2)`

It will fail since BigBang will no longer have a default constructor that is used in the main( ) method.

C. `void k( ) { i++ }   //(3)`

~~D.~~ `void l( ) { j++ }  //(4)`

> It will fail since the method will try to access a private member 'j' of the superclass.

~~E.~~ `abstract void f( ) ;  //(0)`

> If this line is inserted, then class BigBang will have to be declared abstract.

Explanation:

Default constructor (having no arguments) is automatically created only if the class does not define any constructors. So as soon as //2 is inserted the default constructor will not be created.

Back to Question without Answer

61. QID - 2.1128 : Using Operators and Decision Constructs

What will the following code print?

```
int i = 0;
int j = 1;
if( (i++ == 0) & (j++ == 2) ){
   i = 12;
}
System.out.println(i+" "+j);
```

Correct Option is : A

A. 1 2

~~B.~~ 2 3

~~C.~~ 12 2

~~D.~~ 12 1

~~E.~~ It will not compile.

Explanation:

This question is based on 2 concepts:

1. `i = ++j;` is not same as `i = j++;`
In the case of i = ++j, j is first incremented and then compared with i. While in the case of i = j++;, j is first compared with i and then incremented.

2. The `|` and `&` operators, when applied to boolean operands, ensure that both the sides are evaluated. This is opposed to `||` and `&&` operators, which do not evaluate the Right Hand Side operand if the result can be known by just evaluating the Left Hand Side.

Now, let us see the values of i and j at each step:

```
int i = 0;
int j = 1;
if( (i++ == 0) & (j++ == 2) )    //compare i with 0 and increment i
=> returns true and i becomes 1. Evaluate next condition:
        //compare j with 2 and increment j => return false and j
becomes 2.
        //true & false returns false so i= 12 is not executed.{
   i = 12;
}
System.out.println(i+" "+j)); //print 1 and 2
```

Back to Question without Answer

62. QID - 2.1230 : Java Basics

What will the following code print when run?

```
public class TestClass{
  public static long main(String[] args){
     System.out.println("Hello");
     return 10L;
  }
}
```

Correct Option is : D

~~**A.**~~ Hello

~~**B.**~~ It will print nothing.

~~**C.**~~ It will not compile

D. It will throw a Throwable at runtime.

~~**E.**~~ None of the above.

Explanation:

When the program is run, the JVM looks for a method named `main()` which takes an array of `Strings` as input and returns nothing (i.e. the return type is `void`).
But in this case, it doesn't find such a method (the given `main()` method is returning `long`!) so it throws a `java.lang.NoSuchMethodError`.
Note that `java.lang.Error` does not extend `Exception` class. It extends `java.lang.Throwable` and so it can be "thrown".

Back to Question without Answer

63. QID - 2.1000 : Handling Exceptions

What will the following code print when run?

```
public class Test {

    static String s = "";

    public static void m0(int a, int b) {
        s += a;
        m2();
        m1(b);
    }

    public static void m1(int i) {
        s += i;
    }

    public static void m2() {
        throw new NullPointerException("aa");
    }

    public static void m() {
        m0(1, 2);
        m1(3);
    }

    public static void main(String args[]) {
        try {
            m();
        } catch (Exception e) {
        }
        System.out.println(s);
    }
}
```

Correct Option is : A

A. 1

~~**B.**~~ 12

~~**C.**~~ 123

~~**D.**~~ 2

~~**E.**~~ It will throw exception at runtime.

Explanation:

Try to follow the control flow:

1. `m()` calls `m0(1, 2)`.
2. `m0(1, 2)` first executes `s += 1` (so s is now 1) and then calls `m2()`.
3. Now, `m2()` throws an exception which is not caught by `m2()` so it is propagated back to `m0(1, 2)`. Since, within m0 method, the call to `m2()` is not wrapped in a try catch block, this exception further propagates up to `m()`. (The next line in `m(1, 2)`, which is `m1(2)`, is not executed).
4. Again, `m()` also does not have the try catch block so the same exception is further propagated up to the `main()` method. (The next line in `m()`, which is a call to `m1(3)` is not called).
4. In main method, the call to `m()` is wrapped in a try catch block and so the exception is handled here.
5. Finally, s stays as "1".

The point to note here is that if you do not catch an exception, it is propagated up the stack of method calls until it is handled. If nobody handles it, the JVM handles that exception and kills the thread. If that thread is the only user thread running, the

program ends.

Back to Question without Answer

64. QID - 2.1051 : Using Operators and Decision Constructs

Assuming that a valid integer will be passed in the command line as first argument, which statements regarding the following code are correct?

```
public class TestClass{
   public static void main(String args[]){
      int x = Integer.parseInt(args[0]);
      switch(x){
         case x < 5 :   System.out.println("BIG"); break;
         case x > 5 :   System.out.println("SMALL");
         default :    System.out.println("CORRECT"); break;
      }
   }
}
```

Correct Option is : D

~~**A.**~~ `BIG` will never be followed by `SMALL`.

~~**B.**~~ `SMALL` will never follow anything else.

~~**C.**~~ `SMALL` will always be followed by `CORRECT`.

D. It will not compile.

~~**E.**~~ It will throw an exception at runtime.

Explanation:

It will say the following when compiled:

```
TestClass.java: incompatible types
found   : boolean
required: int
case x < 5 :        System.out.println("BIG"); break;


TestClass.java: incompatible types
found   : boolean
required: int
case x > 5 :        System.out.println("SMALL");
```

This is because the type of the case labels must be consistent with the type of the switch condition. Here, switch condition is an int, so the case label values must be assignable to the switch condition variable. The expression x<5 is of type boolean, which cannot be assigned it x (since it is an int).

Back to Question without Answer

65. QID - 2.1295 : Working with Inheritance

Given the following class definitions and declaration:

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}

D d = new D();
```

the expression `(d instanceof A)` will return `true`.

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

D extends C, which extends B, which extends A. This means, D is-a C, C is-a B, and B is-a A. Therefore, D is-a A. Hence, d instanceof A will return true.

Back to Question without Answer

66. QID - 2.1291 : Working with Inheritance

What will be the result of attempting to compile and run class B?

```
class A{
   final int fi = 10;
}
public class B extends A{
   int fi = 15;
   public static void main(String[] args){
       B b = new B();
       b.fi = 20;
       System.out.println(b.fi);
       System.out.println(  (  (A) b  ).fi  );
   }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

~~**B.**~~ It will print 10 and then 10

~~**C.**~~ It will print 20 and then 20

~~**D.**~~ It will print 10 and then 20

E. It will print 20 and then 10

Explanation:

Note that a final variable can be shadowed. Here, although fi in A is final, it is shadowed by fi of B. So `b.fi = 20;` is valid since B's fi is not final.

Back to Question without Answer

67. QID - 2.1182 : Using Operators and Decision Constructs

What will be the output of the following class:

```
public class TestClass{
  public void testRefs(String str, StringBuilder sb){
    str = str + sb.toString();
    sb.append(str);
    str = null;
    sb = null;
  }
  public static void main(String[] args){
    String s = "aaa";
    StringBuilder sb = new StringBuilder("bbb");
    new TestClass().testRefs(s, sb);
    System.out.println("s="+s+" sb="+sb);
  }
}
```

Correct Option is : E

~~**A.**~~ `s=aaa sb=bbb`

~~**B.**~~ `s=null sb=null`

~~**C.**~~ `s=aaa sb=null`

~~**D.**~~ `s=null sb=bbbaaa`

E. `s=aaa sb=bbbaaabbb`

Explanation:

Always remember that Strings are immutable, you cannot change them. In this case, `s` refers to `"aaa"`, and no matter what `testRefs()` method does, the variable `s` of `main()` will keep pointing to the same string `"aaa"`.

`StringBuilder` on the other hand, is mutable. So, initially `sb` is pointing to a `StringBuilder` object containing `"bbb"`. Its reference is passed to the `testRefs()` method. In that method, we change the local variable str to point to a new string `"aaa"+"bbb" = "aaabbb"`. Then we append this to `sb`. Therefore sb now contains `"bbbaaabbb"`.
Setting the local reference `str` and `sb` (in method `testRefs()`) to `null`, does not affect the variables `s` and `sb` of the `main()` method.

Back to Question without Answer

68. QID - 2.1166 : Java Basics

Which line of code will not be acceptable to the compiler?

```
public class XBox{
  volatile int root = 20; //1
  private XBox() //2
  {
    volatile int i = 30; //3
  }
  private void XBox() //4
  {
    int local = 30;
  }
}
```

Correct Option is : C

~~**A.** 1~~

~~**B.** 2~~

You can have a private constructor.

C. 3

Only member variable can be volatile or transient.

~~**D.** 4~~

Because of the presence of a return type, this becomes a valid method and does not interfere with the constructor of same signature.

~~**E.**~~ The code will compile fine.

Back to Question without Answer

69. QID - 2.1257 : Using Operators and Decision Constructs

Which of the following statements concerning the switch construct are true?

Correct Options are : A B C

A. A character literal can be used as a value for a case label.

> boolean, long, float and double cannot be used.

B. A 'long' cannot be used as a switch variable.

> boolean, long, float and double cannot be used.

C. An empty switch block is a valid construct.

~~**D.**~~ A switch block must have a default label.

~~**E.**~~ If present, the default label must be the last of all the labels.

> Any order is valid.

Explanation:

Here are the rules for a switch statement:
`boolean, long, float` and `double` cannot be used for the case labels. Any integral type(i.e. `int, char, byte, short`), `String`, or `enum` can be used.

All of the following must be true, or a compile-time error will result:
1.Every case constant expression associated with a switch statement must be assignable to the type of the switch

Expression. i.e. if the switch expression is of type `byte` then all the case constants must fit in a `byte` (e.g. you can't use `200` as a case value)

2. No two of the case constant expressions associated with a switch statement may have the same value.

3. At most, one default label may be associated with the same switch statement. (It is valid to not have default label at all.)

Back to Question without Answer

70. QID - 2.1158 : Working with Methods

What will the following program print?

```
public class InitTest{
   public InitTest(){
      s1 = sM1("1");
   }
   static String s1 = sM1("a");
   String s3 = sM1("2");{
      s1 = sM1("3");
   }
   static{
      s1 = sM1("b");
   }
   static String s2 = sM1("c");
   String s4 = sM1("4");
    public static void main(String args[]){
        InitTest it = new InitTest();
    }
    private static String sM1(String s){
       System.out.println(s);  return s;
    }
}
```

Correct Option is : B

~~A.~~ The program will not compile.

B. It will print : `a b c 2 3 4 1`

~~C.~~ It will print : `2 3 4 1 a b c`

~~**D.**~~ It will print : `1 a 2 3 b c 4`

~~**E.**~~ It will print : `1 a b c 2 3 4`

Explanation:

First, static statements/blocks are called IN THE ORDER they are defined.
Next, instance initializer statements/blocks are called IN THE ORDER they are defined.
Finally, the constructor is called. So, it prints `a b c 2 3 4 1`.

Back to Question without Answer

71. QID - 2.1161 : Working with Inheritance

What will be printed when the following program is compiled and run?

```
class Super{
  public int getNumber( int a){
     return 2;
  }
}
public class SubClass extends Super{
  public int getNumber( int a, char ch){
     return 4;
  }
  public static void main(String[] args){
    System.out.println( new SubClass().getNumber(4) );
  }
}
```

What will be printed?

Correct Option is : B

~~**A.** 4~~

B. 2

~~**C.** It will not compile.~~

~~**D.** It will throw an exception at runtime.~~

~~**E.** None of the above.~~

Explanation:

Note that the parameters of SubClass's getNumber are different than Super's getNumber. So it is not overriding it. So the super class's `getNumber()` will be called which returns 2.

Back to Question without Answer

72. QID - 2.1206 : Using Operators and Decision Constructs

If `a.equals(b)` returns `true`, `b instanceof ClassOfA` must always be `true`.

(Assume that `ClassOfA` is the name of the class of the variable `a`.)

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

This may not always be correct because `equals()` method can be overridden. By default, it tests reference assignment, but any subclass of Object is free to redefine `equals()` as it deems fit. So, it is possible that an equals method may return `true` even if the class of the passed object has no relation to this object.

Back to Question without Answer

73. QID - 2.1096 : Using Operators and Decision Constructs

Note: This question may be considered too advanced for this exam. For what command line arguments will the following program print true?

```
class TestClass{

    public static void main(String[] args){
        Integer i = Integer.parseInt(args[0]);
        Integer j = i;
        i--;
        i++;
        System.out.println((i==j));

    }
}
```

Correct Options are : A B C

A. 0

B. -1

C. 127

~~**D.** -~~256

~~**E.** -~~256

~~**F.** -~~For all the values between 0 and 255 (both included).

Explanation:

All the wrapper objects are immutable. When you do `i++`, what actually happens is something like this:
`i = new Integer( i.intValue()  + 1);` As you can see, a new Integer object is assigned back to i.

However, to save on memory, Java 'reuses' all the wrapper objects whose values fall in the following ranges:

All Boolean values (true and false)
All Byte values
All Character values from \u0000 to \u007f (i.e. 0 to 127 in decimal)
All Short and Integer values from -128 to 127

So == will always return true when their primitive values are the same and belong to the above list of values.

Note that the following will not compile though:

```
Byte b = 1; Integer i = 1;
b == i; //Invalid because both operands are of different class.
```

Back to Question without Answer

74. QID - 2.1241 : Using Operators and Decision Constructs

Consider :

```
class A {}
class B extends A {}
class C extends B {}
```

Which of these boolean expressions correctly identifies when an object 'o' actually refers to an object of class B and not of C?

Correct Options are : C D

~~**A.**~~ `(o instanceof B) && (!(o instanceof A))`

> This will return false if o refers to an Object of class A, B, or C because (o instanceof A) will be true for all the three.

~~**B.**~~ `!((o instanceof A) || (o instanceof B))`

C. `(o instanceof B) && (!(o instanceof C))`

D. `! ( !(o instanceof B) || (o instanceof C))`

> This is the complement of "(o instanceof B) && (!(o instanceof C))" prefixed with a '!'. So in effect, both are same.

~~**E.**~~ `(o instanceof B) && !((o instanceof A) || (o instanceof C))`

Explanation:

The expression `(o instanceof B)` will return `true` if the object referred to by `o` is

of type B or a subtype of B. The expression `(! (o instanceof C))` will return `true` unless the object referred to by `o` is of type C or a subtype of C. Thus, the expression `(o instanceof B) && (!(o instanceof C))` will only return `true` if the object is of type B or a subtype of B that is not C or a subtype of C.
Given objects of classes A, B and C, this expression will only return `true` for objects of class B.

Back to Question without Answer

75. QID - 2.1207 : Creating and Using Arrays

Which of the following are valid code fragments:

Correct Options are : A C

A. `new Object[]{ "aaa", new Object(), new ArrayList(), 10};`

> 10 is a primitive and not an Object but due to auto-boxing it will be converted into an Integer object and that object will then be stored into the array of Objects.

~~B.~~ `new Object[]{ "aaa", new Object(), new ArrayList(), {} };`

> {} is not defining any kind of Object.

C. `new Object[]{ "aaa", new Object(), new ArrayList(), new String[] {""} };`

> Every array is an Object so new String[]{""} is also an Object and can be placed in an array of objects.

~~D.~~ `new Object[1]{ new Object() };`

> You can't specify array length if you are initializing it at the same place.

Explanation:

1. An array of objects can store Objects of any class.
2. Primitives (i.e. int, byte, char, short, boolean, long, double, and float) are NOT objects.
3. An array (of primitives as well as of objects) is an Object.

Back to Question without Answer

76. QID - 2.874 : Creating and Using Arrays

Which of the following are benefits of an array over an `ArrayList` ?

Correct Options are : A B

A. It consumes less memory.

> Because of additional internal data structure and pointers, an `ArrayList` consumes a little more memory than an array.

B. Accessing an element in an array is faster than in `ArrayList`.

> Although very little, but a direct array access using an `index` is faster than calling a method on `ArrayList`.

~~**C.**~~ You do not have to worry about thread safety.

> Neither an `ArrayList` nor an array is thread safe. If you have multiple threads trying to add and remove elements from an `ArrayList` or an array, you have to write additional code to ensure thread safety.

~~**D.**~~ It implements `Collection` interface and can thus be passed where ever a `Collection` is required.

> arrays do not implement `Collection` interface. `ArrayList` does. This is actually an advantage of an `ArrayList` over an array.

Explanation:

An `ArrayList` resized dynamically at run time as per the situation. An array cannot be resized once created. This reduces the amount of boiler plate code that is required to do the same task using an array.

Back to Question without Answer

77. QID - 2.1116 : Constructors

Which of the following are true about the "default" constructor?

Correct Options are : A C

A. It is provided by the compiler only if the class does not define any constructor.

~~**B.**~~ It initializes the instance members of the class.

C. It calls the default 'no-args' constructor of the super class.

~~**D.**~~ It initializes instance as well as class fields of the class.

~~**E.**~~ It is provided by the compiler if the class does not define a 'no- args' constructor.

It is not provided even if the class declares any other constructor.

Explanation:

The default constructor is provided by the compiler only when a class does not define ANY constructor explicitly. For example,

```
public class A{
  public A()  //This constructor is automatically inserted by the cor
    super();  //Note that it calls the super class' default no-args c
  }
}
public class A{
  //Compiler will not generate any constructor because the programme
  public A(int i){
     //do something
```

```
    }
}
```

Back to Question without Answer

78. QID - 2.1247 : Working with Java Data Types - Variables and Objects

Which of these assignments are valid?

Correct Options are : A B D

A. `short s = 12 ;`

> This is valid since 12 can fit into a short and an implicit narrowing conversion can occur.

B. `long g = 012 ;`

> 012 is a valid octal number.

~~**C.**~~ `int i = (int) false;`

> Values of type boolean cannot be converted to any other types.

D. `float f = -123;`

> Implicit widening conversion will occur in this case.

~~**E.**~~ `float d = 0 * 1.5;`

> double cannot be implicitly narrowed to a float even though the value is representable by a float.

Explanation:

Note that
float d = 0 * 1.5f; and float d = 0 * (float)1.5 ; are OK

A narrowing primitive conversion may be used if all of the following conditions are satisfied:

1. The expression is a constant expression of type int.

2. The type of the variable is byte, short, or char.

3. The value of the expression (which is known at compile time, because it is a constant expression) is representable in the type of the variable.

Note that narrowing conversion does not apply to long or double. So, `char ch = 30L;` will fail even though `30` is representable in `char`.

Back to Question without Answer

79. QID - 2.879 : Creating and Using Arrays

What will the following code print?

```
int[] scores1 = { 1, 2, 3, 4, 5, 6};
int[] scores2 = { 0, 0, 0, 0, 0, 0};
System.arraycopy(scores2, 2, scores1, 3, 2);
for(int i :  scores2) System.out.print(i);
```

Correct Option is : B

~~**A.**~~ `123006`

B. `000006`

> Source is `scores2` and destination is `scores1`. So `scores1` will become `1 2 3 0 0 6`. However, you are printing `scores2`, which is still `{0, 0, 0, 0, 0, 0}`.

~~**C.**~~ `000450`

~~**D.**~~ It throw an exception at run time.

Explanation:

The arraycopy method basically copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. The last parameter is the number of elements that you want to copy.

There are questions in the exam on `System.arraycopy` so you should go through the following JavaDoc description for this method:

```
public static void arraycopy(Object src,
             int srcPos,
             Object dest,
             int destPos,
             int length)
```

Copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. A subsequence of array components is copied from the source array referenced by src to the destination array referenced by dest. The number of components copied is equal to the length argument. The components at positions srcPos through srcPos+length-1 in the source array are copied into positions destPos through destPos+length-1, respectively, of the destination array. If the src and dest arguments refer to the same array object, then the copying is performed as if the components at positions srcPos through srcPos+length-1 were first copied to a temporary array with length components and then the contents of the temporary array were copied into positions destPos through destPos+length-1 of the destination array.

If dest is null, then a `NullPointerException` is thrown.

If src is null, then a `NullPointerException` is thrown and the destination array is not modified.

Otherwise, if any of the following is true, an ArrayStoreException is thrown and the destination is not modified:

The src argument refers to an object that is not an array.
The dest argument refers to an object that is not an array.
The src argument and dest argument refer to arrays whose component types are different primitive types.
The src argument refers to an array with a primitive component type and the dest argument refers to an array with a reference component type.
The src argument refers to an array with a reference component type and the dest

argument refers to an array with a primitive component type.
Otherwise, if any of the following is true, an IndexOutOfBoundsException is thrown and the destination is not modified:

The srcPos argument is negative.
The destPos argument is negative.
The length argument is negative.
srcPos+length is greater than src.length, the length of the source array.
destPos+length is greater than dest.length, the length of the destination array.
Otherwise, if any actual component of the source array from position srcPos through srcPos+length-1 cannot be converted to the component type of the destination array by assignment conversion, an ArrayStoreException is thrown. In this case, let k be the smallest nonnegative integer less than length such that src[srcPos+k] cannot be converted to the component type of the destination array; when the exception is thrown, source array components from positions srcPos through srcPos+k-1 will already have been copied to destination array positions destPos through destPos+k-1 and no other positions of the destination array will have been modified. (Because of the restrictions already itemized, this paragraph effectively applies only to the situation where both arrays have component types that are reference types.)

Parameters:
src - the source array.
srcPos - starting position in the source array.
dest - the destination array.
destPos - starting position in the destination data.
length - the number of array elements to be copied.
Throws:
IndexOutOfBoundsException - if copying would cause access of data outside array bounds.
ArrayStoreException - if an element in the src array could not be stored into the dest array because of a type mismatch.
NullPointerException - if either src or dest is null.

Back to Question without Answer

80. QID - 2.888 : Working with Inheritance

What will the following code print when run?

```
class A {
}

class AA extends A {
}


public class TestClass {
    public static void main(String[] args) throws Exception {
        A a = new A();
        AA aa = new AA();
        a = aa;
        System.out.println("a = "+a.getClass());
        System.out.println("aa = "+aa.getClass());
    }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw ClassCastException at runtime.

C. a = class AA
aa = class AA

~~**D.**~~ a = class A
aa = class AA

Explanation:

At the time of printing the class names, both - a and aa, are pointing to an object of class AA. So both will print AA.

Back to Question without Answer

81. QID - 2.900 : Working with Inheritance

What, if anything, is wrong with the following code?

```
interface T1{
}
interface T2{
   int VALUE = 10;
   void m1();
}

interface T3 extends T1, T2{
   public void m1();
   public void m1(int x);
}
```

Correct Option is : B

~~**A.**~~ T3 cannot implement both T1 and T2 because it leads to ambiguity.

B. There is nothing wrong with the code.

~~**C.**~~ The code will work fine only if VALUE is removed from T2 interface.

~~**D.**~~ The code will work fine only if m1() is removed from either T2 and T3.

~~**E.**~~ None of the above.

Explanation:

Having ambiguous fields or methods does not cause any problems by themselves but

referring to such fields/methods in an ambiguous way will cause a compile time error. `T3.m1()` is also fine because even though `m1()` is declared in `T2` as well as `T3` , the definition to both resolves unambiguously to only one `m1()`. Explicit cast is not required for calling the method `m1()` : `( ( T2) t).m1();`

`m1(int x)` is valid because it is a totally independent method declared by `T3`.

Back to Question without Answer

82. QID - 2.832 : Working with Methods

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showOne(100);
        System.out.println(ct.myValue);
        ct.showTwo(200);
        System.out.println(ct.myValue);
    }
}
```

Correct Option is : C

~~**A.**~~ 0 followed by 100.

~~**B.**~~ 100 followed by 100.

C. 0 followed by 200.

~~**D.**~~ 100 followed by 200.

Explanation:

There are a couple of important concepts in this question:

1. Within an instance method, you can access the current object of the same class using '`this`'. Therefore, when you access `this.myValue`, you are accessing the instance member `myValue` of the `ChangeTest` instance.

2. If you declare a local variable (or a method parameter) with the same name as the instance field name, the local variable "shadows" the member field. Ideally, you should be able to access the member field in the method directly by using the name of the member (in this example, `myValue`). However, because of shadowing, when you use `myValue`, it refers to the local variable instead of the instance field.

Within the `showOne()` method, there are two variables accessible with the same name `myValue`. One is the method parameter and another is the member field of `ChangeTest` object. Ideally, you should be able to access the member field in the method directly by using the name `myValue` but in this case, the method parameter shadows the member field because it has the same name. So when you use `myValue`, you are actually using the method parameter instead of the member field.

Therefore, when you do : `myValue = myValue;` you are actually assigning the value contained in method parameter myValue to itself. You are not changing the member field `myValue`. Hence, when you do `System.out.println(ct);` in the next line, it prints `0`.

Now, in `showTwo()`, you are assigning the value contained in `myValue` (i.e. `200`) to `this.myValue`, which is the instance member. Therefore, in the next line, when you print ct.myValue, it prints `200`.

Back to Question without Answer

83. QID - 2.919 : Java Basics - Garbage Collection

After which line will the object created at line XXX be eligible for garbage collection?

```
public Object getObject(Object a) //0
   {

Object b = new Object();  //XXX

Object c, d = new Object(); //1
c = b; //2
b = a = null; //3
return c; //4
}
```

Correct Option is : D

~~**A.**~~ //2

~~**B.**~~ //3

~~**C.**~~ //4

D. Never in this method.

~~**E.**~~ Cannot be determined.

Explanation:

Note that at line 2, c is assigned the reference of b. i.e. c starts pointing to the object created at //XXX. So even if at //3 b and a are set to null, the object is not without any reference.
Also, at //4 c is being returned. So the object referred to by c cannot be garbage collected in this method!

Back to Question without Answer

84. QID - 2.909 : Working with Java Data Types - Variables and Objects

Which of the following declaration are valid:

```
1.  bool b = null;

2. boolean b = 1;

3. boolean b = true|false;

4 bool b = (10<11);

5. boolean b = true||false;
```

Correct Option is : D

~~**A.**~~ 1 and 4

~~**B.**~~ 2, 3, and 5

~~**C.**~~ 2 and 3

D. 3 and 5

~~**E.**~~ 5

Explanation:

`bool` is an invalid keyword. Therefore, 1 and 4 can't be right. (Although 1 could be right if `bool` were a user defined class but as per Java coding conventions, a class name should start with a capital letter.)

`boolean b = 1;` is wrong because you can only assign `true` or `false` to a `boolean` variable. 1 is an integral value it cannot be converted to `boolean`. Also note that `boolean b = null;` would be invalid as well because `null` is not a true or false value. A primitive (whether it is a boolean or an int or a double), can never be assigned `null`.

`boolean b = true|false;` and `boolean b = true||false;` are both valid and the difference between true|false and true||false is not material in this case. However, there is a lot of difference between | (and &) and || (and &&) as explained below:

|| and && perform short circuit evaluation, while & and | do not. Which means, if you use the || and && forms, Java will not bother to evaluate the right-hand operand if the result of the expression can be known by just evaluating the left hand operand.

Consider the following example.

```
Boolean b = true;
if(b || foo.timeConsumingCall()) {
  //entered here without calling timeConsumingCall()
}
```

Another example:

```
String s = null;
if(s != null && s.isEmpty())  //No NullPointerException because
string.isEmpty() is not called.
//If you use & instead of && , s.isEmpty will be called and a
NullPointerException will be thrown.{
   ...
}
```

Back to Question without Answer

85. QID - 2.896 : Constructors

Which of the following classes have a default constructor?

```
class A{  }
class B {  B(){ } }
class C{  C(String s){ } }
```

Correct Option is : A

A. A

~~**B.**~~ A and B

~~**C.**~~ B

~~**D.**~~ C

Since class C has a constructor defined in it, the default constructor will not be provided for it by the compiler.

~~**E.**~~ B and C

Explanation:

There is only one rule regarding the "default" constructor:
The Java compiler automatically adds a constructor that takes no argument and has the same access as the class, if and only if the programmer does not define ANY constructor in the class.

In this case, the programmer has not defined any constructor for class A, hence it will

have the default constructor.

For class B, the programmer has defined a constructor that is exactly same as the default constructor that would have been provided automatically. It is a matter of interpretation whether it can be called a default constructor or not.

Based on Java Language Specification section 8.8.9, quoted below, our interpretation is that class B will not get a default constructor:

(`http://docs.oracle.com/javase/specs/jls/se7/html/jls-8.html`)

8.8.9 Default Constructor
If a class contains no constructor declarations, then a default constructor with no formal parameters and no throws clause is implicitly declared.

If the class being declared is the primordial class Object, then the default constructor has an empty body. Otherwise, the default constructor simply invokes the superclass constructor with no arguments.

It is a compile-time error if a default constructor is implicitly declared but the superclass does not have an accessible constructor (6.6) that takes no arguments and has no throws clause.

It follows that if the nullary constructor of the superclass has a throws clause, then a compile-time error will occur.

Back to Question without Answer

86. QID - 2.881 : Handling Exceptions

Java's Exception mechanism helps in which of the following ways?

Correct Options are : A B

A. It allows creation of new exceptions that are custom to a particular application domain.

> You can define your own exceptions based of your application business domain. For example, in a banking application, you might want to create a InsufficientFundsException. This increases code clarity as compared to having a single (or a few standard) exception class(es) and looking at the exception code to determine what happened.

B. It improves code because error handling code is clearly separated from the main program logic.

> The error handling logic is put in the catch block, which makes the main flow of the program clean and easily understandable.

~~**C.**~~ It enhances the security of the application by reporting errors in the logs.

> Exception handling as such as nothing to do with the security of the application.

~~**D.**~~ It improves the code because the exception is handled right at the place where it occured.

> Just the opposite is true. It improves the code because the code does not have to include error handling code if it is not capable of handling it. It can propagate the exception up the chain and it can be handled at a somewhere at a more appropriate place.

~~**E.**~~ It provides a vast set of standard exceptions that covers all possible exceptions.

Although it does provide a vast set of standard exceptions, they cannot cover all scenarios. But you can always create new exceptions tailored for your application.

Back to Question without Answer

87. QID - 2.883 : Java Basics - Garbage Collection

When is the Object created at line //1 eligible for garbage collection?

```
public class TestClass{
  public Object getObject(){
     Object obj = new String("aaaaa");    //1
     Object objArr[] = new Object[1]; //2
     objArr[0] = obj; //3
     obj = null;      //4
     objArr[0] = null;//5
     return obj;      //6
  }
}
```

Correct Option is : D

~~**A.**~~ Just after line 2.

~~**B.**~~ Just after line 3.

~~**C.**~~ Just after line 4.

D. Just after line 5.

~~**E.**~~ Just after line 6.

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

After line 3, both, `obj` and `objArr[0]` are pointing to the same String object.
After line 4, `obj` points to `null` but `objArr[0]` is still pointing to the String object.
After line 5, `objArr[0]` also starts pointing to `null` so there is no reference left that is pointing to the String object. So it is now available for Garbage collection.

Back to Question without Answer

88. QID - 2.861 : Working with Java Data Types - String, StringBuilder

You want to find out whether two strings are equal or not, in terms of the actual characters within the strings. What is the best way to do this?

Correct Option is : A

A. use String's equals method.

For example:

```
String x1 = "a";
String x2 = new String("a");
```

`x1.equals(x2)` will return true. Because even though x1 and x2 are pointing to different objects, the content of the objects are same, which is what String's equals method checks.

`x1 == x2` will return false, because == only checks if the two references are pointing to the same object or not. In this case, they are not.

~~**B.**~~ use String's `equalsIgnoreCase` method.

If you use this method, `"a"` will be considered equal to `"A"`, which is not what the question is asking for.

~~**C.**~~ Use == operator.

== checks for the equality of the references and not for the equality of the objects themselves. Therefore, this will return true only if two string references are pointing to the same String object, which is not what the question is asking for.

~~**D.**~~ Use String's match method.

There is no method named match in String class.
There is a matches method, which checks whether the String matches a regular expression but that is beyond the scope of this exam.

```
public boolean matches(String regex)
```

Tells whether or not this string matches the given regular expression.
An invocation of this method of the form str.matches(regex) yields exactly the same result as the expression Pattern.matches(regex, str)

Back to Question without Answer

89. QID - 2.876 : Encapsulation

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){         this.side = length;     }

    public double getSide() {  return side;     }

    public void setSide(double side) {  this.side = side;    }

    double getArea() {   return area;    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Correct Options are : D E

~~**A.**~~ Make `setSide()` method private.

~~**B.**~~ Make `getArea()` method private.

It should be made public so that other classes can get the area.

~~**C.**~~ Make `side` and `area` fields private.

> There is no need to keep the area field because that would amount to duplicating the data. If you change side, the value of area will become obsolete.

D. Make the side field private and remove the area field.

E. Change getArea method to:

```
public double getArea(){ return side*side; }
```

~~**F.**~~ Add a `setArea()` method.

> This is not required because area is calculated using the side. So if you allow other classes to set the area, it could make side and area inconsistent with each other.

Explanation:

There can be multiple ways to accomplish this. The exam asks you questions on the similar pattern.
The key is that your data variable should be private and the functionality that is to be exposed outside should be public. Further, your setter methods should be coded such that they don't leave the data members inconsistent with each other.

Back to Question without Answer

90. QID - 2.1365 : Handling Exceptions

Which of the following standard java exception classes extend `java.lang.RuntimeException`?

Correct Options are : A B C E

A. `java.lang.SecurityException`

> `SecurityException extends RuntimeException`: Usually thrown by the JVM. It is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception.

B. `java.lang.ClassCastException`

> `ClassCastException extends RuntimeException`: Usually thrown by the JVM. Thrown to indicate that the code has attempted to cast an object to a subclass of which it is not an instance. For example, the following code generates a `ClassCastException`:
>
> ```
> Object x = new Integer(0);
> System.out.println((String)x);
> ```

C. `java.lang.NullPointerException`

> `NullPointerException extends RuntimeException`: Usually thrown by the JVM. Thrown when an application attempts to use `null` in a case where an object is required. These include:
>
> Calling the instance method of a null object.
> Accessing or modifying the field of a null object.
> Taking the length of null as if it were an array.
> Accessing or modifying the slots of null as if it were an array.

Throwing null as if it were a Throwable value.
Applications should throw instances of this class to indicate other illegal uses of the null object.

D. ~~`java.lang.CloneNotSupportedException`~~

`public class CloneNotSupportedException extends Exception`
Thrown to indicate that the clone method in class Object has been called to clone an object, but that the object's class does not implement the Cloneable interface.
Applications that override the clone method can also throw this exception to indicate that an object could not or should not be cloned.

E. `java.lang.IndexOutOfBoundsException`

`IndexOutOfBoundsException extends RuntimeException`:
Usually thrown by the JVM. Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range. Applications can subclass this class to indicate similar exceptions.
`ArrayIndexOutOfBoundsException and StringIndexOutOfBoundsException` both extend `IndexOutOfBoundsException`.

Explanation:

The other two exceptions you should know about are:
`IllegalArgumentException` extends `RuntimeException`: If a parameter passed to a method is not valid. Usually thrown by the application.

`IllegalStateException extends RuntimeException`: Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation. Usually thrown by the application.

Back to Question without Answer

Objective wise Questions

Constructors

Exam Objectives -

Differentiate between default and user-defined constructors Create and overload constructors

01. QID - 2.947

Which line contains a valid constructor in the following class definition?

```
public class TestClass{
   int i, j;
   public TestClass getInstance() {  return new TestClass();     }  /,
   public void TestClass(int x, int y) {   i = x;   j = y;    }     /,
   public TestClass TestClass() {   return new TestClass();    }    /,
   public ~TestClass() {     }                    //4
}
```

Select 1 option

A. Line 1

B. Line 2

C. Line 3

D. Line 4

E. None of the above.

Check Answer

02. QID - 2.1098

What will be the result of attempting to compile the following program?

```
public class TestClass{
   long l1;
   public void TestClass(long pLong) { l1 = pLong ; }  //(1)
   public static void main(String args[]){
      TestClass a, b ;
      a = new TestClass();  //(2)
      b = new TestClass(5);  //(3)
   }
}
```

Select 1 option

A. A compilation error will be encountered at (1), since constructors should not specify a return value.

B. A compilation error will be encountered at (2), since the class does not have a default constructor.

C. A compilation error will be encountered at (3).

D. The program will compile correctly.

E. It will not compile because parameter type of the constructor is different than the type of value passed to it.

Check Answer

03. QID - 2.1216

Which of these statements are true?

Select 2 options

A. All classes must explicitly define a constructor.

B. A constructor can be declared private.

C. A constructor can declare a return value.

D. A constructor must initialize all the member variables of a class.

E. A constructor can access the non-static members of a class.

Check Answer

04. QID - 2.1018

Note: This question may be considered too advanced for this exam. Given:

```
class MySuper{
    public MySuper(int i){ }
}

abstract class MySub extends MySuper{
    public MySub(int i){ super(i); }
    public abstract void m1();
}

class MyTest{
    public static void main(String[] args){
        MySub ms = new MySub(){
            public void m1() { System.out.println("In MySub.m1()");
        };
        ms.m1();
    }
}
```

What will be the output when the above code is compiled and run?

Select 1 option

A. It will not compile.

B. It will throw an exception at run time.

C. It will print `In MySub.m1()`

D. It will compile and run without any exception but will not print anything.

Check Answer

05. QID - 2.858

Which of the following are true about the "default" constructor?

Select 1 option

A. It is provided by the compiler only if the class and any of its super classes does not define any constructor.

B. It takes no arguments.

C. A default constructor is used to return a default value.

D. To define a default constructor, you must use the default keyword.

E. It is always public.

Check Answer

06. QID - 2.1012

Which lines contain a valid constructor in the following code?

```
public class TestClass{
   public TestClass(int a, int b) { } // 1
   public void TestClass(int a) { }   // 2
   public TestClass(String s); // 3
   private TestClass(String s, int a) { }     //4
   public TestClass(String s1, String s2) { }; //5
}
```

Select 3 options

A. Line // 1

B. Line // 2

C. Line // 3

D. Line // 4

E. Line // 5

Check Answer

07. QID - 2.1102

Under what situations does a class get a default constructor?

Select 1 option

A. All classes in Java get a default constructor.

B. You have to define at least one constructor to get the default constructor.

C. If the class does not define any constructors explicitly.

D. All classes get default constructor from Object class.

E. None of the above.

Check Answer

08. QID - 2.934

You can call only public and protected constructors of the super class from a subclass if the subclass is not in the same package because only those are inherited.

Select 1 option

A. True

B. False

Check Answer

09. QID - 2.1349

Given a class named Test, which of these would be valid definitions for the constructors for the class?

Select 1 option

A. `Test(Test b) { }`

B. `Test Test( ) { }`

C. `private final Test( ) { }`

D. `void Test( ) { }`

E. `public static void Test(String args[ ] ) { }`

Check Answer

10. QID - 2.1355

Given the following code, which of these constructors can be added to class B without causing a compile time error?

```
class A{
   int i;
   public A(int x) { this.i = x; }
}
class B extends A{
   int j;
   public B(int x, int y) { super(x);  this.j = y; }
}
```

Select 2 options

A. `B( ) { }`

B. `B(int y ) { j = y; }`

C. `B(int y ) { super(y*2 ); j = y; }`

D. `B(int y ) { i = y; j = y*2; }`

E. `B(int z ) { this(z, z); }`

Check Answer

Constructors (Answered)

01. QID - 2.947 : Constructors

Which line contains a valid constructor in the following class definition?

```
public class TestClass{
   int i, j;
   public TestClass getInstance() {  return new TestClass();     }   /,
   public void TestClass(int x, int y) {    i = x;    j = y;    }      /,
   public TestClass TestClass() {    return new TestClass();    }     /,
   public ~TestClass() {     }                        //4
}
```

Correct Option is : E

~~**A.**~~ Line 1

> This cannot be a constructor because even the name of the method (getInstance) is not same as the class name!

~~**B.**~~ Line 2

> Constructors cannot return anything. Not even void.

~~**C.**~~ Line 3

> Constructors cannot return anything. Not even void.

~~**D.**~~ Line 4

> This could have been a destructor in C++ world. And there nothing like this in java. Java has a finalize() method, which is similar to a destructor but does not work exactly as a destructor.

E. None of the above.

Back to Question without Answer

02. QID - 2.1098 : Constructors

What will be the result of attempting to compile the following program?

```
public class TestClass{
   long l1;
   public void TestClass(long pLong) { l1 = pLong ; }  //(1)
   public static void main(String args[]){
      TestClass a, b ;
      a = new TestClass();  //(2)
      b = new TestClass(5);  //(3)
   }
}
```

Correct Option is : C

~~**A.**~~ A compilation error will be encountered at (1), since constructors should not specify a return value.

> But it becomes a valid method if you give a return type.

~~**B.**~~ A compilation error will be encountered at (2), since the class does not have a default constructor.

> The class has an implicit default constructor since the class doesn't have any constructor defined.

C. A compilation error will be encountered at (3).

> Because (1) is a method and not a constructor. So there is no constructor that take a parameter.

~~**D.**~~ The program will compile correctly.

~~E.~~ It will not compile because parameter type of the constructor is different than the type of value passed to it.

> If (1) was a valid constructor 'int' would be promoted to long at the time of passing.

Explanation:

The declaration at `(1)` declares a method, not a constructor because it has a return value. The method happens to have the same name as the class, but that is ok.
The class has an implicit default constructor since the class contains no constructor declarations. This allows the instantiation at `(2)` to work.

Back to Question without Answer

03. QID - 2.1216 : Constructors

Which of these statements are true?

Correct Options are : B E

~~**A.**~~ ~~All classes must explicitly define a constructor.~~

> A default no args one will be provided if not defined any.

B. A constructor can be declared private.

> This feature is used for implementing Singleton Classes.

~~**C.**~~ ~~A constructor can declare a return value.~~

~~**D.**~~ ~~A constructor must initialize all the member variables of a class.~~

> All non-final instance variables get default values if not explicitly initialized.

E. A constructor can access the non-static members of a class.

> A constructor is non-static, and so it can access directly both the static and non-static members of the class.

Explanation:

Constructors need not initialize *all* the member variables of the class. A non-final member(i.e. an instance) variable will be assigned a default value if not explicitly initialized.

Back to Question without Answer

04. QID - 2.1018 : Constructors

Note: This question may be considered too advanced for this exam. Given:

```
class MySuper{
    public MySuper(int i){ }
}

abstract class MySub extends MySuper{
    public MySub(int i){ super(i); }
    public abstract void m1();
}

class MyTest{
    public static void main(String[] args){
        MySub ms = new MySub(){
            public void m1() { System.out.println("In MySub.m1()");
        };
        ms.m1();
    }
}
```

What will be the output when the above code is compiled and run?

Correct Option is : A

A. It will not compile.

When you define and instantiate an anonymous inner class for an abstract class as being done here, the anonymous class is actually a subclass of the abstract class (in this case, it is a subclass of MySub). Now, since the anonymous class does not define any constructor, the compiler will add the default no-args constructor to the anonymous class, which will try to call the no-args constructor of its super class MySub. But MySub doesn't have any no-args constructor. Therefore, it will not compile. The anonymous inner class should have been created like this:

```
MySub ms = new MySub( someInteger ){ ... };
```

~~**B.**~~ It will throw an exception at run time.

~~**C.**~~ It will print `In MySub.m1()`

~~**D.**~~ It will compile and run without any exception but will not print anything.

Back to Question without Answer

05. QID - 2.858 : Constructors

Which of the following are true about the "default" constructor?

Correct Option is : B

~~**A.**~~ It is provided by the compiler only if the class and any of its super classes does not define any constructor.

> It is provided by the compiler if the class does not define any constructor. It is immaterial if the super class provides a constructor or not.

B. It takes no arguments.

~~**C.**~~ A default constructor is used to return a default value.

> A constructor does not return any value at all. It is meant to initialize the state of an object.

~~**D.**~~ To define a default constructor, you must use the default keyword.

~~**E.**~~ It is always public.

> The access type of a default constructor is same as the access type of the class. Thus, if a class is public, the default constructor will be public.

Explanation:

The default constructor is provided by the compiler only when a class does not define ANY constructor explicitly. For example,

```
public class A{
```

```
  public A()  //This constructor is automatically inserted by the cor
  {
    super();  //Note that it calls the super class's default no-args
  }
}
public class A{
  //Compiler will not generate any constructor because the programme
  public A(int i){
     //do something
  }
}
```

Back to Question without Answer

06. QID - 2.1012 : Constructors

Which lines contain a valid constructor in the following code?

```
public class TestClass{
   public TestClass(int a, int b) { } // 1
   public void TestClass(int a) { }   // 2
   public TestClass(String s); // 3
   private TestClass(String s, int a) { }     //4
   public TestClass(String s1, String s2) { }; //5
}
```

Correct Options are : A D E

A. Line // 1

~~**B.**~~ Line // 2

> Constructors cannot return anything. Not even void.

~~**C.**~~ Line // 3

> Constructors cannot have empty bodies (i.e. they cannot be abstract)

D. Line // 4

> You can apply public, private, protected to a constructor. But not static, final, synchronized, native and abstract.

E. Line // 5

> The compiler ignores the extra semi-colon.

Explanation:

It is interesting to note that `public void TestClass(int a) {} // 2` will actually compile. It is not a constructor, but compiler considers it as a valid method!

Back to Question without Answer

07. QID - 2.1102 : Constructors

Under what situations does a class get a default constructor?

Correct Option is : C

~~**A.**~~ All classes in Java get a default constructor.

> No. If a class defines a constructor explicitly, it will not get the default constructor.

~~**B.**~~ You have to define at least one constructor to get the default constructor.

> A default (no args one) will be given if the class doesn't define any.

C. If the class does not define any constructors explicitly.

> In this case, the compiler will add a no args constructor for this class.

~~**D.**~~ All classes get default constructor from Object class.

> Constructors are NEVER inherited.

~~**E.**~~ None of the above.

Back to Question without Answer

08. QID - 2.934 : Constructors

You can call only public and protected constructors of the super class from a subclass if the subclass is not in the same package because only those are inherited.

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

Most of the statement is correct but remember that constructors are NEVER (whether public or otherwise) inherited. (The above statement is true for other methods though.) So, if you have a class :

```
class A{
     public A(int i){}
}
```

and another class:

```
class B extends A{
}
```

You cannot do : new B(10); because that is A's constructor and it is not inherited by B. To invoke A's constructor you have to do:

```
class B extends A{
     public B(int i){  super(i); }
```

```
}
```

Back to Question without Answer

09. QID - 2.1349 : Constructors

Given a class named Test, which of these would be valid definitions for the constructors for the class?

Correct Option is : A

A. `Test(Test b) { }`

> The constructor can take the same type as a parameter.

~~B.~~ `Test Test( ) { }`

> A constructor cannot return anything.

~~C.~~ `private final Test( ) { }`

> A constructor cannot be final, static or abstract.

~~D.~~ `void Test( ) { }`

> A constructor cannot return anything not even void.

~~E.~~ `public static void Test(String args[ ] ) { }`

> A constructor cannot be final, static or abstract.

Back to Question without Answer

10. QID - 2.1355 : Constructors

Given the following code, which of these constructors can be added to class B without causing a compile time error?

```
class A{
   int i;
   public A(int x) { this.i = x; }
}
class B extends A{
   int j;
   public B(int x, int y) { super(x);  this.j = y; }
}
```

Correct Options are : C E

~~**A.**~~ `B( ) { }`

~~**B.**~~ `B(int y ) { j = y; }`

C. `B(int y ) { super(y*2 ); j = y; }`

~~**D.**~~ `B(int y ) { i = y; j = y*2; }`

E. `B(int z ) { this(z, z); }`

Explanation:

1.To construct an instance of a sub class, its super class needs need to be constructed first. Since an instance can only be created via a constructor, some constructor of the

super class has to be invoked.
Either you explicitly call it or the compiler will add `super()` (i.e. no args constructor) as the first line of the sub class constructor.

Now, if the super class (here, A) does not have a no args constructor, the call `super()` will fail. Hence, choices `B( ) { }`, `B(int y ) { j = y; }` and `B(int y ) { i = y; j = y*2; }` are not valid and choice `B(int y ) { super(y*2 ); j = y; }` is valid because it explicitly calls `super( int )` which is available in A.

2.Instead of calling `super(...)` you can also call another constructor of the base class in the first line (as given in choice `B(int z ) { this(z, z); }`). Here, `this(int, int)` is called in the first line which in turn calls `super(int)`. So the super class A is correctly instantiated.

Back to Question without Answer

Creating and Using Arrays

Exam Objectives -

Declare, instantiate, initialize and use a one-dimensional array Declare, instantiate, initialize and use multi-dimensional array Declare and use an ArrayList

01. QID - 2.897

Given the following code :

```
public class TestClass {

    int[][] matrix = new int[2][3];

    int a[] = {1, 2, 3};
    int b[] = {4, 5, 6};

    public int compute(int x, int y){
        //1 : Insert Line of Code here
    }

    public void loadMatrix(){
        for(int x=0; x<matrix.length; x++){
            for(int y=0; y<matrix[x].length; y++){
                //2: Insert Line of Code here
            }
        }
    }
}
```

What can be inserted at //1 and //2?

Select 1 option

A. `return a(x)*b(y);`
and
`matrix(x, y) = compute(x, y);`

B. `return a[x]*b[y];`
and
`matrix[x, y] = compute(x, y);`

C. `return a[x]*b[y];`
and
`matrix[x][y] = compute(x, y);`

D. `return a(x)*b(y);`
and
`matrix(x)(y) = compute(x, y);`

E. `return a[x]*b[y];`
and
`matrix[[x][y]] = compute(x, y);`

Check Answer

02. QID - 2.1286

Consider the following array definitions:

```
int[] array1, array2[];
int[][] array3;
int[] array4[], array5[];
```

Which of the following are valid statements?

Select 3 options

A. `array2 = array3;`

B. `array2 = array4;`

C. `array1 = array2;`

D. `array4 = array1;`

E. `array5 = array3`

Check Answer

03. QID - 2.987

What will be the result of attempting to run the following program?

```
public class StringArrayTest{
   public static void main(String args[]){
      String[][][] arr  ={{ { "a", "b" , "c"}, { "d", "e", null } },
{ {"x"}, null },{{"y"}},{ { "z","p"}, {} }
      };
      System.out.println(arr[0][1][2]);
   }
}
```

Select 1 option

A. It will throw NullPointerBoundsException.

B. It will throw ArrayIndexOutOfBoundsException.

C. It will print null.

D. It will run without any error but will print nothing.

E. None of the above.

Check Answer

04. QID - 2.843

Given the following declaration:

```
int[][] twoD = { { 1, 2, 3} , { 4, 5, 6, 7}, { 8, 9, 10 } };
```

What will the following lines of code print?

```
System.out.println(twoD[1].length);
System.out.println(twoD[2].getClass().isArray());
System.out.println(twoD[1][2]);
```

Select 1 option

A.
```
4
true
6
```

B.
```
3
true
3
```

C.
```
3
false
3
```

D.
```
4
false
6
```

Check Answer

05. QID - 2.1214

Consider the following program:

```
public class TestClass{
  public static void main(String[] args){
    String tom = args[0];
    String dick = args[1];
    String harry = args[2];
  }
}
```

What will the value of 'harry' if the program is run from the command line:
java TestClass 111 222 333

Select 1 option

A. 111

B. 222

C. 333

D. It will throw an `ArrayIndexOutOfBoundsException`

E. None of the above.

Check Answer

06. QID - 2.971

What will be the result of trying to compile and execute of the following program?

```
public class TestClass{
   public static void main(String args[] ){
      int i = 0 ;
      int[] iA = {10, 20} ;
      iA[i] = i = 30 ;
      System.out.println(""+ iA[ 0 ] + " " + iA[ 1 ] + "  "+i) ;
    }
}
```

Select 1 option

A. It will throw ArrayIndexOutofBoundsException at Runtime.

B. Compile time Error.

C. It will prints 10 20 30

D. It will prints 30 20 30

E. It will prints 0 20 30

Check Answer

07. QID - 2.946

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
    public static void main(String args[ ] ){
       int i = 1;
       int[] iArr = {1};
       incr(i) ;
       incr(iArr) ;
       System.out.println( "i = " + i + "  iArr[0] = " + iArr [ 0 ] );
    }
    public static void incr(int   n ) { n++ ; }
    public static void incr(int[ ] n ) { n [ 0 ]++ ; }
}
```

Select 1 option

A. `The code will print i = 1 iArr[0] = 1;`

B. `The code will print i = 1 iArr[0] = 2;`

C. `The code will print i = 2 iArr[0] = 1;`

D. `The code will print i = 2 iArr[0] = 2;`

E. The code will not compile.

Check Answer

08. QID - 2.938

What will the following program print?

```
class Test{
   public static void main(String[] args){
      int i = 4;
      int ia[][][] = new int[i][i = 3][i];
      System.out.println( ia.length + ", " + ia[0].length+", "+ ia[0
   }
}
```

Select 1 option

A. It will not compile.

B. 3, 4, 3

C. 3, 3, 3

D. 4, 3, 4

E. 4, 3, 3

Check Answer

09. QID - 2.1296

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    String str = "111";
    boolean[] bA = new boolean[1];
    if( bA[0] ) str = "222";
    System.out.println(str);
  }
}
```

Select 1 option

A. 111

B. 222

C. It will not compile as `bA[0]` is uninitialized.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

10. QID - 2.1264

What sequence of digits will the following program print?

```
import java.util.* ;
public class ListTest{
   public static void main(String args[]){
      List s1 = new ArrayList( );
      s1.add("a");
      s1.add("b");
      s1.add(1, "c");
      List s2 = new ArrayList(  s1.subList(1, 1) );
      s1.addAll(s2);
      System.out.println(s1);
   }
}
```

Select 1 option

A. The sequence a, b, c is printed.

B. The sequence a, b, c, b is printed.

C. The sequence a, c, b, c is printed.

D. The sequence a, c, b is printed.

E. None of the above.

Check Answer

11. QID - 2.1115

Consider the following program...

```
class ArrayTest{
   public static void main(String[] args){
      int ia[][] = { {1, 2}, null };
      for (int i = 0; i < 2; i++)
         for (int j = 0; j < 2; j++)
            System.out.println(ia[i][j]);
   }
}
```

Which of the following statements are true?

Select 1 option

A. It will not compile.

B. It will throw an `ArrayIndexOutOfBoundsException` at Runtime.

C. It will throw a `NullPointerException` at Runtime.

D. It will compile and run without throwing any exceptions.

E. None of the above.

Check Answer

12. QID - 2.1124

Given the following declaration, select the correct way to get the number of elements in the array, assuming that the array has been initialized.

```
int[] intArr;
```

Select 1 option

A. `intArr[ ].length( )`

B. `intArr.length( )`

C. `intArr.length`

D. `intArr[ ].size( )`

E. `intArr.size( )`

Check Answer

13. QID - 2.1036

Consider the following class...

```
class Test{
   public static void main(String[ ] args){
      int[] a = { 1, 2, 3, 4 };
      int[] b = { 2, 3, 1, 0 };
      System.out.println( a [ (a = b)[3] ] );
   }
}
```

What will it print when compiled and run ?

Select 1 option

A. It will not compile.

B. It will throw `ArrayIndexOutOfBoundsException` when run.

C. It will print 1.

D. It will print 3.

E. It will print 4

Check Answer

14. QID - 2.1076

What would be the result of compiling and running the following program?

```
class SomeClass{
   public static void main(String args[]){
      int size = 10;
      int[] arr = new int[size];
      for (int i = 0 ; i < size ; ++i) System.out.println(arr[i]);
   }
}
```

Select 1 option

A. The code will fail to compile, because the `int[]` array declaration is incorrect.

B. The program will compile, but will throw an `IndexArrayOutOfBoundsException` when run.

C. The program will compile and run without error, and will print nothing.

D. The program will compile and run without error and will print `null` ten times.

E. The program will compile and run without error and will print `0` ten times.

Check Answer

15. QID - 2.1244

Given the following program, which statements are true?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      A[] a, a1;
      B[] b;
      a = new A[10]; a1  = a;
      b =  new B[20];
      a = b;  // 1
      b = (B[]) a;  // 2
      b = (B[]) a1; // 3
   }
}
class A { }
class B extends A { }
```

Select 2 options

A. Compile time error at line 3.

B. The program will throw a java.lang.ClassCastException at the line labelled 2 when run.

C. The program will throw a java.lang.ClassCastException at the line labelled 3 when run.

D. The program will compile and run if the (B[ ]) cast in the line 2 and the whole line 3 is removed.

E. The cast at line 2 is needed.

Check Answer

16. QID - 2.1068

Which of the following statements are valid ?

Select 2 options

A. `String[ ] sa = new String[3]{ "a", "b", "c"};`

B. `String sa[ ] = { "a ", " b", "c"};`

C. `String sa = new String[ ]{"a", "b", "c"};`

D. `String sa[ ] = new String[ ]{"a", "b", "c"};`

E. `String sa[ ] = new String[ ] {"a" "b" "c"};`

Check Answer

17. QID - 2.913

Which of the following statements about an array are correct?

Select 1 option

A. An array can dynamically grow in size.

B. Arrays can be created only for primitive types.

C. Every array has a built in property named 'size' which tells you the number of elements in the array.

D. Every array has in implicit method named '`length`' which tells you the number of elements in the array.

E. Element indexing starts at 0.

Check Answer

18. QID - 2.1202

What would be the result of trying to compile and run the following program?

```
public class Test{
   int[] ia = new int[1];
   Object oA[]  = new Object[1];
   boolean bool;
   public static void main(String args[]){
      Test test = new Test();
      System.out.println(test.ia[0] + "  " +
test.oA[0]+"  "+test.bool);
   }
}
```

Select 1 option

A. The program will fail to compile, because of uninitialized variable 'bool'.

B. The program will throw a java.lang.NullPointerException when run.

C. The program will print "0 null false".

D. The program will print "0 null true".

E. The program will print null and false but will print junk value for ia[0].

Check Answer

19. QID - 2.1354

Which of the following correctly declare a variable which can hold an array of 10 integers?

Select 2 options

A. `int[ ] iA`

B. `int[10] iA`

C. `int iA[ ]`

D. `Object[ ] iA`

E. `Object[10] iA`

Check Answer

20. QID - 2.1191

Which of the following code fragments will successfully initialize a two-dimensional array of chars named cA with a size such that cA[2][3] refers to a valid element?

```
1.
  char[][] cA = {  { 'a', 'b', 'c' },  { 'a', 'b', 'c' }   };
2.
  char cA[][] = new char[3][];
  for (int i=0; i<cA.length; i++) cA[i] = new char[4];
3.
  char cA[][] = { new char[ ]{ 'a', 'b', 'c' }  ,   new char[ ]{
'a', 'b', 'c' }  };
4
  char cA[3][2] = new char[][] {  { 'a', 'b', 'c' },   { 'a', 'b',
'c' }   };
5.
  char[][] cA = { "1234", "1234",  "1234"  };
```

Select 1 option

A. 1, 3

B. 4, 5

C. 2, 3

D. 1, 2, 3

E. 2

Check Answer

21. QID - 2.969

Which of these array declarations and initializations are NOT legal?

Select 2 options

A.
```
int[ ] i[ ] = { { 1, 2 }, { 1 }, { }, { 1, 2, 3 } } ;
```

B.
```
int i[ ] = new int[2] {1, 2} ;
```

C.
```
int i[ ][ ] = new int[ ][ ] { {1, 2, 3}, {4, 5, 6} } ;
```

D.
```
int i[ ][ ] = { { 1, 2 }, new int[ 2 ] } ;
```

E.
```
int i[4] = { 1, 2, 3, 4 } ;
```

Check Answer

22. QID - 2.1326

What will happen when the following code is compiled and run?

```
class AX{
  static int[] x = new int[0];
  static{
   x[0] = 10;
  }
  public static void main(String[] args){
    AX ax = new AX();
  }
}
```

Select 1 option

A. It will throw `NullPointerException` at runtime.

B. It will throw `ArrayIndexOutOfBoundsException` at runtime.

C. It will throw `ExceptionInInitializerError` at runtime.

D. It will not compile.

Check Answer

23. QID - 2.1262

Consider the following code snippet ...

```
boolean[] b1 = new boolean[2];
boolean[] b2 = {true , false};
System.out.println( "" + (b1[0] == b2[0]) + ", "+ (b1[1] ==
b2[1])  );
```

What will it print ?

Select 1 option

A. It will not compile.

B. It will throw ArrayIndexOutOfBoundsError at Runtime.

C. false, true

D. true, false

E. It will print false, false.

Check Answer

24. QID - 2.1029

The following class will print 'index = 2' when compiled and run.

```
class Test{
   public static int[ ] getArray() {  return null;  }
   public static void main(String[] args){
      int index = 1;
      try{
         getArray()[index=2]++;
      }
      catch (Exception e){  }  //empty catch
      System.out.println("index = " + index);
   }
}
```

Select 1 option

A. True

B. False

Check Answer

25. QID - 2.1106

Is it possible to create arrays of length zero?

Select 1 option

A. Yes, you can create arrays of any type with length zero.

B. Yes, but only for primitive datatypes.

C. Yes, but only for arrays of object references.

D. Yes, and it is same as a null Array.

E. No, arrays of length zero do not exist in Java.

Check Answer

26. QID - 2.1137

What will the following code snippet print?

```
int index = 1;
String[] strArr = new String[5];
String   myStr  = strArr[index];
System.out.println(myStr);
```

Select 1 option

A. nothing

B. `null`

C. It will throw `ArrayIndexOutOfBounds` at runtime.

D. It will print some junk value.

E. None of the above.

Check Answer

27. QID - 2.1316

Which of the following statements will correctly create and initialize an array of Strings to non null elements?

Select 4 options

A. `String[] sA = new String[1] { "aaa"};`

B. `String[] sA = new String[] { "aaa"};`

C. `String[] sA = new String[1] ; sA[0] = "aaa";`

D. `String[] sA = {new String( "aaa")};`

E. `String[] sA = { "aaa"};`

Check Answer

28. QID - 2.1229

Which of these array declarations and instantiations are legal?

Select 4 options

A. `int[ ] a[ ] = new int [5][4] ;`

B. `int a[ ][ ] = new int [5][4] ;`

C. `int a[ ][ ] = new int [ ][4] ;`

D. `int[ ] a[ ] = new int[4][ ] ;`

E. `int[ ][ ] a = new int[5][4] ;`

Check Answer

29. QID - 2.942

Consider the following code to count objects and save the most recent object ...

```
int i = 0 ;
Object prevObject ;
public void saveObject(List e ){
    prevObject = e ;
    i++ ;
}
```

Which of the following calls will work without throwing an exception?

Select 3 options

A. `saveObject( new ArrayList() );`

B. `Collection c = new ArrayList(); saveObject( c );`

C. `List l = new ArrayList(); saveObject(l);`

D. `saveObject(null);`

E. `saveObject(0);` //The argument is the number zero and not the letter o

Check Answer

30. QID - 2.850

Identify the correct statements about `ArrayList`?

Select 3 options

A. ArrayList extends java.util.AbstractList.

B. It allows you to access its elements in random order.

C. You must specify the class of objects you want to store in `ArrayList` when you declare a variable of type ArrayList.

D. `ArrayList` does not implement `RandomAccess`.

E. You can sort its elements using `Collections.sort()` method.

Check Answer

31. QID - 2.851

Identify the correct statements about ArrayList?

Select 3 options

A. Standard JDK provides no subclasses of ArrayList.

B. You cannot store primitives in an ArrayList.

C. It allows constant time access to all its elements.

D. ArrayList cannot resize dynamically if you add more number of elements than its capacity.

E. An ArrayList is backed by an array.

Check Answer

32. QID - 2.1307

Given:

```
double daaa[][][] = new double[3][][];
double d = 100.0;
double[][] daa = new double[1][1];
```

Which of the following will not cause any problem at compile time or runtime?

Select 2 options

A. `daaa[0] = d;`

B. `daaa[0] = daa;`

C. `daaa[0] = daa[0];`

D. `daa[1][1] = d;`

E. `daa = daaa[0]`

Check Answer

33. QID - 2.873

Which of the following are benefits of ArrayList over an array?

Select 1 option

A. You do not have to worry about the size of the ArrayList while inserting elements.

B. It consumes less memory space.

C. You do not have to worry about thread safety.

D. It allows you to write type safe code.

Check Answer

34. QID - 2.870

What will the following code print when compiled and run?

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<String> al = new ArrayList<String>();
        al.add("111");
        al.add("222");
        System.out.println(al.get(al.size()));
     }
}
```

Select 1 option

A. It will not compile.

B. It will throw a `NullPointerException` at run time.

C. It will throw an `IndexOutOfBoundsException` at run time.

D. `222`

E. `null`

Check Answer

Creating and Using Arrays (Answered)

01. QID - 2.897 : Creating and Using Arrays

Given the following code :

```
public class TestClass {

    int[][] matrix = new int[2][3];

    int a[] = {1, 2, 3};
    int b[] = {4, 5, 6};

    public int compute(int x, int y){
        //1 : Insert Line of Code here
    }

    public void loadMatrix(){
        for(int x=0; x<matrix.length; x++){
            for(int y=0; y<matrix[x].length; y++){
                //2: Insert Line of Code here
            }
        }
    }
}
```

What can be inserted at //1 and //2?

Correct Option is : C

~~**A.**~~ `return a(x)*b(y);`

and

`matrix(x, y) = compute(x, y);`

> (and) are used to call a method on an object. To access array elements, you need to use [and].

~~**B.**~~ `return a[x]*b[y];`

and

```
matrix[x, y] = compute(x, y);
```

C. `return a[x]*b[y];`

and

```
matrix[x][y] = compute(x, y);
```

~~**D.**~~ `return a(x)*b(y);`

and

```
matrix(x)(y) = compute(x, y);
```

> `a(x), b(y),` and `matrix(x)(y)` are invalid because a, b, and matrix are not methods.

~~**E.**~~ `return a[x]*b[y];`

and

```
matrix[[x][y]] = compute(x, y);
```

> `[[x][y]]` is invalid syntax.

Explanation:

The correct syntax to access any element within an array is to use the square brackets - `[ ]`. Thus, to access the first element in an array, you would use `array[0]`.

For a multi dimensional array, to reach an individual item, you need to specify index for each dimension. For example, since matrix is a two dimensional array, matrix is an array of array and `matrix[0]` will give you the first array of the arrays. `matrix[0][0]` will give you the first element of the first array of the arrays.

Back to Question without Answer

02. QID - 2.1286 : Creating and Using Arrays

Consider the following array definitions:

```
int[] array1, array2[];
int[][] array3;
int[] array4[], array5[];
```

Which of the following are valid statements?

Correct Options are : A B E

A. `array2 = array3;`

B. `array2 = array4;`

~~C.~~ `array1 = array2;`

~~D.~~ `array4 = array1;`

E. `array5 = array3`

Explanation:

There is a subtle difference between: `int[] i;` and `int i[];` although in both the cases, i is an array of integers.
The difference is if you declare multiple variables in the same statement such as:
`int[] i, j;` and `int i[], j;`, j is not of the same type in the two cases.
In the first case, j is an array of integers while in the second case, j is just an integer.
Therefore, in this question:
`array1` is an array of `int`
`array2, array3, array4,` and `array5` are arrays of `int` arrays

Therefore, option 1, 2 and 5 are valid.

Back to Question without Answer

03. QID - 2.987 : Creating and Using Arrays

What will be the result of attempting to run the following program?

```
public class StringArrayTest{
   public static void main(String args[]){
      String[][][] arr  ={{ { "a", "b" , "c"}, { "d", "e", null } },
{ {"x"}, null },{{"y"}},{ { "z","p"}, {} }
      };
      System.out.println(arr[0][1][2]);
   }
}
```

Correct Option is : C

~~**A.**~~ It will throw NullPointerBoundsException.

There is no such exception.

~~**B.**~~ It will throw ArrayIndexOutOfBoundsException.

C. It will print null.

~~**D.**~~ It will run without any error but will print nothing.

~~**E.**~~ None of the above.

Explanation:

arr[0][1][2] => [0] = { { "a", "b" , "c"}, { "d", "e", null } }, [1] = { "d", "e", null } and [2] = null.
So it will print null.

Back to Question without Answer

04. QID - 2.843 : Creating and Using Arrays

Given the following declaration:

```
int[][] twoD = { { 1, 2, 3} , { 4, 5, 6, 7}, { 8, 9, 10 } };
```

What will the following lines of code print?

```
System.out.println(twoD[1].length);
System.out.println(twoD[2].getClass().isArray());
System.out.println(twoD[1][2]);
```

Correct Option is : A

A.
```
4
true
6
```

~~B.~~
```
3
true
3
```

~~C.~~
```
3
false
3
```

~~D.~~
```
4
false
6
```

Explanation:

In Java, array numbering starts from 0. So in this case, `twoD` is an array containing 3

other arrays.
`twoD[0]` is `{ 1, 2, 3}` , `twoD[1]` is `{ 4, 5, 6, 7}`, and `twoD[2]` is `{ 8, 9, 10 }`.

Thus, `twoD[1].length` is `4` and `twoD[1][2]` is the third element in `{ 4, 5, 6, 7}`, which is `6`.

In Java, arrays are just like regular Objects and arrays of different types have different class names. For example, the class name of an `int[]` is `[I` and the class name for `int[][]` is `[[I`.

For array classes, the `isArray()` method of a Class returns true. For example, `twoD.getClass().isArray()` will return true.

There are a few questions in the exam that require you to know about this.

Back to Question without Answer

05. QID - 2.1214 : Creating and Using Arrays

Consider the following program:

```
public class TestClass{
  public static void main(String[] args){
    String tom = args[0];
    String dick = args[1];
    String harry = args[2];
  }
}
```

What will the value of 'harry' if the program is run from the command line:
java TestClass 111 222 333

Correct Option is : C

~~**A.**~~ 111

~~**B.**~~ 222

C. 333

~~**D.**~~ It will throw an `ArrayIndexOutOfBoundsException`

~~**E.**~~ None of the above.

Explanation:

java and classname are not part of the args array. So tom gets "111", dick gets "222" and harry gets "333".

Back to Question without Answer

06. QID - 2.971 : Creating and Using Arrays

What will be the result of trying to compile and execute of the following program?

```
public class TestClass{
   public static void main(String args[] ){
      int i = 0 ;
      int[] iA = {10, 20} ;
      iA[i] = i = 30 ;
      System.out.println(""+ iA[ 0 ] + " " + iA[ 1 ] + "  "+i) ;
    }
}
```

Correct Option is : D

~~**A.**~~ It will throw ArrayIndexOutofBoundsException at Runtime.

~~**B.**~~ Compile time Error.

~~**C.**~~ It will prints 10 20 30

D. It will prints 30 20 30

~~**E.**~~ It will prints 0 20 30

Explanation:

The statement iA[i] = i = 30 ; will be processed as follows:
iA[i] = i = 30; => iA[0] = i = 30 ; => i = 30; iA[0] = i ; => iA[0] = 30 ;

Here is what JLS says on this:

1 Evaluate Left-Hand Operand First
2 Evaluate Operands before Operation
3 Evaluation Respects Parentheses and Precedence
4 Argument Lists are Evaluated Left-to-Right

For Arrays: First, the dimension expressions are evaluated, left-to-right. If any of the expression evaluations completes abruptly, the expressions to the right of it are not evaluated.

Back to Question without Answer

07. QID - 2.946 : Creating and Using Arrays

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
    public static void main(String args[ ] ){
       int i = 1;
       int[] iArr = {1};
       incr(i) ;
       incr(iArr) ;
       System.out.println( "i = " + i + "  iArr[0] = " + iArr [ 0 ] );
    }
    public static void incr(int   n ) { n++ ; }
    public static void incr(int[ ] n ) { n [ 0 ]++ ; }
}
```

Correct Option is : B

~~**A.**~~ `The code will print i = 1 iArr[0] = 1;`

B. `The code will print i = 1 iArr[0] = 2;`

~~**C.**~~ `The code will print i = 2 iArr[0] = 1;`

~~**D.**~~ `The code will print i = 2 iArr[0] = 2;`

~~**E.**~~ The code will not compile.

There is no problem with the code.

Explanation:

Arrays are proper objects (i.e. `iArr instanceof Object` returns `true`) and Object references are passed by value (so effectively, it seems as though objects are being passed by reference).
So the value of reference of `iArr` is passed to the method `incr(int[] i);` This method changes the actual value of the int element at 0.

Back to Question without Answer

08. QID - 2.938 : Creating and Using Arrays

What will the following program print?

```
class Test{
   public static void main(String[] args){
      int i = 4;
      int ia[][][] = new int[i][i = 3][i];
      System.out.println( ia.length + ", " + ia[0].length+", "+ ia[0
   }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

~~**B.**~~ 3, 4, 3

~~**C.**~~ 3, 3, 3

~~**D.**~~ 4, 3, 4

E. 4, 3, 3

Explanation:

In an array creation expression, there may be one or more dimension expressions, each within brackets. Each dimension expression is fully evaluated before any part of any dimension expression to its right. The first dimension is calculated as 4 before the second dimension expression sets 'i' to 3.

Note that if evaluation of a dimension expression completes abruptly, no part of any dimension expression to its right will appear to have been evaluated.

Back to Question without Answer

09. QID - 2.1296 : Creating and Using Arrays

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    String str = "111";
    boolean[] bA = new boolean[1];
    if( bA[0] ) str = "222";
    System.out.println(str);
  }
}
```

Correct Option is : A

A. 111

~~**B.**~~ 222

~~**C.**~~ It will not compile as `bA[0]` is uninitialized.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

All the arrays are initialized to contain the default values of their type. This means, `int[] iA = new int[10];` will contain 10 integers with a value of 0. `Object[] oA = new Object[10];` will contain 10 object references pointing to `null`.

`boolean[] bA = new boolean[10]` will contain 10 booleans of value `false`. So, as `bA[0]` is `false`, the if condition fails and `str` remains `111`.

Back to Question without Answer

10. QID - 2.1264 : Creating and Using Arrays

What sequence of digits will the following program print?

```
import java.util.* ;
public class ListTest{
   public static void main(String args[]){
      List s1 = new ArrayList( );
      s1.add("a");
      s1.add("b");
      s1.add(1, "c");
      List s2 = new ArrayList(  s1.subList(1, 1) );
      s1.addAll(s2);
      System.out.println(s1);
   }
}
```

Correct Option is : D

~~**A.**~~ The sequence a, b, c is printed.

~~**B.**~~ The sequence a, b, c, b is printed.

~~**C.**~~ The sequence a, c, b, c is printed.

D. The sequence a, c, b is printed.

add(1, "c") will insert 'c' between 'a' and 'b' . sublist(1 , 1) will return an empty list.

~~**E.**~~ None of the above.

Explanation:

First, "a" and "b" are appended to an empty list. Next, "c" is added between "1" and "2".
Then a new list s2 is created using the sublist view allowing access to elements from index 1 to index 1(exclusive) (i.e. no elements).
Now, s2 is added to s1.
So s1 remains :a, c, b

Back to Question without Answer

11. QID - 2.1115 : Creating and Using Arrays

Consider the following program...

```
class ArrayTest{
   public static void main(String[] args){
      int ia[][] = { {1, 2}, null };
      for (int i = 0; i < 2; i++)
         for (int j = 0; j < 2; j++)
            System.out.println(ia[i][j]);
   }
}
```

Which of the following statements are true?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an `ArrayIndexOutOfBoundsException` at Runtime.

C. It will throw a `NullPointerException` at Runtime.

~~**D.**~~ It will compile and run without throwing any exceptions.

~~**E.**~~ None of the above.

Explanation:

It will throw a `NullPointerException` for `ia[1][0]` because `ia[1]` is `null`. Note that `null` is not same as having less number of elements in an array than

expected.

If you try to access `ia[2][0]`, it would have thrown `ArrayIndexOutOfBoundsException` because the length of `ia` is only 2 and so `ia[2]` tries to access an element out of that range. `ia[2]` is not `null`, it simply does not exist.

Back to Question without Answer

12. QID - 2.1124 : Creating and Using Arrays

Given the following declaration, select the correct way to get the number of elements in the array, assuming that the array has been initialized.

```
int[] intArr;
```

Correct Option is : C

~~A.~~ `intArr[ ].length( )`

~~B.~~ `intArr.length( )`

C. `intArr.length`

> Each array object has a member variable named public final length of type 'int' that contains the size of the array.

~~D.~~ `intArr[ ].size( )`

~~E.~~ `intArr.size( )`

Explanation:

FYI, All types of arrays are objects. i.e. `intArr instanceof Object` is `true`.

Back to Question without Answer

13. QID - 2.1036 : Creating and Using Arrays

Consider the following class...

```
class Test{
   public static void main(String[ ] args){
      int[] a = { 1, 2, 3, 4 };
      int[] b = { 2, 3, 1, 0 };
      System.out.println( a [ (a = b)[3] ] );
   }
}
```

What will it print when compiled and run ?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw `ArrayIndexOutOfBoundsException` when run.

C. It will print 1.

~~**D.**~~ It will print 3.

~~**E.**~~ It will print 4

Explanation:

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated. In the expression `a[(a=b)[3]]`, the expression a is fully evaluated before the

expression `(a=b)[3];` this means that the original value of a is fetched and remembered while the expression `(a=b)[3]` is evaluated. This array referenced by the original value of a is then subscripted by a value that is element 3 of another array (possibly the same array) that was referenced by b and is now also referenced by a. So, it is actually `a[0] = 1`.
Note that if evaluation of the expression to the left of the brackets completes abruptly, no part of the expression within the brackets will appear to have been evaluated.

Back to Question without Answer

14. QID - 2.1076 : Creating and Using Arrays

What would be the result of compiling and running the following program?

```
class SomeClass{
   public static void main(String args[]){
      int size = 10;
      int[] arr = new int[size];
      for (int i = 0 ; i < size ; ++i) System.out.println(arr[i]);
   }
}
```

Correct Option is : E

~~**A.**~~ The code will fail to compile, because the `int[]` array declaration is incorrect.

~~**B.**~~ The program will compile, but will throw an `IndexArrayOutOfBoundsException` when run.

~~**C.**~~ The program will compile and run without error, and will print nothing.

~~**D.**~~ The program will compile and run without error and will print `null` ten times.

> Here, all the array elements are initialized to have 0.

E. The program will compile and run without error and will print `0` ten times.

> It correctly will declare and initialize an array of length `10` containing `int` values initialized to have `0`.

Explanation:

Elements of Arrays of primitive types are initialized to their default value (i.e. 0 for integral types, 0.0 for float/double and false for boolean)
Elements of Arrays of non-primitive types are initialized to null.

Back to Question without Answer

15. QID - 2.1244 : Creating and Using Arrays

Given the following program, which statements are true?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      A[] a, a1;
      B[] b;
      a = new A[10]; a1  = a;
      b =  new B[20];
      a = b;  // 1
      b = (B[]) a;  // 2
      b = (B[]) a1; // 3
   }
}
class A { }
class B extends A { }
```

Correct Options are : C E

~~**A.**~~ Compile time error at line 3.

~~**B.**~~ The program will throw a java.lang.ClassCastException at the line labelled 2 when run.

C. The program will throw a java.lang.ClassCastException at the line labelled 3 when run.

~~**D.**~~ The program will compile and run if the (B[ ]) cast in the line 2 and the whole line 3 is removed.

E. The cast at line 2 is needed.

Explanation:

The line 1 will be allowed during compilation, since assignment is done from a subclass reference to a superclass reference.
The cast in line 2 is needed because a superclass reference is assigned to a subclass reference variable. And this works at runtime because the object referenced to by a is actually of an array of B.
Now, the cast at line 3 tells the compiler not to worry, that I'm a good programmer and I know what I am doing and the object referenced by the super class reference (a1) will actually be of class B at run time. So there is no compile time error. But at run time, this fails because the actual object is not an array of B but is an array of A.

Back to Question without Answer

16. QID - 2.1068 : Creating and Using Arrays

Which of the following statements are valid ?

Correct Options are : B D

~~**A.**~~ `String[ ] sa = new String[3]{ "a", "b", "c"};`

> You cannot specify the length of the array (i.e. 3, here) if you are using the initializer block while declaring the array.

B. `String sa[ ] = { "a ", " b", "c"};`

~~**C.**~~ `String sa = new String[ ]{"a", "b", "c"};`

> here `sa` is not declared as array of strings but just as a String.

D. `String sa[ ] = new String[ ]{"a", "b", "c"};`

~~**E.**~~ `String sa[ ] = new String[ ] {"a" "b" "c"};`

> There are no commas separating the strings.

Back to Question without Answer

17. QID - 2.913 : Creating and Using Arrays

Which of the following statements about an array are correct?

Correct Option is : E

~~**A.**~~ An array can dynamically grow in size.

> Arrays cannot grow in size once created. ArrayLists can do that.

~~**B.**~~ Arrays can be created only for primitive types.

> You can have arrays for objects also. For example:
> ```
> Object[] objArray = new Object[4];
> String[] arrayOfStrings = { "a", "b" };
> ```

~~**C.**~~ Every array has a built in property named 'size' which tells you the number of elements in the array.

> It is named `length` and not `size`. `ArrayList` has a method named `size()` that returns the number of elements in the ArrayList.
>
> ```
> String[] sa = { "a", "b" };
> int k = sa.length; //k will be assigned a value of 2.
>
> ArrayList al = new ArrayList();
> int k = al.size(); //k will be assigned a value of 0.
> ```

~~**D.**~~ Every array has in implicit method named '`length`' which tells you the number of elements in the array.

E. Element indexing starts at 0.

Back to Question without Answer

18. QID - 2.1202 : Creating and Using Arrays

What would be the result of trying to compile and run the following program?

```
public class Test{
   int[] ia = new int[1];
   Object oA[]  = new Object[1];
   boolean bool;
   public static void main(String args[]){
      Test test = new Test();
      System.out.println(test.ia[0] + "  " +
test.oA[0]+"  "+test.bool);
   }
}
```

Correct Option is : C

~~**A.**~~ The program will fail to compile, because of uninitialized variable 'bool'.

No, All the instance variables are initialized by default values.

~~**B.**~~ The program will throw a java.lang.NullPointerException when run.

No reason for this at all.

C. The program will print "0 null false".

~~**D.**~~ The program will print "0 null true".

All the variables, including the array elements, will be initialized to their default values.

~~**E.**~~ The program will print null and false but will print junk value for ia[0].

All the elements of the arrays of primitives are initialized to default values.

Explanation:

Following are the default values that instance variables are initialized with if not initialized explicitly:

types (byte, short, char, int, long, float, double) to 0 (or 0.0).

All Object types to null.

boolean to false.

Back to Question without Answer

19. QID - 2.1354 : Creating and Using Arrays

Which of the following correctly declare a variable which can hold an array of 10 integers?

Correct Options are : A C

A. `int[ ] iA`

~~B.~~ `int[10] iA`

> Size of the array is NEVER specified on the Left Hand Side.

C. `int iA[ ]`

~~D.~~ `Object[ ] iA`

> Here, iA is an array of Objects. It cannot hold an array of integers.

~~E.~~ `Object[10] iA`

> Size of the array is NEVER specified on the LHS.

Explanation:

Note that an array of integers IS an Object :

```
Object obj = new int[]{ 1, 2, 3 }; // is valid.
```

But it is not an Array of objects.

```
Object[] o = new int[10]; // is not valid.
```

Difference between the placement of square brackets:

```
int[] i, j; //here i and j are both array of integers.
```

```
int i[], j; //here only i is an array of integers. j is just an
integer.
```

Back to Question without Answer

20. QID - 2.1191 : Creating and Using Arrays

Which of the following code fragments will successfully initialize a two-dimensional array of chars named cA with a size such that cA[2][3] refers to a valid element?

```
1.
  char[][] cA = {  { 'a', 'b', 'c' },  { 'a', 'b', 'c' }   };
2.
  char cA[][] = new char[3][];
  for (int i=0; i<cA.length; i++) cA[i] = new char[4];
3.
  char cA[][] = { new char[ ]{ 'a', 'b', 'c' }  ,   new char[ ]{
'a', 'b', 'c' }  };
4
  char cA[3][2] = new char[][] {  { 'a', 'b', 'c' },   { 'a', 'b',
'c' }   };
5.
  char[][] cA = { "1234", "1234",  "1234"  };
```

Correct Option is : E

~~**A.**~~ 1, 3

~~**B.**~~ 4, 5

~~**C.**~~ 2, 3

~~**D.**~~ 1, 2, 3

E. 2

Explanation:

1 and 3 declare a two dimensional array alright but they create the array of size 2, 3.
And cA[2][3] means we need an array of
size 3, 4 because the numbering starts from 0.
4 : You cannot put array size information on LHS.
5 : This is a one dimensional array and that too of strings. Note that a java String is not equivalent to 1 dimensional array of chars.
This leaves us with only one choice 2.

Back to Question without Answer

21. QID - 2.969 : Creating and Using Arrays

Which of these array declarations and initializations are NOT legal?

Correct Options are : B E

~~**A.**~~ `int[ ] i[ ] = { { 1, 2 }, { 1 }, { }, { 1, 2, 3 } } ;`

B. `int i[ ] = new int[2] {1, 2} ;`

> If you give the elements explicitly you can't give the size. So it should be just int[] { 1, 2 } or just { 1, 2 }

~~**C.**~~ `int i[ ][ ] = new int[ ][ ] { {1, 2, 3}, {4, 5, 6} } ;`

~~**D.**~~ `int i[ ][ ] = { { 1, 2 }, new int[ 2 ] } ;`

E. `int i[4] = { 1, 2, 3, 4 } ;`

> You cannot specify the size on left hand side .

Explanation:

If you explicitly specify the members then you can't give the size. So option 2 is wrong.
The size of the array is never given during the declaration of an array reference. So option 5 is wrong.
The size of an array is always associated with the array instance, not the array reference.

Back to Question without Answer

22. QID - 2.1326 : Creating and Using Arrays

What will happen when the following code is compiled and run?

```
class AX{
  static int[] x = new int[0];
  static{
   x[0] = 10;
  }
  public static void main(String[] args){
    AX ax = new AX();
  }
}
```

Correct Option is : C

~~**A.**~~ It will throw `NullPointerException` at runtime.

~~**B.**~~ It will throw `ArrayIndexOutOfBoundsException` at runtime.

C. It will throw `ExceptionInInitializerError` at runtime.

The following is the output when the program is run:

```
java.lang.ExceptionInInitializerError
Caused by: java.lang.ArrayIndexOutOfBoundsException: 0
        at AX.<clinit>(SM.java:6)
Exception in thread "main"
Java Result: 1
```

Note that the program ends with `ExceptionInInitializerError` because any exception thrown in a static block is wrapped into `ExceptionInInitializerError` and then that `ExceptionInInitializerError` is thrown.

~~**D.**~~ It will not compile.

Explanation:

Even though the line `x[0] = 10;` will throw java.lang.ArrayIndexOutOfBoundsException, JVM will wrap it and rethrow java.lang.ExceptionInInitializerError.

Back to Question without Answer

23. QID - 2.1262 : Creating and Using Arrays

Consider the following code snippet ...

```
boolean[] b1 = new boolean[2];
boolean[] b2 = {true , false};
System.out.println( "" + (b1[0] == b2[0]) + ", "+ (b1[1] ==
b2[1])  );
```

What will it print ?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw ArrayIndexOutOfBoundsError at Runtime.

C. false, true

~~**D.**~~ true, false

~~**E.**~~ It will print false, false.

Explanation:

Note that whenever you create an array all of its elements are automatically given defaults values. Numeric types are initialized to 0, objects are initialized to null, and booleans to false.

So if you have, float[ ] f = new float[3]; f[0], f[1] and f[2] will all be 0.0.

if you have Object[ ] o = new String[3]; o[0], o[1] and o[2] will all be null.
In this case, b1[0] and b1[1] are false.
whereas b2[0] and b2[1] are true and false.
So the answer is false and true.

Back to Question without Answer

24. QID - 2.1029 : Creating and Using Arrays

The following class will print 'index = 2' when compiled and run.

```
class Test{
   public static int[ ] getArray() {  return null;  }
   public static void main(String[] args){
      int index = 1;
      try{
         getArray()[index=2]++;
      }
      catch (Exception e){  }  //empty catch
      System.out.println("index = " + index);
   }
}
```

Correct Option is : A

A. True

B. False

Explanation:

If the array reference expression produces null instead of a reference to an array, then a NullPointerException is thrown at runtime, but only after all parts of the array reference expression have been evaluated and only if these evaluations completed normally.

This means, first index = 2 will be executed, which assigns 2 to index. After that null[2] is executed, which throws a NullPointerException. But this exception is caught by the catch block, which prints nothing. So it seems like NullPointerException is not thrown but it actually is.

In other words, the embedded assignment of 2 to index occurs before the check for array reference produced by getArray().

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated. Note that if evaluation of the expression to the left of the brackets completes abruptly, no part of the expression within the brackets will appear to have been evaluated.

Back to Question without Answer

25. QID - 2.1106 : Creating and Using Arrays

Is it possible to create arrays of length zero?

Correct Option is : A

A. Yes, you can create arrays of any type with length zero.

> Java allows arrays of length zero to be created.

~~**B.**~~ Yes, but only for primitive datatypes.

~~**C.**~~ Yes, but only for arrays of object references.

~~**D.**~~ Yes, and it is same as a null Array.

> No. A null pointer is different from an array of length Zero. For example, if you have `int[] intArr = new int[0];` then `(intArr == null)` is `false`.

~~**E.**~~ No, arrays of length zero do not exist in Java.

Explanation:

Example: When a Java program is run without any program arguments, the `String[] args` argument to `main()` gets an array of length Zero.

Back to Question without Answer

26. QID - 2.1137 : Creating and Using Arrays

What will the following code snippet print?

```
int index = 1;
String[] strArr = new String[5];
String   myStr  = strArr[index];
System.out.println(myStr);
```

Correct Option is : B

~~**A.**~~ nothing

B. `null`

~~**C.**~~ It will throw `ArrayIndexOutOfBounds` at runtime.

~~**D.**~~ It will print some junk value.

~~**E.**~~ None of the above.

Explanation:

When you create an array of Objects (here, Strings) all the elements are initialized to null. So in the line 3, null is assigned to myStr.
Note that. empty string is "" (String str = "";) and is not same as null.

Back to Question without Answer

27. QID - 2.1316 : Creating and Using Arrays

Which of the following statements will correctly create and initialize an array of Strings to non null elements?

Correct Options are : B C D E

A. ~~`String[] sA = new String[1] { "aaa"};`~~

> Array size cannot be given here as the array is being initialized in the declaration.

B. `String[] sA = new String[] { "aaa"};`

C. `String[] sA = new String[1] ; sA[0] = "aaa";`

D. `String[] sA = {new String( "aaa")};`

E. `String[] sA = { "aaa"};`

Back to Question without Answer

28. QID - 2.1229 : Creating and Using Arrays

Which of these array declarations and instantiations are legal?

Correct Options are : A B D E

A. `int[ ] a[ ] = new int [5][4] ;`

This will create an array of length 5. Each element of this array will be an array of 4 ints.

B. `int a[ ][ ] = new int [5][4] ;`

This will create an array of length 5. Each element of this array will be an array of 4 ints.

C. ~~`int a[ ][ ] = new int [ ][4] ;`~~

The statement int[ ][4] will not compile, because the dimensions must be created from left to right.

D. `int[ ] a[ ] = new int[4][ ] ;`

This will create an array of length 4. Each element of this array will be `null`. But you can assign an array of ints of any length to any of the elements. For example:

```
a[0] = new int[10];//valid
a[1] = new int[4];//valid
a[2] = new int[]; //invalid because you must specify the length
a[3] = new Object[] //invalid because a[3] can only refer to an
array of ints.
```

This shows that while creating a one dimensional array, the length must be

specified but while creating multidimensional arrays, the length of the last dimension can be left unspecified. Further, the length of multiple higher dimensions after the first one can also be left unspecified if none of the dimensions are specified after it. So for example,
`a[][][][] = new int[4][3][3][5];` is same as `a[][][][] = new int[4][][][];` (Note that the first dimension must be specified.)

Thus, multidimensional arrays do not have to be symmetrical.

E. `int[ ][ ] a = new int[5][4] ;`

This will create an array of length 5. Each element of this array will be an array of 4 ints.

Explanation:

The [] notation can be placed both before and after the variable name in an array declaration.

```
int[] ia, ba;  // here ia and ba both are int arrays.
int ia[], ba; //here only ia is int array and ba is an int.
```

Multidimensional arrays are created by creating arrays that can contain references to other arrays .

Back to Question without Answer

29. QID - 2.942 : Creating and Using Arrays

Consider the following code to count objects and save the most recent object ...

```
int i = 0 ;
Object prevObject ;
public void saveObject(List e ){
    prevObject = e ;
    i++ ;
}
```

Which of the following calls will work without throwing an exception?

Correct Options are : A C D

A. `saveObject( new ArrayList() );`

> Because an `ArrayList` is a List.

~~B.~~ `Collection c = new ArrayList(); saveObject( c );`

> `saveObject()` cannot accept c because c is declared of type Collection, which is a superclass of List, but the `saveObject()` method expects a List.

C. `List l = new ArrayList(); saveObject(l);`

D. `saveObject(null);`

> In this case `prevObj` will be set to null.

~~E.~~ `saveObject(0);` //The argument is the number zero and not the letter o

> `0` is an int, which means it is a primitive. So it will be boxed into an `Integer`

object when you pass it to a method that expects an `Object`. However, `Integer` cannot be passed to a method that expects a `List`. Therefore, this option is not valid. Had the method bean `saveObject(Object obj)`, it would have been valid because an `Integer` is an `Object`.

Back to Question without Answer

30. QID - 2.850 : Creating and Using Arrays

Identify the correct statements about `ArrayList`?

Correct Options are : A B E

A. ArrayList extends java.util.AbstractList.

ArrayList is a subclass of AbstractList.

```
java.lang.Object
 -  java.util.AbstractCollection<E>
   -    java.util.AbstractList<E>
     -      java.util.ArrayList<E>
```

All Implemented Interfaces:

```
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, Rand
```

B. It allows you to access its elements in random order.

This is true because you can directly access any element using `get(index)` method. (This is unlike a `LinkedList`, in which you have to go through all the elements occuring before Nth element before you can access the Nth element.)

~~**C.**~~ You must specify the class of objects you want to store in `ArrayList` when you declare a variable of type ArrayList.

This is not true because you can still use non-generic form. For example, instead of using

```
ArrayList<String> listOfStrings;
```

you can use:

```
ArrayList listOfStrings;
```

Of course, if you use non generic version, you will lose the compile time type checking.

~~D.~~ `ArrayList` does not implement `RandomAccess`.

It does.
`RandomAccess` is a marker interface used by List implementations to indicate that they support fast (generally constant time) random access. The primary purpose of this interface is to allow generic algorithms to alter their behavior to provide good performance when applied to either random or sequential access lists.

E. You can sort its elements using `Collections.sort()` method.

An `ArrayList` is a `List` so you can use it where ever a List is required. This include Collections methods such as `sort, reverse, and shuffle`.

Back to Question without Answer

31. QID - 2.851 : Creating and Using Arrays

Identify the correct statements about ArrayList?

Correct Options are : B C E

~~**A.**~~ Standard JDK provides no subclasses of ArrayList.

> It does.
> Direct Known Subclasses:
> AttributeList, RoleList, RoleUnresolvedList

B. You cannot store primitives in an ArrayList.

> This is true because only Objects can be stored in it.

C. It allows constant time access to all its elements.

> This is true because it implements java.util.RandomAccess interface, which is a marker interface that signifies that you can directly access any element of this collection. This also implies that it takes the same amount of time to access any element.
>
> (Compare that with a LinkedList, which takes more time to access the last element than the first element.)

~~**D.**~~ ArrayList cannot resize dynamically if you add more number of elements than its capacity.

> It does resize dynamically. Compare that to an array, which cannot be resized once created.

E. An ArrayList is backed by an array.

> This is true. The elements are actually stored in an array and that is why is it called an ArrayList.
> (The expression "backed by an array" means that the implementation of ArrayList actually uses an array to store elements.)

Explanation:

ArrayList is a subclass of AbstractList.

```
java.lang.Object
-  java.util.AbstractCollection<E>
   -    java.util.AbstractList<E>
     -      java.util.ArrayList<E>
```

All Implemented Interfaces:
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:
AttributeList, RoleList, RoleUnresolvedList

Back to Question without Answer

32. QID - 2.1307 : Creating and Using Arrays

Given:

```
double daaa[][][] = new double[3][][];
double d = 100.0;
double[][] daa = new double[1][1];
```

Which of the following will not cause any problem at compile time or runtime?

Correct Options are : B E

~~**A.**~~ `daaa[0] = d;`

> `daaa[0]` should be a 2 dimensional array because `daaa` is a 3 dimensional array.

B. `daaa[0] = daa;`

~~**C.**~~ `daaa[0] = daa[0];`

> `daaa[0]` should be a 2 dimensional array while `daa[0]` is a one dimensional array.

~~**D.**~~ `daa[1][1] = d;`

> `daa[1][1]` will cause an `ArrayIndexOutofBoundsException` because `daa`'s length is only 1 and the indexing starts from 0. To access the first element, you should use `daa[0][0]`.

E. `daa = daaa[0]`

Back to Question without Answer

33. QID - 2.873 : Creating and Using Arrays

Which of the following are benefits of ArrayList over an array?

Correct Option is : A

A. You do not have to worry about the size of the ArrayList while inserting elements.

> An ArrayList resized dynamically at run time as per the situation. An array cannot be resized once created. This reduces the amount of boiler plate code that is required to do the same task using an array.

~~**B.**~~ It consumes less memory space.

> Because of additional internal data structure and pointers, it actually consumes a little more memory than an array.

~~**C.**~~ You do not have to worry about thread safety.

> An ArrayList, just like an array is not thread safe. If you have multiple threads trying to add and remove elements from an ArrayList, you have to write additional code to ensure thread safety.

~~**D.**~~ It allows you to write type safe code.

> Since ArrayList is a generics enabled class, it helps you write type safe code. For example, if you have:
> `ArrayList<String> al = new ArrayList<>();`
> `al.add(new Integer(10));` will not compile because the compiler knows that al can only contain Strings.
>
> However, this is not an advantage over an array because arrays are also type

safe. For example, if you have:

```
String[] sa = new String[10];
```

you cannot do `sa[0] = new Integer(10);` either.

But you can do `Object[] oa = sa;` and `oa[0]  = new Integer(10);` This will compile fine but will fail at runtime. This is a hole in the type safety provided by arrays.

Back to Question without Answer

34. QID - 2.870 : Creating and Using Arrays

What will the following code print when compiled and run?

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        ArrayList<String> al = new ArrayList<String>();
        al.add("111");
        al.add("222");
        System.out.println(al.get(al.size()));
     }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw a `NullPointerException` at run time.

C. It will throw an `IndexOutOfBoundsException` at run time.

size() method of ArrayList returns the number of elements. Here, it returns 2. Since numbering in ArrayList starts with 0. al.get(2) will cause an IndexOutOfBoundsException to be thrown because only 0 and 1 are valid indexes for a list of size 2.

~~**D.**~~ `222`

~~**E.**~~ `null`

Back to Question without Answer

Encapsulation

Exam Objectives -

Apply encapsulation principles to a class

01. QID - 2.901

Given:

```
public class Triangle{
    public int base;
    public int height;
    public double area;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }

    void updateArea(){
        area = base*height/2;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

The above class needs to protect an invariant on the "area" field. Which three members must have the public access modifiers removed to ensure that the invariant is maintained?

Select 3 options

A. the base field

B. the height field

C. the area field

D. the Triangle constructor

E. the setBase method

F. the setHeight method

Check Answer

02. QID - 2.1322

What is meant by "encapsulation" ?

Select 1 option

A. There is no way to access member variable.

B. There are no member variables.

C. Member fields are declared private but public accessor/mutator methods are provided to access and change their values.

D. Data fields are declared public and accessor methods are provided to access and change their values.

E. None of the above.

Check Answer

03. QID - 2.877

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){        this.side = length;    }

    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

    double getArea() {   return area;   }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Select 4 options

A. Add a calculateArea method:

```
private void calculateArea(){
  this.area = this.side*this.side;
}
```

B. Make `side` and `area` fields private.

C. Change setSide method to:

```
public void setSide(double d){
this.side = d;
calculateArea();
}
```

D. Make the `getArea` method public.

E. Add a `setArea() method:`

```
public void setArea(double d){ area = d; }
```

Check Answer

04. QID - 2.1213

When a class whose members should be accessible only to members of that class is coded such a way that its members are accessible to other classes as well, this is called ...

Select 1 option

A. strong coupling

B. weak coupling

C. strong typing

D. weak encapsulation

E. weak polymorphism

F. high cohesion

G. low cohesion

Check Answer

05. QID - 2.997

Consider the following class written by a novice programmer.

```
class Elliptical{
    public int radiusA, radiusB;
    public int sum = 100;

    public void setRadius(int r){
        if(r>99) throw new IllegalArgumentException();
        radiusA = r;
        radiusB = sum - radiusA;

    }
}
```

After some time, the requirements changed and the programmer now wants to make sure that radiusB is always (200 - radiusA) instead of (100 - radiusA) without breaking existing code that other people have written. Which of the following will accomplish his goal?

Select 1 option

A. `Make sum = 200;`

B. `Make sum = 200` and make it private.

C. `Make sum = 200` and make all fields (radiusA, radiusB, and sum) private.

D. Write another method `setRadius2(int r)` and `set radiusB` accordingly in this method.

E. His goal cannot be accomplished.

F. This class will not compile.

Check Answer

Encapsulation (Answered)

01. QID - 2.901 : Encapsulation

Given:

```
public class Triangle{
    public int base;
    public int height;
    public double area;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }

    void updateArea(){
        area = base*height/2;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

The above class needs to protect an invariant on the "area" field. Which three members must have the public access modifiers removed to ensure that the invariant is maintained?

Correct Options are : A B C

A. the base field

B. the height field

C. the area field

~~**D.**~~ the Triangle constructor

~~**E.** the setBase method~~

~~**F.** the setHeight method~~

Explanation:

An invariant means a certain condition that constrains the state stored in the object. For example, in this case the value of the area field of the Triangle must always be consistent with its base and height fields. Thus, it should never have a value that is different from `base*height/2`.

If you allow other classes to directly change the value of base, height, or area, using direct field access, the area field may not contain the correct area thereby breaking the invariant.

To prevent this inconsistency from happening, you need to prohibit changing the instance fields directly and instead permit the changes only through the setter method because these methods call the updateArea method and keep the area and base and height consistent.

Back to Question without Answer

02. QID - 2.1322 : Encapsulation

What is meant by "encapsulation" ?

Correct Option is : C

~~**A.**~~ There is no way to access member variable.

~~**B.**~~ There are no member variables.

C. Member fields are declared private but public accessor/mutator methods are provided to access and change their values.

~~**D.**~~ Data fields are declared public and accessor methods are provided to access and change their values.

~~**E.**~~ None of the above.

Explanation:

Encapsulation is one of the 4 fundamentals of OOP (Object Oriented Programming).

Encapsulation means that the internal representation of an object is generally hidden from view outside of the object's definition. Typically, only the object's own methods can directly inspect or manipulate its fields. Some languages like Smalltalk and Ruby only allow access via object methods, but most others (e.g. C++ or Java) offer the programmer a degree of control over what is hidden, typically via keywords like public and private.

Hiding the internals of the object protects its integrity by preventing users from setting the internal data of the component into an invalid or inconsistent state. A benefit of encapsulation is that it can reduce system complexity, and thus increases robustness, by allowing the developer to limit the interdependencies between software components.

Back to Question without Answer

03. QID - 2.877 : Encapsulation

Consider the following two classes (in the same package but defined in different source files):

```
public class Square {
    double side = 0;
    double area;

    public Square(double length){        this.side = length;    }

    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

    double getArea() {   return area;   }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Square sq = new Square(10.0);
        sq.area = sq.getSide()*sq.getSide();
        System.out.println(sq.getArea());
    }
}
```

You are assigned the task of refactoring the Square class to make it better in terms of encapsulation. What changes will you make to this class?

Correct Options are : A B C D

A. Add a calculateArea method:

```
private void calculateArea(){
  this.area = this.side*this.side;
}
```

B. Make `side` and `area` fields private.

C. Change setSide method to:

```
public void setSide(double d){
this.side = d;
calculateArea();
}
```

D. Make the `getArea` method public.

~~**E.**~~ Add a `setArea() method:`

```
public void setArea(double d){ area = d; }
```

> This is not required because area is calculated using the side. So if you allow other classes to set the area, it could make side and area inconsistent with each other.

Explanation:

There can be multiple ways to accomplish this. The exam asks you questions on the similar pattern.
The key is that your data variable should be private and the functionality that is to be exposed outside should be public. Further, your setter methods should be coded such that they don't leave the data members inconsistent with each other.

Back to Question without Answer

04. QID - 2.1213 : Encapsulation

When a class whose members should be accessible only to members of that class is coded such a way that its members are accessible to other classes as well, this is called ...

Correct Option is : D

~~**A.** strong coupling~~

~~**B.** weak coupling~~

~~**C.** strong typing~~

D. weak encapsulation

~~**E.** weak polymorphism~~

~~**F.** high cohesion~~

~~**G.** low cohesion~~

Explanation:

When a class is properly encapsulated, only the members that are part of its public API are publicly accessible to other classes. Rest is all private or protected.

Back to Question without Answer

05. QID - 2.997 : Encapsulation

Consider the following class written by a novice programmer.

```
class Elliptical{
    public int radiusA, radiusB;
    public int sum = 100;

    public void setRadius(int r){
        if(r>99) throw new IllegalArgumentException();
        radiusA = r;
        radiusB = sum - radiusA;

    }
}
```

After some time, the requirements changed and the programmer now wants to make sure that radiusB is always (200 - radiusA) instead of (100 - radiusA) without breaking existing code that other people have written. Which of the following will accomplish his goal?

Correct Option is : E

~~**A.**~~ `Make sum = 200;`

~~**B.**~~ `Make sum = 200` and make it private.

~~**C.**~~ `Make sum = 200` and make all fields (radiusA, radiusB, and sum) private.

> This should have been done when the class was first written.

~~**D.**~~ Write another method `setRadius2(int r)` and `set radiusB` accordingly in this method.

E. His goal cannot be accomplished.

~~**F.**~~ This class will not compile.

There is no problem with the code. Remember, IllegalArgumentException extends from RuntimeException and is a super class of NumberFormatException

Explanation:

setRadius method makes sure that radiusB is set to sum - radiusA. So changing sum to 200 should do it. However, note that radiusA, radiusB, and sum are public which means that any other class can access these fields directly without going through the setRadius method. So there is no way to make sure that the value of radiusB is correctly set at all times. If you make them private now, other classes that are accessing the fields directly will break.

The class should have been coded with proper encapsulation of the fields in the first place.

Back to Question without Answer

Handling Exceptions

Exam Objectives -

Differentiate among checked exceptions, RuntimeExceptions and Errors Create a try-catch block and determine how exceptions alter normal program flow Describe what exceptions are used for in Java Invoke a method that throws an exception Recognize common exception classes and categories

01. QID - 2.1305

Which of these statements are true?

Select 2 options

A. If a RuntimeException is not caught, the method will terminate and normal execution of the thread will resume.

B. An overriding method must declare that it throws the same exception classes as the method it overrides.

C. The main method of a program can declare that it throws checked exceptions.

D. A method declaring that it throws a certain exception class may throw instances of any subclass of that exception class.

E. finally blocks are executed if and only if an exception gets thrown while inside the corresponding try block.

Check Answer

02. QID - 2.1345

Assume that a method named 'method1' contains code which may raise a non-runtime (checked) Exception.
What is the correct way to declare that method so that it indicates that it expects the caller to handle that exception?

Select 2 options

A. `public void method1() throws Throwable`

B. `public void method1() throw Exception`

C. `public void method1() throw new Exception`

D. `public void method1() throws Exception`

E. `public void method1()`

Check Answer

03. QID - 2.1034

Identify the exceptions that will be received when the code snippets on the left hand side are executed.

1.
```
int factorial(int n){
 if(n==1) return 1;
 else return n*factorial(n-1);
}
```
Assume that it is called with a very big integer.

2.
```
void printMe(Object[] oa){
 for(int i=0; i<=oa.length; i++)
    System.out.println(oa[i]);
}
```
Assume that it is called as such: printMe(null);

3.
```
Object m1(){
 return new Object();
}
void m2(){
  String s = (String) m1();
}
```

ClassCastException ArrayIndexOutOfBoundsException NullPointerException

StackOverflowError Will Not Compile. No Exception Will Be Thrown.

Check Answer

04. QID - 2.1133

Which of the following can be thrown using a throw statement?

Select 3 options

A. Event

B. Object

C. Throwable

D. Exception

E. RuntimeException

Check Answer

05. QID - 2.1006

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}
class AnotherException extends Exception {}
public class ExceptionTest{
   public static void main(String [] args) throws Exception{
      try{
         m2();
      }
      finally{ m3(); }
    }
    public static void m2() throws NewException{  throw new NewExcep
    public static void m3() throws AnotherException{  throw new Anot
}
```

Select 1 option

A. It will compile but will throw `AnotherException` when run.

B. It will compile but will throw `NewException` when run.

C. It will compile and run without throwing any exceptions.

D. It will not compile.

E. None of the above.

Check Answer

06. QID - 2.1112

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}

class AnotherException extends Exception {}

public class ExceptionTest{
    public static void main(String[] args) throws Exception{
        try{
            m2();
        }
        finally{
            m3();
        }
        catch (NewException e){}
    }

    public static void m2() throws NewException { throw new NewExcep

    public static void m3() throws AnotherException{ throw new Anothe

}
```

Select 1 option

A. It will compile but will throw `AnotherException` when run.

B. It will compile but will throw `NewException` when run.

C. It will compile and run without throwing any exceptions.

D. It will not compile.

E. None of the above.

Check Answer

07. QID - 2.880

Which of the following are standard Java exception classes?

Select 2 options

A. `FileNotFoundException`

B. `InputException`

C. `CPUError`

D. `MemoryException`

E. `SecurityException`

Check Answer

08. QID - 2.826

What will be the output when the following program is run?

```
package exceptions;
public class TestClass {
    public static void main(String[] args) {
        try{
            doTest();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void doTest() throws MyException{
        int[] array = new int[10];
        array[10] = 1000;
        doAnotherTest();
    }

    static void doAnotherTest() throws MyException{
        throw new MyException("Exception from doAnotherTest");
    }
}
class MyException extends Exception {
    public MyException(String msg){
     super(msg);
    }
}
```

Select 1 option

A.
```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 10
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

B.
```
Error in thread "main" java.lang.ArrayIndexOutOfBoundsException
```

C.
```
exceptions.MyException: Exception from doAnotherTest
```

D.
```
exceptions.MyException: Exception from doAnotherTest
    at exceptions.TestClass.doAnotherTest(TestClass.java:29)
    at exceptions.TestClass.doTest(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

Check Answer

09. QID - 2.967

What will the following code print when compiled and run?

```
abstract class Calculator{
   abstract void calculate();
   public static void main(String[] args){
      System.out.println("calculating");
      Calculator x = null;
      x.calculate();
   }
}
```

Select 1 option

A. It will not compile.

B. It will not print anything and will throw `NullPointerException`

C. It will print `calculating` and then throw `NullPointerException`.

D. It will print `calculating` and will throw `NoSuchMethodError`

E. It will print `calculating` and will throw `MethodNotImplementedException`

Check Answer

10. QID - 2.1235

A Java programmer is developing a desktop application. Which of the following exceptions would be appropriate for him to throw explicitly from his code?

Select 1 option

A. `NullPointerException`

B. `ClassCastException`

C. `ArrayIndexOutofBoundsException`

D. `Exception`

E. `NoClassDefFoundError`

Check Answer

11. QID - 2.1148

What will the following code print?

```
public class Test{
   public int luckyNumber(int seed){
      if(seed > 10) return seed%10;
         int x = 0;
            try{
               if(seed%2 == 0) throw new Exception("No Even no.");
               else return x;
            }
            catch(Exception e){
               return 3;
            }
            finally{
               return 7;
            }
         }

         public static void main(String args[]){
            int amount = 100, seed = 6;
            switch( new Test().luckyNumber(6) ){
                case 3: amount = amount * 2;
                case 7: amount = amount * 2;
                case 6: amount = amount + amount;
                default :
            }
           System.out.println(amount);
        }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. 800

D. 200

E. 400

Check Answer

12. QID - 2.1347

Consider the following code...

```
public class TestClass{
     class MyException extends Exception {}
     public void myMethod() throws XXXX{
         throw new MyException();
     }
}
```

What can replace XXXX?

Select 3 options

A. MyException

B. Exception

C. No throws clause is necessary

D. Throwable

E. RuntimeException

Check Answer

13. QID - 2.1033

A try statement must always have a ..... associated with it.

Select 1 option

A. catch

B. throws

C. finally

D. catch, finally or both

E. throw

Check Answer

14. QID - 2.1005

Consider the following code:

```
class A {
    public void doA(int k) throws Exception {  // 0
        for(int i=0; i< 10; i++) {
            if(i == k) throw new Exception("Index of k is "+i); // 1
        }
    }
    public void doB(boolean f) { // 2
        if(f) {
            doA(15); // 3
        }
        else return;
    }
    public static void main(String[] args) { // 4
        A a = new A();
        a.doB(args.length>0); // 5
    }
}
```

Which of the following statements are correct?

Select 1 option

A. This will compile and run without any errors or exception.

B. This will compile if `throws Exception` is added at line //2

C. This will compile if `throws Exception` is added at line //4

D. This will compile if `throws Exception` is added at line //2 as well as //4

E. This will compile if line marked // 1 is enclosed in a try - catch block.

Check Answer

15. QID - 2.1350

What will the following code snippet print:

```
Float f = null;
try{
    f = Float.valueOf("12.3");
    String s = f.toString();
    int i = Integer.parseInt(s);
    System.out.println("i = "+i);
}
catch(Exception e){
    System.out.println("trouble : "+f);
}
```

Select 1 option

A. 12

B. 13

C. trouble : null

D. trouble : 12.3

E. trouble : 0.0

Check Answer

16. QID - 2.1358

What will the following program print when run using the command line: `java TestClass`

```
public class TestClass {

   public static void methodX() throws Exception {
      throw new AssertionError();
   }

   public static void main(String[] args) {
      try{
         methodX();
      }
      catch(Exception e) {
        System.out.println("EXCEPTION");
      }
   }
}
```

Select 1 option

A. It will throw `AssertionError` out of the main method.

B. It will print `EXCEPTION`.

C. It will not compile because of the throws clause in `methodX()`.

D. It will end without printing anything because assertions are disabled by default.

Check Answer

17. QID - 2.1301

What is wrong with the following code written in a single file named TestClass.java?

```
class SomeThrowable extends Throwable { }
class MyThrowable extends SomeThrowable { }
public class TestClass{
   public static void main(String args[]) throws SomeThrowable{
      try{
         m1();
      }catch(SomeThrowable e){
         throw e;
      }finally{
         System.out.println("Done");
      }
   }
   public static void m1() throws MyThrowable{
      throw new MyThrowable();
   }
}
```

Select 2 options

A. The main declares that it throws `SomeThrowable` but throws `MyThrowable`.

B. You cannot have more than 2 classes in one file.

C. The catch block in the main method must declare that it catches `MyThrowable` rather than `SomeThrowable`.

D. There is nothing wrong with the code.

E. `Done` will be printed.

Check Answer

18. QID - 2.950

Consider the following method -

```
public float parseFloat( String s ){
   float f = 0.0f;
   try{
      f = Float.valueOf( s ).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      f = Float.NaN ;
      return f;
   }
   finally{
      f = 10.0f;
      return f;
   }
}
```

What will it return if the method is called with the input "0.0" ?

Select 1 option

A. It will not compile.

B. It will return 10.0

C. It will return Float.Nan

D. It will return 0.0

E. None of the above.

Check Answer

19. QID - 2.1028

Identify the exceptions that SHOULD be thrown in the situations shown below.

```
1.
public void closeReportManager(ReportManager rep)
{
 if(!rep.isClosed()) report.close();
 else throw new [          ]
}


2.
List validFormats = Arrays.asList("HTML", "JAVA", "JSP");
public void reformat(String format)
{
 if(validFormats.contains(format)) applyFormatting(format);
 else throw new [          ]
}
```

NullPointerException(); IllegalStateException();

IllegalArgumentException(); Exception();

Check Answer

20. QID - 2.1254

What will the following code print when run?

```
public class Test{
 static String j = "";
 public static void method( int i){
  try{
   if(i == 2){
     throw new Exception();
   }
   j += "1";
  }
  catch (Exception e){
   j += "2";
   return;
  }
  finally{
   j += "3";
  }
  j += "4";
 }
 public static void main(String args[]){
  method(1);
  method(2);
  System.out.println(j);
 }
}
```

Select 1 option

A. 13432

B. 13423

C. 14324

D. 12434

E. 12342

Check Answer

21. QID - 2.1094

What will the following program print when run?

```
public class TestClass{
  public static void main(String[] args){
     try{
        System.exit(0);
     }
     finally{
         System.out.println("finally is always executed!");
     }
  }
}
```

Select 1 option

A. It will print "finally is always executed!"

B. It will not compile as there is no catch block.

C. It will not print anything.

D. An exception will be thrown

E. None of the above.

Check Answer

22. QID - 2.895

What two changes can you do, independent of each other, to make the following code compile:

```
//assume appropriate imports
class PortConnector {

    public PortConnector(int port) {
        if (Math.random() > 0.5) {
            throw new IOException();
        }

        throw new RuntimeException();
    }
}


public class TestClass {

    public static void main(String[] args) {
        try {
            PortConnector pc = new PortConnector(10);
        } catch (RuntimeException re) {
            re.printStackTrace();
        }
    }
}
```

Select 2 options

A. add `throws IOException` to the `main` method.

B. add `throws IOException` to `PortConnector` constructor.

C. add `throws IOException` to the `main` method as well as to `PortConnector` constructor.

D. Change `RuntimeException` to `java.io.IOException`.

E. add `throws Exception` to `PortConnector` constructor and change `catch(RuntimeException re)` to `catch(Exception re)` in the main method.

Check Answer

23. QID - 2.1172

Considering the following program, which of the options are true?

```
public class FinallyTest{
   public static void main(String args[]){
      try{
          if (args.length == 0) return;
          else throw new Exception("Some Exception");
      }
      catch(Exception e){
          System.out.println("Exception in Main");
      }
      finally{
          System.out.println("The end");
      }
   }
}
```

Select 2 options

A. If run with no arguments, the program will only print 'The end'.

B. If run with one argument, the program will only print 'The end'.

C. If run with one argument, the program will print 'Exception in Main' and 'The end'.

D. If run with one argument, the program will only print 'Exception in Main'.

E. Only one of the above is correct.

Check Answer

24. QID - 2.1223

Consider the following hierarchy of Exception classes :

```
java.lang.RuntimeException
  +-------- IndexOutOfBoundsException
                  +---------ArrayIndexOutOfBoundsException,
StringIndexOutOfBoundsException
```

Which of the following statements are correct for a method that can throw `ArrayIndexOutOfBounds` as well as `StringIndexOutOfBounds` Exceptions but does not have try catch blocks to catch the same?

Select 3 options

A. The method calling this method will either have to catch these 2 exceptions or declare them in its `throws` clause.

B. It is ok if it declares just `throws ArrayIndexOutOfBoundsException`

C. It must declare `throws ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException`

D. It is ok if it declares just `throws IndexOutOfBoundsException`

E. It does not need to declare any throws clause.

Check Answer

25. QID - 2.1368

Identify the exceptions that are usually thrown by the JVM and the exceptions usually thrown by an application.

Select 1 option

A. JVM : IllegalStateException, IllegalArgumentException
Application : ClassCastException, NullPointerException, SecurityException

B. JVM : IllegalStateException, IllegalArgumentException, ClassCastException,
Application : NullPointerException, SecurityException

C. JVM : IllegalStateException, IllegalArgumentException, ClassCastException, NullPointerException
Application : SecurityException

D. JVM : ClassCastException, NullPointerException, SecurityException
Application : IllegalStateException, IllegalArgumentException

E. JVM : ClassCastException, NullPointerException
Application : IllegalStateException, IllegalArgumentException, SecurityException

F. JVM : ClassCastException, NullPointerException, IllegalStateException
Application : IllegalArgumentException, SecurityException

Check Answer

26. QID - 2.866

What can be the type of a `catch` argument ?

Select 1 option

A. Any class that extends `java.lang.Exception`

B. Any class that extends `java.lang.Exception` except any class that extends `java.lang.RuntimeException`

C. Any class that is-a `Throwable`.

D. Any Object

E. Any class that extends `Error`

Check Answer

27. QID - 2.1276

What is wrong with the following code?

```
class MyException extends Exception {}
public class TestClass{
   public static void main(String[] args){
      TestClass tc = new TestClass();
      try{
         tc.m1();
      }
      catch (MyException e){
         tc.m1();
      }
      finally{
         tc.m2();
      }
   }
   public void m1() throws MyException{
      throw new MyException();
   }
   public void m2() throws RuntimeException{
      throw new NullPointerException();
   }
}
```

Select 1 option

A. It will not compile because you cannot throw an exception in finally block.

B. It will not compile because you cannot throw an exception in catch block.

C. It will not compile because NullPointerException cannot be created this way.

D. It will not compile because of unhandled exception.

E. It will compile but will throw an exception when run.

Check Answer

28. QID - 2.1031

Identify the exceptions that will be received when the code snippets on the left hand side are executed.

1.
```
String s = null;
System.out.println(s.length());
```

2.
```
int[] ia = new int[]{ 1, 2, 3};
System.out.println(ia[3]);
```

3.
```
Class.forName("java.lang.String");
```

4.
```
public class X {
static {
throw new NullPointerException();
}
}
```

ArrayIndexOutOfBoundsException

ExceptionInInitializerError

ClassNotFoundException

NullPointerException

Will Not Compile.

No Exception Will Be Thrown.

Check Answer

29. QID - 2.1025

Identify the correct constructs.

Select 1 option

A.

```
try {
  for( ;; );
}finally { }
```

B.

```
try {
  File f = new File("c:\a.txt");
} catch {  f = null; }
```

C.

```
int k = 0;
try {
  k = callValidMethod();
}
System.out.println(k);
catch {  k = -1; }
```

D.

```
try {
  try {
     Socket s = new ServerSocket(3030);
  }catch(Exception e) {
```

```
    s = new ServerSocket(4040);
  }
}
```

E.

```
try {
       s = new ServerSocket(3030);
}
catch(Exception t){ t.printStackTrace(); }
catch(IOException e) {
    s = new ServerSocket(4040);
}
catch(Throwable t){ t.printStackTrace();  }
```

F.

```
int x = validMethod();
try {
  if(x == 5) throw new IOException();
  else if(x == 6) throw new Exception();
}finally {
 x = 8;
}
catch(Exception e){ x = 9; }
```

Check Answer

30. QID - 2.1210

Consider the following code snippet:

```
void m1() throws Exception{
   try{
      // line1
   }
   catch (IOException e){
       throw new SQLException();
   }
   catch(SQLException e){
       throw new InstantiationException();
   }
   finally{
      throw new CloneNotSupportedException();   // this is not a Runt
   }
}
```

Which of the following statements are true?

Select 2 options

A. If IOException gets thrown at line1, then the whole method will end up throwing SQLException.

B. If IOException gets thrown at line1, then the whole method will end up throwing CloneNotSupportedException.

C. If IOException gets thrown at line1, then the whole method will end up throwing InstantiationException()

D. If no exception is thrown at line1, then the whole method will end up throwing

CloneNotSupportedException.

E. If SQLException gets thrown at line1, then the whole method will end up throwing InstantiationException()

Check Answer

31. QID - 2.842

A new Java programmer has written the following method that takes an array of integers and sums up all the integers that are less than 100.

```
public void processArray(int[] values){
        int sum = 0;
        int i = 0;
        try{
            while(values[i]<100){
                sum = sum +values[i];
                i++;
            }
        }
        catch(Exception e){ }
        System.out.println("sum = "+sum);
    }
```

Which of the following are best practices to improve this code?

Select 2 options

A. Use `ArrayIndexOutOfBoundsException` for the catch argument.

B. Use `ArrayIndexOutOfBoundsException` for the catch argument and add code in the catch block to log or print the exception.

C. Add code in the catch block to handle the exception.

D. Use flow control to terminate the loop.

Check Answer

32. QID - 2.1348

Which digits and in what order will be printed when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int k = 0;
      try{
         int i = 5/k;
      }
      catch (ArithmeticException e){
         System.out.println("1");
      }
      catch (RuntimeException e){
         System.out.println("2");
         return ;
      }
      catch (Exception e){
         System.out.println("3");
      }
      finally{
         System.out.println("4");
      }
      System.out.println("5");
   }
}
```

Select 1 option

A. The program will print 5.

B. The program will print 1 and 4, in that order.

C. The program will print 1, 2 and 4, in that order.

D. The program will print 1, 4 and 5, in that order.

E. The program will print 1,2, 4 and 5, in that order.

Check Answer

33. QID - 2.954

What class of objects can be declared by the throws clause?

Select 3 options

A. Exception

B. Error

C. Event

D. Object

E. RuntimeException

Check Answer

34. QID - 2.959

What will the following class print ?

```
class Test{
   public static void main(String[] args){
      int[][] a = { { 00, 01 }, { 10, 11 } };
      int i = 99;
      try {
         a[val()][i = 1]++;
      } catch (Exception e) {
         System.out.println( i+", "+a[1][1]);
      }
   }
   static int val() throws Exception {
     throw new Exception("unimplemented");
   }
}
```

Select 1 option

A. 99 , 11

B. 1 , 11

C. 1 and an unknown value.

D. 99 and an unknown value.

E. It will throw an exception at Run time.

Check Answer

35. QID - 2.1211

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      Exception e = null;
      throw e;
   }
}
```

Select 1 option

A. The code will fail to compile.

B. The program will fail to compile, since it cannot throw a `null`.

C. The program will compile without error and will throw an `Exception` when run.

D. The program will compile without error and will throw `java.lang.NullPointerException` when run

E. The program will compile without error and will run and terminate without any output.

Check Answer

36. QID - 2.1236

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     int x = 1;
     int y = 0;
     if( x/y ) System.out.println("Good");
     else  System.out.println("Bad");
  }
}
```

Select 1 option

A. Good

B. Bad

C. Exception at runtime saying division by Zero.

D. It will not compile.

E. None of the above.

Check Answer

37. QID - 2.979

What will be the output when the following code is compiled and run?

```
//in file Test.java
class E1 extends Exception{ }
class E2 extends E1 { }
class Test{
   public static void main(String[] args){
      try{
         throw new E2();
      }
      catch(E1 e){
         System.out.println("E1");
      }
      catch(Exception e){
         System.out.println("E");
      }
      finally{
         System.out.println("Finally");
      }
   }
}
```

Select 1 option

A. It will not compile.

B. It will print E1 and Finally.

C. It will print E1, E and Finally.

D. It will print E and Finally.

E. It will print Finally.

Check Answer

38. QID - 2.1017

Identify the exceptions that are usually thrown by the JVM and the exceptions usually thrown by an application.

1. IllegalStateException
2. IllegalArgumentException
3. ClassCastException
4. NullPointerException

JVM ApplicationProgrammer

Check Answer

39. QID - 2.1030

Identify the exceptions that SHOULD be thrown in the situations shown below.

(Assume that CLUBS, DIAMONDS, HEARTS, SPADES are valid elements of an enum.)

```
1.
switch(suit) {
 case CLUBS:
 ...
 break;
 case DIAMONDS:
  ...
 break;
 case HEARTS:
 ...
 break;
 case SPADES:
 ...
 break;
 default : throw new [          ]
}

2.
public void applyCode(String code)
{
 if(code.startsWith("XA")) apply(code.substring(2));
 else throw new [          ]
}
```

AssertionError(); IllegalStateException(); IllegalArgumentException(); Exception();

Check Answer

40. QID - 2.1046

What will be the output of the following program:

```
public class TestClass{
   public static void main(String args[]){
      try{
         m1();
      }catch(IndexOutOfBoundsException e){
         System.out.println("1");
         throw new NullPointerException();
      }catch(NullPointerException e){
         System.out.println("2");
         return;
      }catch (Exception e) {
         System.out.println("3");
      }finally{
         System.out.println("4");
      }
      System.out.println("END");
   }
   // IndexOutOfBoundsException is a subclass of RuntimeException.
   static void m1(){
      System.out.println("m1 Starts");
      throw new IndexOutOfBoundsException( "Big Bang " );
   }
}
```

Select 3 options

A. The program will print `m1 Starts`.

B. The program will print `m1 Starts, 1` and `4`, in that order.

C. The program will print `m1 Starts, 1` and `2`, in that order.

D. The program will print `m1 Starts, 1, 2` and `4` in that order.

E. `END` will not be printed.

Check Answer

41. QID - 2.1093

Which statements regarding the following code are correct ?

```
class Base{
   void method1() throws java.io.IOException, NullPointerException{
      someMethod("arguments");
      // some I/O operations
   }
   int someMethod(String str){
      if(str == null) throw new NullPointerException();
      else return str.length();
   }
}
public class NewBase extends Base{
      void method1(){
           someMethod("args");
      }
}
```

Select 2 options

A. method1 in class NewBase does not need to specify any exceptions.

B. The code will not compile because RuntimeExceptions cannot be given in throws clause.

C. method1 in class NewBase must at least give `IOException` in its throws clause.

D. method1 in class NewBase must at least give `NullpointerException` in its throws clause.

E. There is no problem with the code.

Check Answer

42. QID - 2.1097

What will be the output of the following program?

```
class TestClass{
   public static void main(String[] args) throws Exception{
      try{
         amethod();
         System.out.println("try");
      }
      catch(Exception e){
         System.out.println("catch");
      }
      finally   {
         System.out.println("finally");
      }
      System.out.println("out");
   }
   public static void amethod(){ }
}
```

Select 1 option

A. `try finally`

B. `try finally out`

C. `try out`

D. `catch finally out`

E. It will not compile because `amethod()` does not throw any exception.

Check Answer

43. QID - 2.827

What will be the output when the following program is run?

```
package exceptions;
public class TestClass{
    public static void main(String[] args) {
        try{
            hello();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void hello() throws MyException{
        int[] dear = new int[7];
        dear[0] = 747;
        foo();
    }

    static void foo() throws MyException{
        throw new MyException("Exception from foo");
    }
}

class MyException extends Exception {
    public MyException(String msg){
        super(msg);
    }
}
```

(Assume that line numbers printed in the messages given below are correct.)

Select 1 option

A. `Exception in thread "main"`
`java.lang.ArrayIndexOutOfBoundsException: 10`

```
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

B.
```
Error in thread "main" java.lang.ArrayIndexOutOfBoundsException
```

C.
```
exceptions.MyException: Exception from foo
```

D.
```
exceptions.MyException: Exception from foo
    at exceptions.TestClass.foo(TestClass.java:29)
    at exceptions.TestClass.hello(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

Check Answer

44. QID - 2.1026

Given the class

```
// Filename: Test.java
public class Test{
   public static void main(String args[]){
      for(int i = 0; i< args.length; i++){
         System.out.print("  "+args[i]);
      }
   }
}
```

Now consider the following 3 options for running the program:

```
a: java Test
b: java Test param1
c: java Test param1 param2
```

Which of the following statements are true?

Select 2 options

A. The program will throw `java.lang.ArrayIndexOutofBoundsException` on option a.

B. The program will throw `java.lang.NullPointerException` on option a.

C. The program will print `Test param1` on option b.

D. It will print `param1 param2` on option c.

E. It will not print anything on option a.

Check Answer

45. QID - 2.1323

What is the result of compiling and running this code?

```
class MyException extends Throwable{}
class MyException1 extends MyException{}
class MyException2 extends MyException{}
class MyException3 extends MyException2{}
public class ExceptionTest{
   void myMethod() throws MyException{
      throw new MyException3();
   }
   public static void main(String[] args){
      ExceptionTest et = new ExceptionTest();
      try{
         et.myMethod();
      }
      catch(MyException me){
         System.out.println("MyException thrown");
      }
      catch(MyException3 me3){
         System.out.println("MyException3 thrown");
      }
      finally{
         System.out.println(" Done");
      }
   }
}
```

Select 1 option

A. MyException thrown

B. MyException3 thrown

C. MyException thrown Done

D. MyException3 thrown Done

E. It fails to compile

Check Answer

46. QID - 2.964

What letters, and in what order, will be printed when the following program is compiled and run?

```
public class FinallyTest{
   public static void main(String args[]) throws Exception{
       try{
          m1();
          System.out.println("A");
       }
       finally{
          System.out.println("B");
       }
       System.out.println("C");
   }
   public static void m1() throws Exception { throw new Exception();
}
```

Select 1 option

A. It will print C and B, in that order.

B. It will print A and B, in that order.

C. It will print B and throw Exception.

D. It will print A, B and C in that order.

E. Compile time error.

Check Answer

47. QID - 2.1260

What will be the output of the following class...

```
class Test{
   public static void main(String[] args){
      int j = 1;
      try{
         int i = doIt() / (j = 2);
      } catch (Exception e){
         System.out.println(" j = " + j);
      }
   }
   public static int doIt() throws Exception {  throw new Exception('
}
```

Select 1 option

A. `It will print j = 1;`

B. `It will print j = 2;`

C. The value of j cannot be determined.

D. It will not compile.

E. None of the above.

Check Answer

48. QID - 2.1311

The following class will not throw a `NullPointerException` when compiled and run.

```
class Test{
   public static void main(String[] args) throws Exception{
      int[] a = null;
      int i = a [ m1() ];
   }
   public static int m1() throws Exception{
      throw new Exception("Some Exception");
   }
}
```

Select 1 option

A. True

B. False

Check Answer

49. QID - 2.1255

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      loop :          // 1
      {
         System.out.println("Loop Lable line");
         try{
            for (  ;  true ;  i++ ){
               if( i >5) break loop;        // 2
            }
         }
         catch(Exception e){
            System.out.println("Exception in loop.");
         }
         finally{
            System.out.println("In Finally");       // 3
         }
      }
   }
}
```

Select 1 option

A. Compilation error at line 1 as this is an invalid syntax for defining a label.

B. Compilation error at line 2 as 'loop' is not visible here.

C. No compilation error and line 3 will be executed.

D. No compilation error and line 3 will NOT be executed.

E. Only the line with the label Loop will be printed.

Check Answer

50. QID - 2.984

Following is a supposedly robust method to parse an input for a float :

```
public float parseFloat(String s){
   float f = 0.0f;
   try{
      f = Float.valueOf(s).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      System.out.println("Invalid input " + s);
      f = Float.NaN ;
      return f;
   }
   finally { System.out.println("finally");  }
   return f ;
}
```

Which of the following statements about the above method are true??

Select 1 option

A. If input is `"0.1"` then it will return `0.1` and print `finally`.

B. If input is `"0x.1"` then it will return `Float.Nan` and print `Invalid Input 0x.1` and `finally`.

C. If input is `"1"` then it will return `1.0` and print `finally`.

D. If input is `"0x1"` then it will return `0.0` and print `Invalid Input 0x1` and `finally`.

E. The code will not compile.

Check Answer

51. QID - 2.1048

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      try{
         RuntimeException re = null;
         throw re;
      }
      catch(Exception e){
         System.out.println(e);
      }
   }
}
```

Select 1 option

A. The code will fail to compile, since `RuntimeException` cannot be caught by catching an Exception.

B. The program will fail to compile, since `re` is `null`.

C. The program will compile without error and will print `java.lang.RuntimeException` when run.

D. The program will compile without error and will print `java.lang.NullPointerException` when run.

E. The program will compile without error and will run and print `null`.

Check Answer

52. QID - 2.1023

Identify the exceptions that will be received when the code snippets on the left hand side are executed.

1.
```
int[] ia = new int[]{ 1, 2, 3};
System.out.println(ia[-1]);
```

2.
```
public class X {
static int k = 0;
static {
k = 10/0;
}
}
```

3.
```
SomeClass sc = new SomeClass();
```
(Assume that SomeClass is not available in runtime classpath.)

4.
```
public class X {
static {
if(true) throw new NullPointerException();
}
}
```

NoClassDefFoundError NullPointerException ArrayAccessException

ExceptionInInitializerError ArrayIndexOutOfBoundsException

IllegalArrayAccessException NoSuchClassException

Check Answer

Handling Exceptions (Answered)

01. QID - 2.1305 : Handling Exceptions

Which of these statements are true?

Correct Options are : C D

~~**A.**~~ If a RuntimeException is not caught, the method will terminate and normal execution of the thread will resume.

> Any remaining code of the method will not be executed. Further, any uncaught exception will cause the JVM to kill the thread.

~~**B.**~~ An overriding method must declare that it throws the same exception classes as the method it overrides.

> It can throw any subset of the exceptions thrown by overridden class.

C. The main method of a program can declare that it throws checked exceptions.

> Any method can do that !

D. A method declaring that it throws a certain exception class may throw instances of any subclass of that exception class.

> Note that it cannot throw the instances of any superclasses of the exception.

~~**E.**~~ finally blocks are executed if and only if an exception gets thrown while inside the corresponding try block.

> Finally is ALWAYS executed. (Only exception is System.exit())

Explanation:

Normal execution will not resume if an exception is uncaught by a method. The exception will propagate up the method invocation stack until some method handles it. If no one handles it then the exception will be handled by the JVM and the JVM will terminated that thread.

An overriding method only needs to declare that it can throw a subset of the exceptions the overridden method can throw. Having no throws clause in the overriding method is OK.

Back to Question without Answer

02. QID - 2.1345 : Handling Exceptions

Assume that a method named 'method1' contains code which may raise a non-runtime (checked) Exception.
What is the correct way to declare that method so that it indicates that it expects the caller to handle that exception?

Correct Options are : A D

A. `public void method1() throws Throwable`

~~B.~~ `public void method1() throw Exception`

> Note that it should be 'throws' and not 'throw'

~~C.~~ `public void method1() throw new Exception`

> This is not the right syntax.

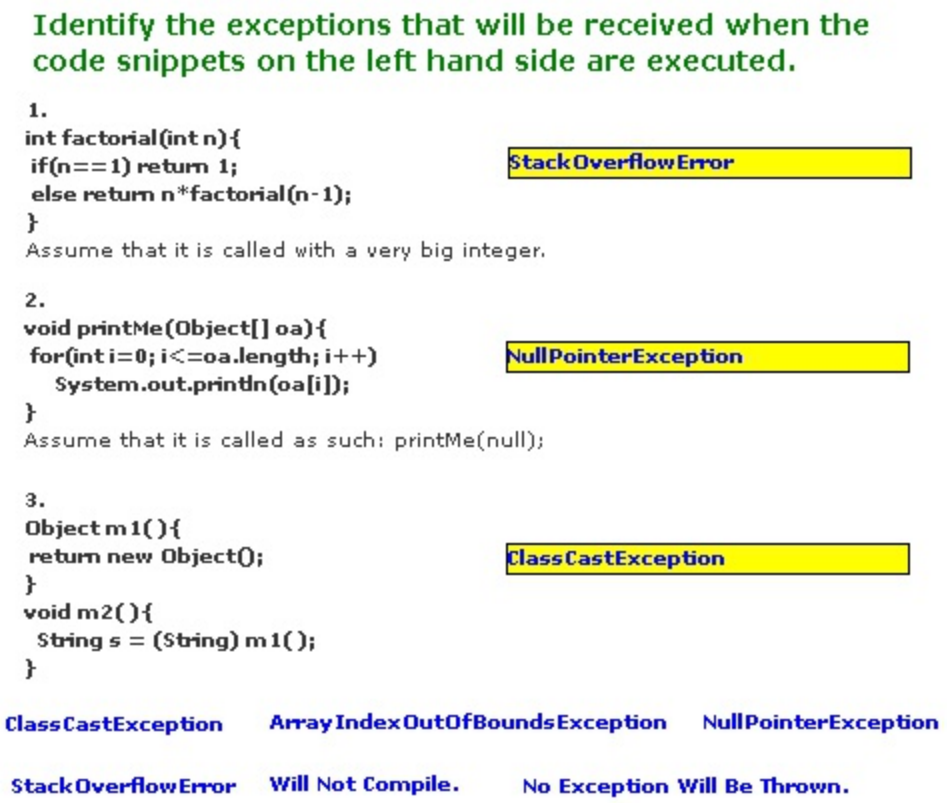
D. `public void method1() throws Exception`

~~E.~~ `public void method1()`

> Non runtime exception must be declared in the throws clause.

Back to Question without Answer

03. QID - 2.1034 : Handling Exceptions



Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

04. QID - 2.1133 : Handling Exceptions

Which of the following can be thrown using a throw statement?

Correct Options are : C D E

~~**A.**~~ Event

~~**B.**~~ Object

C. Throwable

D. Exception

E. RuntimeException

Explanation:

You can only throw a Throwable using a throws clause. Exception and Error are two main subclasses of Throwable.

Back to Question without Answer

05. QID - 2.1006 : Handling Exceptions

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}
class AnotherException extends Exception {}
public class ExceptionTest{
   public static void main(String [] args) throws Exception{
      try{
         m2();
      }
      finally{ m3(); }
    }
    public static void m2() throws NewException{  throw new NewExcep
    public static void m3() throws AnotherException{  throw new Anot
}
```

Correct Option is : A

A. It will compile but will throw `AnotherException` when run.

~~**B.**~~ It will compile but will throw `NewException` when run.

~~**C.**~~ It will compile and run without throwing any exceptions.

~~**D.**~~ It will not compile.

~~**E.**~~ None of the above.

Explanation:

`m2()` throws `NewException`, which is not caught anywhere. But before exiting out of the main method, finally must be executed. Since finally throw `AnotherException` (due to a call to `m3()`), the `NewException` thrown in the try block (due to call to `m2()`) is ignored and `AnotherException` is thrown from the main method.

Back to Question without Answer

06. QID - 2.1112 : Handling Exceptions

What will be the result of compiling and running the following program ?

```
class NewException extends Exception {}

class AnotherException extends Exception {}

public class ExceptionTest{
    public static void main(String[] args) throws Exception{
        try{
            m2();
        }
        finally{
            m3();
        }
        catch (NewException e){}
    }

    public static void m2() throws NewException { throw new NewExcep

    public static void m3() throws AnotherException{ throw new Anothe

}
```

Correct Option is : D

~~**A.**~~ It will compile but will throw `AnotherException` when run.

~~**B.**~~ It will compile but will throw `NewException` when run.

~~**C.**~~ It will compile and run without throwing any exceptions.

D. It will not compile.

> Because a catch block cannot follow a finally block!

~~**E.**~~ None of the above.

Explanation:

Syntax of try/catch/finally is:

```
try{
}
catch(Exception1 e) {... }
catch(Exception2 e) {... }
...
catch(ExceptionN e) {... }
finally { ...  }
```

With a try, either a catch and or finally or both can occur.
A try MUST be followed by at least one catch or finally. (Unless it is a try with resources statement, which is not in scope for this exam.)

In Java 7, you can collapse the catch blocks into a single one:

```
try {
  ...
}
catch (SQLException | IOException | RuntimeException e) {
  //In this block, the class of the actual exception object will be
whatever exception is thrown at runtime.
  //But the class of the reference e will be the closest common
super class of all the exceptions in the catch block.
  //In this case, it will be java.lang.Exception because that is the
most specific class that is a super class for all the three
exceptions.
  e.printStackTrace();
}
```

Back to Question without Answer

07. QID - 2.880 : Handling Exceptions

Which of the following are standard Java exception classes?

Correct Options are : A E

A. `FileNotFoundException`

~~B.~~ `InputException`

> There is an `java.io.IOException` but no InputException or OutputException.

~~C.~~ `CPUError`

~~D.~~ `MemoryException`

> There is an OutOfMemoryError but no MemoryException. There is also a StackOverflowError.

E. `SecurityException`

> Java has a `java.lang.SecurityException`. This exception extends `RuntimeException` and is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception.

Back to Question without Answer

08. QID - 2.826 : Handling Exceptions

What will be the output when the following program is run?

```
package exceptions;
public class TestClass {
    public static void main(String[] args) {
        try{
            doTest();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void doTest() throws MyException{
        int[] array = new int[10];
        array[10] = 1000;
        doAnotherTest();
    }

    static void doAnotherTest() throws MyException{
        throw new MyException("Exception from doAnotherTest");
    }
}
class MyException extends Exception {
    public MyException(String msg){
     super(msg);
    }
}
```

Correct Option is : A

A.
```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 10
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

You are creating an array of length 10. Since array numbering starts with 0, the last element would be `array[9]`.
`array[10]` would be outside the range of the array and therefore an `ArrayIndexOutOfBoundsException` will be thrown, which cannot be caught by `catch(MyException )` clause.
The exception is thus thrown out of the main method and is handled by the JVM's uncaught exception handling mechanism, which prints the stack trace.

B. ~~`Error in thread "main" java.lang.ArrayIndexOutOfBoundsException`~~

java.lang.ArrayIndexOutOfBoundsException extends java.lang.RuntimeException, which in turn extends java.lang.Exception. Therefore, ArrayIndexOutOfBoundsException is an Exception and not an Error.

C. ~~`exceptions.MyException: Exception from doAnotherTest`~~

D. ~~`exceptions.MyException: Exception from doAnotherTest`~~

```
    at exceptions.TestClass.doAnotherTest(TestClass.java:29)
    at exceptions.TestClass.doTest(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

Explanation:

Note that there are questions in the exam that test your knowledge about how exception messages are printed.
When you use `System.out.println(exception)`, a stack trace is not printed. Just the name of the exception class and the message is printed.
When you use `exception.printStackTrace()`, a complete chain of the names of the methods called, along with the line numbers, is printed from the point where the exception was thrown and up to the point where the exception was caught.

Back to Question without Answer

09. QID - 2.967 : Handling Exceptions

What will the following code print when compiled and run?

```
abstract class Calculator{
   abstract void calculate();
   public static void main(String[] args){
      System.out.println("calculating");
      Calculator x = null;
      x.calculate();
   }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

> It will compile fine.

~~**B.**~~ It will not print anything and will throw `NullPointerException`

C. It will print `calculating` and then throw `NullPointerException`.

> After printing, when it tries to call `calculate()` on `x`, it will throw `NullPointerException` since `x` is `null`.

~~**D.**~~ It will print `calculating` and will throw `NoSuchMethodError`

~~**E.**~~ It will print `calculating` and will throw `MethodNotImplementedException`

Back to Question without Answer

10. QID - 2.1235 : Handling Exceptions

A Java programmer is developing a desktop application. Which of the following exceptions would be appropriate for him to throw explicitly from his code?

Correct Option is : D

~~**A.**~~ `NullPointerException`

~~**B.**~~ `ClassCastException`

~~**C.**~~ `ArrayIndexOutofBoundsException`

D. `Exception`

~~**E.**~~ `NoClassDefFoundError`

> `NoClassDefFoundError` is thrown by the JVM when it attempts to load a class and is unable to find the class file.
> Note that it extends `java.lang.Error` and Errors are always thrown by the JVM. A programmer should never throw an Error explicitly.

Explanation:

Observe that all the exceptions given in the options other than `Exception` and `NoClassDefFoundError` are `RuntimeException`s. These are usually thrown implicitly. A programmer should not throw these exceptions explicitly. `java.lang.Exception` and its subclasses (except `RuntimeException`) should be used by the programmer to reflect known exceptional situations, while RuntimeExceptions are used to reflect unforseen or unrecoverable exceptional

situations.

Note: There is no hard and fast rule that says `RuntimeException`s (such as the ones mentioned in this questions) must not be thrown explicitly. It is ok to throw these exceptions explicitly in certain situations. For example, framework/library classes such as Struts, Spring, and Hibernate, and standard JDK classes throw these exceptions explicitly. But for the purpose of the exam, it is a good way to determine if a given application should be thrown expliclity by the programmer or not.

Back to Question without Answer

11. QID - 2.1148 : Handling Exceptions

What will the following code print?

```
public class Test{
   public int luckyNumber(int seed){
      if(seed > 10) return seed%10;
         int x = 0;
            try{
               if(seed%2 == 0) throw new Exception("No Even no.");
               else return x;
            }
            catch(Exception e){
               return 3;
            }
            finally{
               return 7;
            }
         }

        public static void main(String args[]){
           int amount = 100, seed = 6;
           switch( new Test().luckyNumber(6) ){
               case 3: amount = amount * 2;
               case 7: amount = amount * 2;
               case 6: amount = amount + amount;
               default :
           }
          System.out.println(amount);
       }
}
```

Correct Option is : E

~~A.~~ It will not compile.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ 800

~~**D.**~~ 200

E. 400

Explanation:

When you pass 6 to `luckyNumber()`, `if(seed%2 == 0) throw new Exception("No Even no.");` is executed and the exception is caught by the catch block where it tries to `return 3;` But as there is a finally associated with the try/catch block, it is executed before anything is returned. Now, as finally has `return 7;`, this value supersedes 3.
In fact, this method will always return 7 if seed <= 10.

Now, in the switch there is no break statement. So both -
`case 7: amount = amount * 2;`
and
`case 6: amount = amount + amount;`
are executed. so the final amount becomes 400.

Back to Question without Answer

12. QID - 2.1347 : Handling Exceptions

Consider the following code...

```
public class TestClass{
     class MyException extends Exception {}
     public void myMethod() throws XXXX{
         throw new MyException();
     }
}
```

What can replace XXXX?

Correct Options are : A B D

A. MyException

B. Exception

> Because Exception is a superclass of MyException.

~~**C.**~~ No throws clause is necessary

> It is needed because MyException is a checked exception. Any exception that extends java.lang.Exception but is not a subclass of java.lang.RuntimeException is a checked exception.

D. Throwable

> Because Throwable is a super class of Exception.

~~**E.**~~ RuntimeException

Explanation:

You can use `Throwable` as well as `Exception` as both of them are super classes of `MyException`.
`RuntimeException` (and its subclasses such as `NullPointerException` and `ArrayIndexOutOfBoundsException`) is not a checked exception. So it cannot cover for `MyException` which is a checked exception.
You cannot use `Error` as well because it is not in the hierarchy of `MyException`, which is Object <- Throwable <- Exception <- MyException.

Back to Question without Answer

13. QID - 2.1033 : Handling Exceptions

A try statement must always have a ..... associated with it.

Correct Option is : D

~~**A.** catch~~

~~**B.** throws~~

~~**C.** finally~~

D. catch, finally or both

~~**E.** throw~~

Explanation:

A try without resources must have either a catch or a finally. It may have both as well. Thus, the following constructs are valid:

1.

```
try{
}
catch(Exception e){  }          // no finally
```

2.

```
try{
}
finally{  }          // no catch
```

3.

```
try{
}
catch(Exception e){  }
finally{  }
```

4. A catch can catch multiple exceptions:

```
try{
}
catch(Exception1|Exception2|Exception3 e){  }
```

Note: try with resources (which is not on this exam) may omit catch as well as finally blocks.

Back to Question without Answer

14. QID - 2.1005 : Handling Exceptions

Consider the following code:

```
class A {
    public void doA(int k) throws Exception {  // 0
        for(int i=0; i< 10; i++) {
            if(i == k) throw new Exception("Index of k is "+i); // 1
        }
    }
    public void doB(boolean f) { // 2
        if(f) {
            doA(15); // 3
        }
        else return;
    }
    public static void main(String[] args) { // 4
        A a = new A();
        a.doB(args.length>0); // 5
    }
 }
```

Which of the following statements are correct?

Correct Option is : D

~~**A.**~~ This will compile and run without any errors or exception.

~~**B.**~~ This will compile if `throws Exception` is added at line //2

~~**C.**~~ This will compile if `throws Exception` is added at line //4

D. This will compile if `throws Exception` is added at line //2 as well as //4

~~E.~~ This will compile if line marked // 1 is enclosed in a try - catch block.

> Even if // 1 is enclosed in a try block, the method still has `throws Exception` in its declaration, which will force the caller of this method to either declare `Exception` in its throws clause or put the call within a try block.

Explanation:

Any checked exceptions must be either handled using a try block or the method that generates the exception must declare that it throws that exception.
In this case, doA() declares that it throws `Exception`. `doB()` is calling `doA` but it is not handling the exception generated by `doA()`. So, it must declare that it throws `Exception`. Now, the `main()` method is calling `doB()`, which generates an exception (due to a call to `doA()`). Therefore, `main()` must also either wrap the call to `doB()` in a try block or declare it in its `throws` clause.

The `main(String[] args)` method is the last point in your program where any unhandled checked exception can bubble up to. After that the exception is thrown to the JVM and the JVM kills the thread.

Back to Question without Answer

15. QID - 2.1350 : Handling Exceptions

What will the following code snippet print:

```
Float f = null;
try{
    f = Float.valueOf("12.3");
    String s = f.toString();
    int i = Integer.parseInt(s);
    System.out.println("i = "+i);
}
catch(Exception e){
    System.out.println("trouble : "+f);
}
```

Correct Option is : D

~~**A.**~~ 12

~~**B.**~~ 13

~~**C.**~~ trouble : null

D. trouble : 12.3

~~**E.**~~ trouble : 0.0

Explanation:

`f = Float.valueOf("12.3");` executes without any problem.

`int i = Integer.parseInt(s);` throws a NumberFormatException because 12.3 is not an integer.

Thus, the catch block prints trouble : 12.3

Back to Question without Answer

16. QID - 2.1358 : Handling Exceptions

What will the following program print when run using the command line: `java TestClass`

```
public class TestClass {

   public static void methodX() throws Exception {
      throw new AssertionError();
   }

   public static void main(String[] args) {
      try{
         methodX();
      }
      catch(Exception e) {
        System.out.println("EXCEPTION");
      }
   }
}
```

Correct Option is : A

A. It will throw `AssertionError` out of the main method.

> A subclass of `Error` cannot be caught using a catch block for `Exception` because `java.lang.Error` does not extend `java.lang.Exception`.

~~**B.**~~ It will print `EXCEPTION`.

> The catch block will not be able to catch the Error thrown by `methodX()`.

~~**C.**~~ It will not compile because of the throws clause in `methodX()`.

> The `throws` clause is valid even though unnecessary in this case.

~~**D.**~~ It will end without printing anything because assertions are disabled by default.

> It is true that assertions are disabled by default however, `methodX` is throwing an `AssertionError` explicitly like any other Throwable. Here, the assertion mechanism is not even used.

Back to Question without Answer

17. QID - 2.1301 : Handling Exceptions

What is wrong with the following code written in a single file named TestClass.java?

```
class SomeThrowable extends Throwable { }
class MyThrowable extends SomeThrowable { }
public class TestClass{
   public static void main(String args[]) throws SomeThrowable{
      try{
         m1();
      }catch(SomeThrowable e){
         throw e;
      }finally{
         System.out.println("Done");
      }
   }
   public static void m1() throws MyThrowable{
      throw new MyThrowable();
   }
}
```

Correct Options are : D E

~~**A.**~~ The main declares that it throws `SomeThrowable` but throws `MyThrowable`.

> That's OK. You can put a Super class in the throws clause and then you can throw any subclass exception.

~~**B.**~~ You cannot have more than 2 classes in one file.

> You sure can. The only limitation is you can have only one top level public class in a file.

~~**C.**~~ The catch block in the main method must declare that it catches `MyThrowable` rather than `SomeThrowable`.

You can catch a subclass exception in the catch clause that catches a super class.

D. There is nothing wrong with the code.

E. `Done` will be printed.

`Done` will be followed by an exception. Finally is always executed (Only exception is `System.exit();`)

Back to Question without Answer

18. QID - 2.950 : Handling Exceptions

Consider the following method -

```
public float parseFloat( String s ){
   float f = 0.0f;
   try{
      f = Float.valueOf( s ).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      f = Float.NaN ;
      return f;
   }
   finally{
      f = 10.0f;
      return f;
   }
}
```

What will it return if the method is called with the input "0.0" ?

Correct Option is : B

~~**A.**~~ It will not compile.

B. It will return 10.0

~~**C.**~~ It will return Float.Nan

~~**D.**~~ It will return 0.0

~~**E.**~~ None of the above.

Explanation:

finally block will always execute (except when there is a System.exit() in try or catch). And inside the finally block, it is setting f to 10.0. So no matter what the input is, this method will always return 10.0.

Back to Question without Answer

19. QID - 2.1028 : Handling Exceptions

Identify the exceptions that SHOULD be thrown in the situations shown below.

```
1.
public void closeReportManager(ReportManager rep)
{
 if(!rep.isClosed()) report.close();
 else throw new IllegalStateException();
}


2.
List validFormats = Arrays.asList("HTML", "JAVA", "JSP");
public void reformat(String format)
{
 if(validFormats.contains(format)) applyFormatting(format);
 else throw new IllegalArgumentException();
}
```

NullPointerException(); IllegalStateException();

IllegalArgumentException(); Exception();

Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

20. QID - 2.1254 : Handling Exceptions

What will the following code print when run?

```
public class Test{
 static String j = "";
 public static void method( int i){
  try{
   if(i == 2){
     throw new Exception();
   }
   j += "1";
  }
  catch (Exception e){
   j += "2";
   return;
  }
  finally{
   j += "3";
  }
  j += "4";
 }
 public static void main(String args[]){
  method(1);
  method(2);
  System.out.println(j);
 }
}
```

Correct Option is : B

~~**A.**~~ 13432

B. 13423

~~**C.**~~ 14324

~~**D.**~~ 12434

~~**E.**~~ 12342

Explanation:

Try to follow the flow of control :
1. in method(1) : i is not 2 so, j gets "1" then finally is executed which makes j = "13" and then the last statement (j +=4) is executed which makes j = "134".
2. in method(2) : i is 2, so it goes in the if block which throws an exception. So none of the statements of try block are executed and control goes to catch which makes j = "1342", then finally makes j = "13423" and the control is returned. Note that the last statement (j+=4) is not executed as there was an exception thrown in the try block, which cause the control to go to the catch block, which in turn has a return.

Back to Question without Answer

21. QID - 2.1094 : Handling Exceptions

What will the following program print when run?

```
public class TestClass{
  public static void main(String[] args){
     try{
        System.exit(0);
     }
     finally{
         System.out.println("finally is always executed!");
     }
  }
}
```

Correct Option is : C

~~**A.**~~ It will print "finally is always executed!"

~~**B.**~~ It will not compile as there is no catch block.

C. It will not print anything.

~~**D.**~~ An exception will be thrown

~~**E.**~~ None of the above.

Explanation:

finally is always executed (even if you throw an exception in try or catch) but this is the exception to the rule.

When you call System.exit(...); The JVM exits so there is no way to execute the finally block.

Back to Question without Answer

22. QID - 2.895 : Handling Exceptions

What two changes can you do, independent of each other, to make the following code compile:

```
//assume appropriate imports
class PortConnector {

    public PortConnector(int port) {
        if (Math.random() > 0.5) {
            throw new IOException();
        }

        throw new RuntimeException();
    }
}


public class TestClass {

    public static void main(String[] args) {
        try {
            PortConnector pc = new PortConnector(10);
        } catch (RuntimeException re) {
            re.printStackTrace();
        }
    }
}
```

Correct Options are : C E

~~**A.**~~ add `throws IOException` to the `main` method.

~~**B.**~~ add `throws IOException` to `PortConnector` constructor.

C. add `throws IOException` to the `main` method as well as to `PortConnector` constructor.

~~**D.**~~ Change `RuntimeException` to `java.io.IOException`.

E. add `throws Exception` to `PortConnector` constructor and change `catch(RuntimeException re)` to `catch(Exception re)` in the main method.

Explanation:

`IOException` is a checked exception and since the `PortConnector` constructor throws `IOException`, this exception (or its superclass) must be present in the throws clause of the constructor.

Now, the `main` method has two options, either catch `IOException` (or whatever exception PortConnector throws) in its catch block (i.e. option 5) or put that exception in its throws clause (i.e. option 3).

Back to Question without Answer

23. QID - 2.1172 : Handling Exceptions

Considering the following program, which of the options are true?

```
public class FinallyTest{
   public static void main(String args[]){
      try{
          if (args.length == 0) return;
          else throw new Exception("Some Exception");
      }
      catch(Exception e){
          System.out.println("Exception in Main");
      }
      finally{
          System.out.println("The end");
      }
   }
}
```

Correct Options are : A C

A. If run with no arguments, the program will only print 'The end'.

~~**B.**~~ If run with one argument, the program will only print 'The end'.

C. If run with one argument, the program will print 'Exception in Main' and 'The end'.

~~**D.**~~ If run with one argument, the program will only print 'Exception in Main'.

~~**E.**~~ Only one of the above is correct.

Explanation:

There are two points to understand here:
1. Even if the program is executed without any arguments, the 'args' is NOT NULL. In such case it will be initialized to an array of Strings containing zero elements.
2. The finally block is always executed, no matter how control leaves the try block. Only if, in a try or catch block, System.exit() is called then finally will not be executed.

Back to Question without Answer

24. QID - 2.1223 : Handling Exceptions

Consider the following hierarchy of Exception classes :

```
java.lang.RuntimeException
  +-------- IndexOutOfBoundsException
                  +---------ArrayIndexOutOfBoundsException,
StringIndexOutOfBoundsException
```

Which of the following statements are correct for a method that can throw `ArrayIndexOutOfBounds` as well as `StringIndexOutOfBounds` Exceptions but does not have try catch blocks to catch the same?

Correct Options are : B D E

~~**A.**~~ The method calling this method will either have to catch these 2 exceptions or declare them in its `throws` clause.

B. It is ok if it declares just `throws ArrayIndexOutOfBoundsException`

~~**C.**~~ It must declare `throws ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException`

D. It is ok if it declares just `throws IndexOutOfBoundsException`

E. It does not need to declare any throws clause.

Explanation:

Note that both the exceptions are `RuntimeExceptions` so there is no need to catch

these. But it is ok even if the method declares them explicitly.

Back to Question without Answer

25. QID - 2.1368 : Handling Exceptions

Identify the exceptions that are usually thrown by the JVM and the exceptions usually thrown by an application.

Correct Option is : D

~~A.~~ JVM : IllegalStateException, IllegalArgumentException
Application : ClassCastException, NullPointerException, SecurityException

~~B.~~ JVM : IllegalStateException, IllegalArgumentException, ClassCastException,
Application : NullPointerException, SecurityException

~~C.~~ JVM : IllegalStateException, IllegalArgumentException, ClassCastException, NullPointerException
Application : SecurityException

D. JVM : ClassCastException, NullPointerException, SecurityException
Application : IllegalStateException, IllegalArgumentException

~~E.~~ JVM : ClassCastException, NullPointerException
Application : IllegalStateException, IllegalArgumentException, SecurityException

~~F.~~ JVM : ClassCastException, NullPointerException, IllegalStateException
Application : IllegalArgumentException, SecurityException

Explanation:

Note: The terminology "thrown by the JVM" and "thrown programatically or by the

application" is not precise but is used by popular books. If it helps, you can think of the exception categories as "thrown implicitly" and "thrown explicitly". An exception that is thrown even when there is no `throw` statement, is said to be thrown implicitly. For example, calling a method on null will cause a `NullPointerException` to be thrown automatically, even though there is no `throw` statement. On the other hand, a code may throw an exception explicitly by using the `throw` statement. For example, a method code might check an argument for validity and if it finds the argument inappropriate, it may throw an exception by executing `throw new IllegalArgumentException();`.

A quick way to determine who should throw an exception is to see if the exception extends `java.lang.Error`. Errors are always thrown only by the JVM.

Generally, `RuntimeException`s are also thrown by the JVM. However, it is ok for an application code to throw a `RuntimeException` if it makes sense for the application to throw a `RuntimeException` in a given situation.

You should know about the following common exception classes:

`IndexOutOfBoundsException extends RuntimeException`:
Usually thrown by the JVM. Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range. Applications can subclass this class to indicate similar exceptions.
`ArrayIndexOutOfBoundsException and StringIndexOutOfBoundsException` both extend `IndexOutOfBoundsException`.

`IllegalArgumentException` extends `RuntimeException`: If a parameter passed to a method is not valid. Usually thrown by the application.

`IllegalStateException extends RuntimeException`: Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation. Usually thrown by the application.

`ClassCastException extends RuntimeException`: Usually thrown by the JVM. Thrown to indicate that the code has attempted to cast an object to a subclass of which it is not an instance. For example, the following code generates a `ClassCastException`:

```
Object x = new Integer(0);
System.out.println((String)x);
```

`NullPointerException extends RuntimeException`: Usually thrown by the JVM. Thrown when an application attempts to use `null` in a case where an object is required. These include:

- Calling the instance method of a null object.
- Accessing or modifying the field of a null object.
- Taking the length of null as if it were an array.
- Accessing or modifying the slots of null as if it were an array.
- Throwing null as if it were a Throwable value.

Applications should throw instances of this class to indicate other illegal uses of the null object.

`SecurityException extends RuntimeException`: Usually thrown by the JVM. It is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception.

Back to Question without Answer

26. QID - 2.866 : Handling Exceptions

What can be the type of a `catch` argument ?

Correct Option is : C

~~**A.**~~ Any class that extends `java.lang.Exception`

~~**B.**~~ Any class that extends `java.lang.Exception` except any class that extends `java.lang.RuntimeException`

C. Any class that is-a `Throwable`.

> The catch argument type declares the type of exception that the handler can handle and must be the name of a class that extends `Throwable` or `Throwable` itself.

~~**D.**~~ Any Object

~~**E.**~~ Any class that extends `Error`

Explanation:

You must remember the hierarchy of exception classes:

The base class of all exceptions is `java.lang.Throwable`. `java.lang.Error` and `java.lang.Exception` are the only two subclasses of `Throwable`.

Error is used by the JVM to throw exception that have nothing to do with the program code as such but occur because of environment. For example, `OutOfMemoryError`. It

indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions. Error and its subclasses are regarded as unchecked exceptions for the purposes of compile-time checking of exceptions.

Exception is used by the programmer as well as the JVM when it encounters exceptional situation in the program. Exception and its subclasses (except RuntimeException) are called Checked Exceptions. Checked exceptions need to be declared in a method or constructor's throws clause if they can be thrown by the execution of the method or constructor and propagate outside the method or constructor boundary. For example, `java.io.IOException`.

`RuntimeException` extends `Exception`, which is used to report exceptional situations that cannot be predetermined at compile time. For example, `IndexOutOfBoundsException` or `NullPointerException`. `RuntimeException` and its subclasses are unchecked exceptions. Unchecked exceptions do not need to be declared in a method or constructor's throws clause.

Back to Question without Answer

27. QID - 2.1276 : Handling Exceptions

What is wrong with the following code?

```
class MyException extends Exception {}
public class TestClass{
   public static void main(String[] args){
      TestClass tc = new TestClass();
      try{
         tc.m1();
      }
      catch (MyException e){
         tc.m1();
      }
      finally{
         tc.m2();
      }
   }
   public void m1() throws MyException{
      throw new MyException();
   }
   public void m2() throws RuntimeException{
      throw new NullPointerException();
   }
}
```

Correct Option is : D

~~**A.**~~ It will not compile because you cannot throw an exception in finally block.

> You can, but then you have to declare it in the method's throws clause.

~~**B.**~~ It will not compile because you cannot throw an exception in catch block.

> You can, but then you have to declare it in the method's throws clause.

~~**C.**~~ It will not compile because NullPointerException cannot be created this way.

It does have a no args constructor.

D. It will not compile because of unhandled exception.

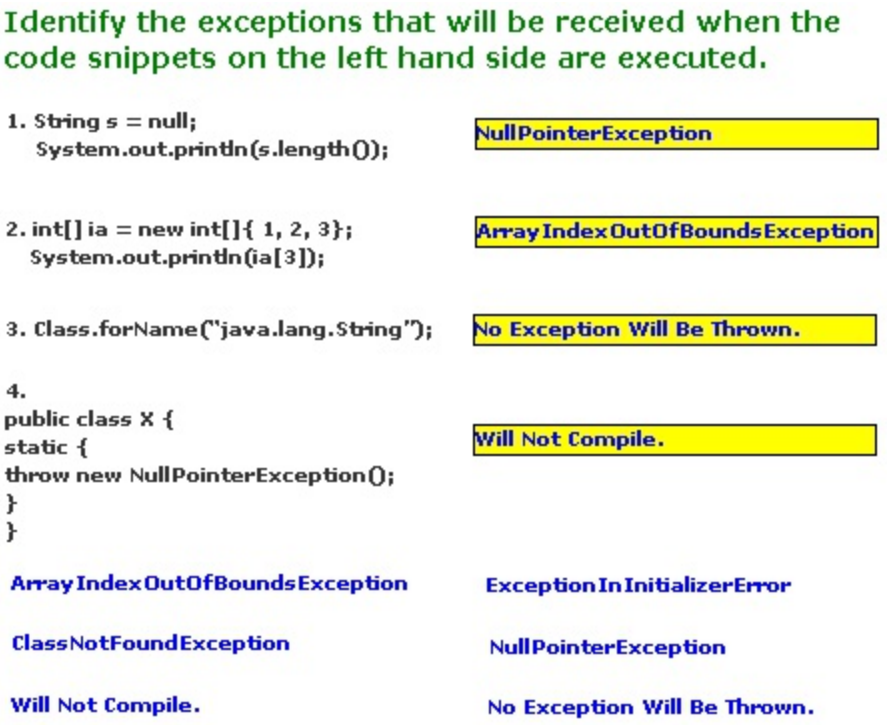
~~**E.**~~ It will compile but will throw an exception when run.

Explanation:

The catch block is throwing a checked exception (i.e. non-RuntimeException) which must be handled by either a try catch block or declared in the throws clause of the enclosing method.
Note that finally is also throwing an exception here, but it is a RuntimeException so there is no need to handle it or declare it in the throws clause.

Back to Question without Answer

28. QID - 2.1031 : Handling Exceptions



Explanation:

Please read ExceptionClassSummary document in the "Study References" section. Note that the question is not asking what exception you need to put in the catch(...) part or throws clause. It is just asking what exceptions will be thrown by the code fragments when they are executed.

Back to Question without Answer

29. QID - 2.1025 : Handling Exceptions

Identify the correct constructs.

Correct Option is : A

A.

```
try {
  for( ;; );
}finally { }
```

A try block must be accompanied by either a catch block or a finally block or both.

~~B.~~

```
try {
  File f = new File("c:\a.txt");
} catch {  f = null; }
```

Invalid syntax for catch. A catch must have a exception: catch(SomeException se){ }

~~C.~~

```
int k = 0;
try {
  k = callValidMethod();
}
System.out.println(k);
catch {  k = -1; }
```

There cannot be any thing between a catch or a finally block and the closing brace of the previous try or catch block.

~~D.~~

```
try {
  try {
     Socket s = new ServerSocket(3030);
  }catch(Exception e) {
    s = new ServerSocket(4040);
  }
}
```

The first try doesn't have any catch or finally block.

~~E.~~

```
try {
       s = new ServerSocket(3030);
}
catch(Exception t){ t.printStackTrace(); }
catch(IOException e) {
    s = new ServerSocket(4040);
}
catch(Throwable t){ t.printStackTrace();  }
```

You can have any number of catch blocks in any order but each catch must be of a different type. Also, a catch for a subclass exception should occur before a catch block for the superclass exception. Here, IOException is placed before Throwable, which is good but Exception is placed before IOException, which is invalid and will not compile.

~~F.~~

```
int x = validMethod();
try {
  if(x == 5) throw new IOException();
  else if(x == 6) throw new Exception();
}finally {
 x = 8;
}
```

```
catch(Exception e){ x = 9; }
```

finally cannot occur before any catch block.

Explanation:

Note that a try with resources block may or may not to have any catch or finally block at all. However, try with resources is not in scope for this exam.

Back to Question without Answer

30. QID - 2.1210 : Handling Exceptions

Consider the following code snippet:

```
void m1() throws Exception{
   try{
      // line1
   }
   catch (IOException e){
       throw new SQLException();
   }
   catch(SQLException e){
       throw new InstantiationException();
   }
   finally{
      throw new CloneNotSupportedException();   // this is not a Runt
   }
}
```

Which of the following statements are true?

Correct Options are : B D

~~A.~~ If IOException gets thrown at line1, then the whole method will end up throwing SQLException.

B. If IOException gets thrown at line1, then the whole method will end up throwing CloneNotSupportedException.

~~C.~~ If IOException gets thrown at line1, then the whole method will end up throwing InstantiationException()

D. If no exception is thrown at line1, then the whole method will end up throwing

CloneNotSupportedException.

~~E.~~ If SQLException gets thrown at line1, then the whole method will end up throwing InstantiationException()

Explanation:

The fundamental concepts at play here are:

1. The Exception that is thrown in the last, is the Exception that is thrown by the method to the caller.
So, when no exception or any exception is thrown at line 1, the control goes to finally or some catch block. Now, even if the catch blocks throws some exception, the control goes to finally. The finally block throws CloneNotSupportedException, so the method ends up throwing CloneNotSupportedException. Other exceptions thrown by the code prior to this point are lost.

2. Exception thrown by a catch cannot be caught by the following catch blocks at the same level. So, if IOException is thrown at line 1, the control goes to first catch which throws SQLException. Now, although there is a catch for SQLException, it won't catch the exception because it is at the same level. So, the control goes to the finally and same process as explained above continues. Any exceptions thrown before this exception are lost.

Back to Question without Answer

31. QID - 2.842 : Handling Exceptions

A new Java programmer has written the following method that takes an array of integers and sums up all the integers that are less than 100.

```
public void processArray(int[] values){
        int sum = 0;
        int i = 0;
        try{
            while(values[i]<100){
                sum = sum +values[i];
                i++;
            }
        }
        catch(Exception e){ }
        System.out.println("sum = "+sum);
    }
```

Which of the following are best practices to improve this code?

Correct Options are : B D

~~A.~~ Use `ArrayIndexOutOfBoundsException` for the catch argument.

B. Use `ArrayIndexOutOfBoundsException` for the catch argument and add code in the catch block to log or print the exception.

> Empty catch blocks are a bad practice because at run time, if the exception is thrown, the program will not show any sign of the exception and may produce bad results that will be hard to debug. Therefore, it is a good practice to at least print out the exception if you don't want to do any thing upon encountering an exception.

~~C.~~ Add code in the catch block to handle the exception.

There are a few questions in the exam that are difficult to interpret. In this case, for example, it is not clear what is meant by handling the exception. The catch block itself is meant to handle the exception. Once you get the exception, you can do what ever is required in the catch block.

D. Use flow control to terminate the loop.

It is considered a bad practice to use exceptions to control the flow of execution. In this case, `values[i]` will throw an `ArrayIndexOutOfBoundsException` once it goes beyond the array length and the programmer is using this fact to control the loop. Instead of doing this, the programmer should use something like: `for(int i=0; i<values.length; i++)` to control the execution of the loop.

Back to Question without Answer

32. QID - 2.1348 : Handling Exceptions

Which digits and in what order will be printed when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int k = 0;
      try{
         int i = 5/k;
      }
      catch (ArithmeticException e){
         System.out.println("1");
      }
      catch (RuntimeException e){
         System.out.println("2");
         return ;
      }
      catch (Exception e){
         System.out.println("3");
      }
      finally{
         System.out.println("4");
      }
      System.out.println("5");
   }
}
```

Correct Option is : D

~~**A.**~~ The program will print 5.

~~**B.**~~ The program will print 1 and 4, in that order.

~~**C.**~~ The program will print 1, 2 and 4, in that order.

D. The program will print 1, 4 and 5, in that order.

~~E.~~ The program will print 1,2, 4 and 5, in that order.

Explanation:

Division by 0 throws a java.lang.ArithmeticException, which is a RuntimeException. This is caught by the first catch clause because it is the first block that can handle ArithmeticException. This prints 1. Now, as the exception is already handled, control goes to finally which prints 4 and then the try/catch/finally ends and 5 is printed. Remember : finally is always executed even if try or catch return; (Except when there is System.exit() in try.)

Back to Question without Answer

33. QID - 2.954 : Handling Exceptions

What class of objects can be declared by the throws clause?

Correct Options are : A B E

A. Exception

B. Error

~~**C.**~~ Event

~~**D.**~~ Object

E. RuntimeException

Explanation:

You can declare anything that is a Throwable or a subclass of Throwable, in the throws clause.

Back to Question without Answer

34. QID - 2.959 : Handling Exceptions

What will the following class print ?

```
class Test{
   public static void main(String[] args){
      int[][] a = { { 00, 01 }, { 10, 11 } };
      int i = 99;
      try {
         a[val()][i = 1]++;
      } catch (Exception e) {
         System.out.println( i+", "+a[1][1]);
      }
   }
   static int val() throws Exception {
     throw new Exception("unimplemented");
   }
}
```

Correct Option is : A

A. 99 , 11

~~**B.**~~ 1 , 11

~~**C.**~~ 1 and an unknown value.

~~**D.**~~ 99 and an unknown value.

~~**E.**~~ It will throw an exception at Run time.

Explanation:

If evaluation of a dimension expression completes abruptly, no part of any dimension expression to its right will appear to have been evaluated.
Thus, while evaluating a[val()][i=1]++, when val() throws an exception, i=1 will not be executed. Therefore, i remains 99 and a[1][1] will print 11.

Back to Question without Answer

35. QID - 2.1211 : Handling Exceptions

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      Exception e = null;
      throw e;
   }
}
```

Correct Option is : A

A. The code will fail to compile.

B. The program will fail to compile, since it cannot throw a `null`.

C. The program will compile without error and will throw an `Exception` when run.

D. The program will compile without error and will throw `java.lang.NullPointerException` when run

E. The program will compile without error and will run and terminate without any output.

Explanation:

You are throwing an exception and there is no try or catch block, or a throws clause. So it will not compile.
If you do either put a try catch block or declare a throws clause for the method then it will throw a `NullPointerException` at run time because e is `null`.

A method that throws a 'checked' exception i.e. an exception that is not a subclass of `Error` or `RuntimeException`, either must declare it in throws clause or put the code that throws the exception in a try/catch block.

Back to Question without Answer

36. QID - 2.1236 : Handling Exceptions

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     int x = 1;
     int y = 0;
     if( x/y ) System.out.println("Good");
     else  System.out.println("Bad");
  }
}
```

Correct Option is : D

~~**A.** Good~~

~~**B.** Bad~~

~~**C.** Exception at runtime saying division by Zero.~~

D. It will not compile.

You need a boolean in the 'if' condition. Here, compiler sees that there is no way x/y can produce a boolean so it generates an error at compile time.

~~**E.** None of the above.~~

Back to Question without Answer

37. QID - 2.979 : Handling Exceptions

What will be the output when the following code is compiled and run?

```
//in file Test.java
class E1 extends Exception{ }
class E2 extends E1 { }
class Test{
   public static void main(String[] args){
      try{
         throw new E2();
      }
      catch(E1 e){
         System.out.println("E1");
      }
      catch(Exception e){
         System.out.println("E");
      }
      finally{
         System.out.println("Finally");
      }
   }
}
```

Correct Option is : B

~~**A.**~~ It will not compile.

B. It will print E1 and Finally.

~~**C.**~~ It will print E1, E and Finally.

~~**D.**~~ It will print E and Finally.

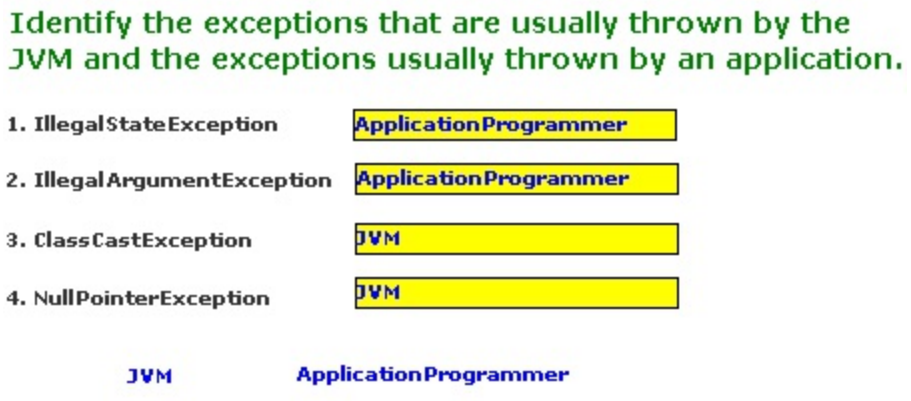
~~**E.**~~ It will print Finally.

Explanation:

Since E2 is a sub class of E1, catch(E1 e) will be able to catch exceptions of class E2. Therefore E1 is printed. Once the exception is caught the rest of the catch blocks at the same level are ignored. So E is not printed. finally is always executed (except in case of System.exit()), so Finally is also printed.

Back to Question without Answer

38. QID - 2.1017 : Handling Exceptions



Explanation:

Note: The terminology "thrown by the JVM" and "thrown programatically or by the application" is not precise but is used by popular books. If it helps, you can think of the exception categories as "thrown implicitly" and "thrown explicitly". An exception that is thrown even when there is no `throw` statement, is said to be thrown implicitly. For example, calling a method on null will cause a `NullPointerException` to be thrown automatically, even though there is no `throw` statement. On the other hand, a code may throw an exception explicitly by using the `throw` statement. For example, a method code might check an argument for validity and if it finds the argument inappropriate, it may throw an exception by executing `throw new IllegalArgumentException();`.

A quick way to determine who should throw an exception is to see if the exception extends `java.lang.Error`. Errors are always thrown only by the JVM.

Generally, `RuntimeException`s are also thrown by the JVM. However, it is ok for an application code to throw a `RuntimeException` if it makes sense for the application to throw a `RuntimeException` in a given situation.

You should know about the following common exception classes:

`IndexOutOfBoundsException extends RuntimeException`:
Usually thrown by the JVM. Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range. Applications can subclass this class to indicate similar exceptions.
`ArrayIndexOutOfBoundsException and StringIndexOutOfBoundsException` both extend `IndexOutOfBoundsException`.

`IllegalArgumentException` extends `RuntimeException`: If a parameter passed to a method is not valid. Usually thrown by the application.

`IllegalStateException extends RuntimeException`: Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation. Usually thrown by the application.

`ClassCastException extends RuntimeException`: Usually thrown by the JVM. Thrown to indicate that the code has attempted to cast an object to a subclass of which it is not an instance. For example, the following code generates a `ClassCastException`:

```
Object x = new Integer(0);
System.out.println((String)x);
```

`NullPointerException extends RuntimeException`: Usually thrown by the JVM. Thrown when an application attempts to use `null` in a case where an object is required. These include:

- Calling the instance method of a null object.
- Accessing or modifying the field of a null object.
- Taking the length of null as if it were an array.
- Accessing or modifying the slots of null as if it were an array.
- Throwing null as if it were a Throwable value.

Applications should throw instances of this class to indicate other illegal uses of the null object.

`SecurityException extends RuntimeException`: Usually thrown by the JVM. It is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception.

Back to Question without Answer

39. QID - 2.1030 : Handling Exceptions

Identify the exceptions that SHOULD be thrown
in the situations shown below.

(Assume that CLUBS, DIAMONDS, HEARTS, SPADES are valid elements of an enum.)

```
1.
switch(suit) {
 case CLUBS:                 AssertionError();   IllegalStateException();
 ...
 break;
 case DIAMONDS:              IllegalArgumentException();   Exception();
  ...
 break;
 case HEARTS:
 ...
 break;
 case SPADES:
 ...
 break;
 default : throw new AssertionError();
}

2.
public void applyCode(String code)
{
 if(code.startsWith("XA")) apply(code.substring(2));
 else throw new IllegalArgumentException();
}
```

Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

40. QID - 2.1046 : Handling Exceptions

What will be the output of the following program:

```
public class TestClass{
   public static void main(String args[]){
      try{
         m1();
      }catch(IndexOutOfBoundsException e){
         System.out.println("1");
         throw new NullPointerException();
      }catch(NullPointerException e){
         System.out.println("2");
         return;
      }catch (Exception e) {
         System.out.println("3");
      }finally{
         System.out.println("4");
      }
      System.out.println("END");
   }
   // IndexOutOfBoundsException is a subclass of RuntimeException.
   static void m1(){
      System.out.println("m1 Starts");
      throw new IndexOutOfBoundsException( "Big Bang " );
   }
}
```

Correct Options are : A B E

A. The program will print `m1 Starts`.

B. The program will print `m1 Starts, 1` and `4`, in that order.

~~**C.**~~ The program will print `m1 Starts, 1` and `2`, in that order.

~~**D.**~~ The program will print `m1 Starts, 1, 2` and `4` in that order.

E. `END` will not be printed.

Explanation:

The `IndexOutOfBoundsException` is handled by the first catch block. Inside this block, a new `NullPointerException` is thrown. As this exception is not thrown inside the try block, it will not be caught by any of the remaining catch blocks. It will actually be sent to the caller of the `main()` method after the `finally` block is executed. (Hence '4' in the output.)

The code that prints `END` is never reached, since the `NullPointerException` remains uncaught after the execution of the finally block.

At the end a stack trace for the `NullPointerException` will be printed.

Back to Question without Answer

41. QID - 2.1093 : Handling Exceptions

Which statements regarding the following code are correct ?

```
class Base{
   void method1() throws java.io.IOException, NullPointerException{
      someMethod("arguments");
      // some I/O operations
   }
   int someMethod(String str){
      if(str == null) throw new NullPointerException();
      else return str.length();
   }
}
public class NewBase extends Base{
      void method1(){
           someMethod("args");
      }
}
```

Correct Options are : A E

A. method1 in class NewBase does not need to specify any exceptions.

~~**B.**~~ The code will not compile because RuntimeExceptions cannot be given in throws clause.

> Any Exception can be specified in the throws clause.

~~**C.**~~ method1 in class NewBase must at least give `IOException` in its throws clause.

~~**D.**~~ method1 in class NewBase must at least give `NullpointerException` in its throws clause.

This is not needed because NullPointerException is a RuntimeException.

E. There is no problem with the code.

Explanation:

Overriding method only needs to specify a subset of the list of exception classes the overridden method can throw. A set of no classes is a valid subset of that list.

Remember that NullPointerException is a subclass of RuntimeException, while IOException is a subclass of Exception.

Back to Question without Answer

42. QID - 2.1097 : Handling Exceptions

What will be the output of the following program?

```
class TestClass{
   public static void main(String[] args) throws Exception{
      try{
         amethod();
         System.out.println("try");
      }
      catch(Exception e){
         System.out.println("catch");
      }
      finally   {
         System.out.println("finally");
      }
      System.out.println("out");
   }
   public static void amethod(){ }
}
```

Correct Option is : B

~~**A.**~~ `try finally`

B. `try finally out`

~~**C.**~~ `try out`

~~**D.**~~ `catch finally out`

~~**E.**~~ It will not compile because `amethod()` does not throw any exception.

Explanation:

Since the method `amethod()` does not throw any exception, `try` is printed and the control goes to finally which prints `finally`. After that `out` is printed.

Back to Question without Answer

43. QID - 2.827 : Handling Exceptions

What will be the output when the following program is run?

```
package exceptions;
public class TestClass{
    public static void main(String[] args) {
        try{
            hello();
        }
        catch(MyException me){
            System.out.println(me);
        }
    }

    static void hello() throws MyException{
        int[] dear = new int[7];
        dear[0] = 747;
        foo();
    }

    static void foo() throws MyException{
        throw new MyException("Exception from foo");
    }
}

class MyException extends Exception {
    public MyException(String msg){
        super(msg);
    }
}
```

(Assume that line numbers printed in the messages given below are correct.)

Correct Option is : C

~~**A.**~~ `Exception in thread "main"`
`java.lang.ArrayIndexOutOfBoundsException: 10`

```
    at exceptions.TestClass.doTest(TestClass.java:24)
    at exceptions.TestClass.main(TestClass.java:14)
```

> You are creating an array of length 7. Since array numbering starts with 0, the first element would be array[0]. So ArrayIndexOutOfBoundsException will NOT be thrown.

B. ~~`Error in thread "main" java.lang.ArrayIndexOutOfBoundsException`~~

> java.lang.ArrayIndexOutOfBoundsException extends java.lang.RuntimeException, which in turn extends java.lang.Exception. Therefore, ArrayIndexOutOfBoundsException is an Exception and not an Error.

C. `exceptions.MyException: Exception from foo`

D. ~~`exceptions.MyException: Exception from foo`~~

```
    at exceptions.TestClass.foo(TestClass.java:29)
    at exceptions.TestClass.hello(TestClass.java:25)
    at exceptions.TestClass.main(TestClass.java:14)
```

> me.printStackTrace() would have produced this output.

Explanation:

Note that there are a few questions in the exam that test your knowledge about how exception messages are printed.

When you use `System.out.println(exception)`, a stack trace is not printed. Just the name of the exception class and the message is printed.
When you use `exception.printStackTrace()`, a complete chain of the names of the methods called, along with the line numbers, is printed from the point where the exception was thrown and up to the point where the exception was caught and `printStackTrace()` was called.

Back to Question without Answer

44. QID - 2.1026 : Handling Exceptions

Given the class

```
// Filename: Test.java
public class Test{
   public static void main(String args[]){
      for(int i = 0; i< args.length; i++){
         System.out.print("  "+args[i]);
      }
   }
}
```

Now consider the following 3 options for running the program:

```
a: java Test
b: java Test param1
c: java Test param1 param2
```

Which of the following statements are true?

Correct Options are : D E

~~A.~~ The program will throw `java.lang.ArrayIndexOutofBoundsException` on option a.

~~B.~~ The program will throw `java.lang.NullPointerException` on option a.

~~C.~~ The program will print `Test param1` on option b.

> Unlike in C++, Name of the file is not passed in args because for a public class it is always same as the name of the class.

D. It will print `param1 param2` on option c.

E. It will not print anything on option a.

Explanation:

It will not throw `NullPointerException` because `args[]` is never `null`. If no argument is given (as in option a) then the length of args is 0.

Back to Question without Answer

45. QID - 2.1323 : Handling Exceptions

What is the result of compiling and running this code?

```
class MyException extends Throwable{}
class MyException1 extends MyException{}
class MyException2 extends MyException{}
class MyException3 extends MyException2{}
public class ExceptionTest{
   void myMethod() throws MyException{
      throw new MyException3();
   }
   public static void main(String[] args){
      ExceptionTest et = new ExceptionTest();
      try{
         et.myMethod();
      }
      catch(MyException me){
         System.out.println("MyException thrown");
      }
      catch(MyException3 me3){
         System.out.println("MyException3 thrown");
      }
      finally{
         System.out.println(" Done");
      }
   }
}
```

Correct Option is : E

~~**A.**~~ MyException thrown

~~**B.**~~ MyException3 thrown

~~**C.**~~ MyException thrown Done

~~**D.**~~ MyException3 thrown Done

E. It fails to compile

Explanation:

You can have multiple catch blocks to catch different kinds of exceptions, including exceptions that are subclasses of other exceptions. However, the catch clause for more specific exceptions (i.e. a "SubClassException") should come before the catch clause for more general exceptions (i.e. a "SuperClassException"). Failure to do so results in a compiler error as the more specific exception is unreachable.

In this case, catch for MyException3 cannot follow catch for MyException because if MyException3 is thrown, it will be caught by the catch clause for MyException. And so, there is no way the catch clause for MyException3 can ever execute. And so it becomes an "unreachable" statement.

Back to Question without Answer

46. QID - 2.964 : Handling Exceptions

What letters, and in what order, will be printed when the following program is compiled and run?

```
public class FinallyTest{
   public static void main(String args[]) throws Exception{
       try{
          m1();
          System.out.println("A");
       }
       finally{
          System.out.println("B");
       }
       System.out.println("C");
   }
   public static void m1() throws Exception { throw new Exception();
}
```

Correct Option is : C

~~**A.**~~ It will print C and B, in that order.

~~**B.**~~ It will print A and B, in that order.

C. It will print B and throw Exception.

~~**D.**~~ It will print A, B and C in that order.

~~**E.**~~ Compile time error.

Explanation:

An `Exception` is thrown in method `m1()` so `println("A")` will not be executed. As there is no catch block the exception will not be handled and the `main()` method will propagate the exception. So `println("C");` will also not be executed. 'finally' block is always executed (even if there is a return in try but not if there is System.exit()) so `println("B")` is executed.

Back to Question without Answer

47. QID - 2.1260 : Handling Exceptions

What will be the output of the following class...

```
class Test{
   public static void main(String[] args){
      int j = 1;
      try{
         int i = doIt() / (j = 2);
      } catch (Exception e){
         System.out.println(" j = " + j);
      }
   }
   public static int doIt() throws Exception {  throw new Exception('
}
```

Correct Option is : A

A. `It will print j = 1;`

B. `It will print j = 2;`

C. The value of j cannot be determined.

D. It will not compile.

E. None of the above.

Explanation:

If evaluation of the left-hand operand of a binary operator completes abruptly, no part

of the right-hand operand appears to have been evaluated.
So, as doIt() throws exception, j = 2 never gets executed.

Back to Question without Answer

48. QID - 2.1311 : Handling Exceptions

The following class will not throw a `NullPointerException` when compiled and run.

```
class Test{
   public static void main(String[] args) throws Exception{
      int[] a = null;
      int i = a [ m1() ];
   }
   public static int m1() throws Exception{
      throw new Exception("Some Exception");
   }
}
```

Correct Option is : A

A. True

B. False

Explanation:

A NullPointerException never occurs because the index expression must be completely evaluated before any part of the indexing operation occurs, and that includes the check as to whether the value of the left-hand operand is null.
If the array reference expression produces null instead of a reference to an array, then a NullPointerException is thrown at runtime, but only after all parts of the array reference expression have been evaluated and only if these evaluations completed normally.

In an array access, the expression to the left of the brackets appears to be fully evaluated before any part of the expression within the brackets is evaluated.
Note that if evaluation of the expression to the left of the brackets completes abruptly,

no part of the expression within the brackets will appear to have been evaluated.

Here, m1() is called first, which throws Exception and so 'a' is never accessed and NullPointerException is never thrown.

Back to Question without Answer

49. QID - 2.1255 : Handling Exceptions

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      loop :          // 1
      {
         System.out.println("Loop Lable line");
         try{
            for (  ;  true ;  i++ ){
               if( i >5) break loop;        // 2
            }
         }
         catch(Exception e){
            System.out.println("Exception in loop.");
         }
         finally{
            System.out.println("In Finally");       // 3
         }
      }
   }
}
```

Correct Option is : C

~~**A.**~~ Compilation error at line 1 as this is an invalid syntax for defining a label.

> You can apply a label to any code block or a block level statement (such as a `for` statement) but not to declarations. For example: `loopX : int i = 10;`

~~**B.**~~ Compilation error at line 2 as 'loop' is not visible here.

C. No compilation error and line 3 will be executed.

> Even if the break takes the control out of the block, the finally clause will be executed.

~~**D.**~~ No compilation error and line 3 will NOT be executed.

~~**E.**~~ Only the line with the label Loop will be printed.

Explanation:

A `break` without a label breaks the current loop (i.e. no iterations any more) and a break with a label tries to pass the control to the given label.
'Tries to' means that if the break is in a try block and the try block has a finally clause associated with it then it will be executed.

Back to Question without Answer

50. QID - 2.984 : Handling Exceptions

Following is a supposedly robust method to parse an input for a float :

```
public float parseFloat(String s){
   float f = 0.0f;
   try{
      f = Float.valueOf(s).floatValue();
      return f ;
   }
   catch(NumberFormatException nfe){
      System.out.println("Invalid input " + s);
      f = Float.NaN ;
      return f;
   }
   finally { System.out.println("finally");  }
   return f ;
}
```

Which of the following statements about the above method are true??

Correct Option is : E

~~**A.**~~ If input is `"0.1"` then it will return `0.1` and print `finally`.

~~**B.**~~ If input is `"0x.1"` then it will return `Float.Nan` and print `Invalid Input 0x.1` and `finally`.

~~**C.**~~ If input is `"1"` then it will return `1.0` and print `finally`.

~~**D.**~~ If input is `"0x1"` then it will return `0.0` and print `Invalid Input 0x1` and `finally`.

E. The code will not compile.

Note that the return statement after finally block is unreachable. Otherwise, if this line were not there, choices 1, 2, 3 are valid.

Back to Question without Answer

51. QID - 2.1048 : Handling Exceptions

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      try{
         RuntimeException re = null;
         throw re;
      }
      catch(Exception e){
         System.out.println(e);
      }
   }
}
```

Correct Option is : D

~~**A.**~~ The code will fail to compile, since `RuntimeException` cannot be caught by catching an Exception.

> `RuntimeException` can be caught by `catch(Exception e)` statement because `RuntimeException` is a subclass of `Exception`.

~~**B.**~~ The program will fail to compile, since `re` is `null`.

~~**C.**~~ The program will compile without error and will print `java.lang.RuntimeException` when run.

D. The program will compile without error and will print `java.lang.NullPointerException` when run.

A `NullPointerException` will be thrown if the expression given to the throw statement results in a null pointer.

~~**E.**~~ The program will compile without error and will run and print `null`.

Explanation:

The try block generates `NullPointerException` which will be caught by the catch block.

Back to Question without Answer

52. QID - 2.1023 : Handling Exceptions

Identify the exceptions that will be received when the code snippets on the left hand side are executed.

```
1. int[] ia = new int[]{ 1, 2, 3};
   System.out.println(ia[-1]);
```

ArrayIndexOutOfBoundsException

```
2.
public class X {
static int k = 0;
static {
k = 10/0;
}
}
```

ExceptionInInitializerError

```
3. SomeClass sc = new SomeClass();
```
(Assume that SomeClass is not available in runtime classpath.)

NoClassDefFoundError

```
4.
public class X {
static {
if(true) throw new NullPointerException();
}
}
```

ExceptionInInitializerError

NoClassDefFoundError NullPointerException ArrayAccessException

ExceptionInInitializerError ArrayIndexOutOfBoundsException

IllegalArrayAccessException NoSuchClassException

Explanation:

Please read ExceptionClassSummary document in the "Study References" section.

Back to Question without Answer

Java Basics

Exam Objectives -

Define the scope of variables Define the structure of a Java class Create executable Java applications with a main method Import other Java packages to make them accessible in your code

01. QID - 2.1122

Note: This question may be considered too advanced for this exam. What will the code shown below print when compiled and run with the following command line?

```
java WarZone self
```

```
interface XMen {
   void shoot(String a);
}

public class WarZone {
   public static void main(String[] args){
       XMen x = null;
       if(args.length() > 0){
          x = new XMen(){
             public void shoot(String s){
                for(int i=0; i<s.length; i++){
                  System.out.println("shot : "+s.charAt(i));
                }
             }
          };
       }

       if(x != null){
          x.shoot(args[0]);
       }
   }
}
```

Select 1 option

A. It will not compile because interface XMen cannot be instantiated.

B. It will print `shot :` 4 times, one at each line.

C. It will print "`shot : s`", "`shot : e`", "`shot : l`", "`shot : f`" one by one on 4 lines.

D. It will compile but will throw an exception at runtime.

E. None of these options is correct.

Check Answer

02. QID - 2.1131

What would be the result of attempting to compile and run the following program?

```
class TestClass{
   static TestClass ref;
   String[] arguments;
   public static void main(String args[]){
      ref = new TestClass();
      ref.func(args);
   }
   public void func(String[] args){
      ref.arguments = args;
   }
}
```

Select 1 option

A. The program will fail to compile, since the static method `main` is trying to call the non-static method `func`.

B. The program will fail to compile, since the non-static method `func` cannot access the static member variable `ref`.

C. The program will fail to compile, since the argument `args` passed to the static method `main` cannot be passed on to the non-static method `func`.

D. The program will fail to compile, since method `func` is trying to assign to the non-static member variable '`arguments`' through the static member variable `ref`.

E. The program will compile and run successfully.

Check Answer

03. QID - 2.1318

Which of the following statements is correct?

Select 1 option

A. new, delete and goto are keywords in the Java language

B. try, catch and thrown are keywords in the Java language

C. static, unsigned and long are keywords in the Java language

D. exit, class and while are keywords in the Java language

E. return, goto and default are keywords in the Java language

Check Answer

04. QID - 2.1232

Which of the following lines can be inserted at line 1 to make the program run?

```
//line 1
public class TestClass{
  public static void main(String[] args){
     PrintWriter pw = new PrintWriter(System.out);
     OutputStreamWriter osw  =  new OutputStreamWriter( System.out
);
     pw.print("hello");
  }
}
```

Assume that `PrintWriter` and `OutputStreamWriter` are valid classes in `java.io` package.

Select 1 option

A. `import java.lang.*;`

B. `import java.io.*;`

C. `import java.io.OutputStreamWriter;`

D. `include java.io.*;`

E. `include java.lang.System;`

Check Answer

05. QID - 2.992

Given the following program, which statement is true?

```
class SomeClass{
   public static void main( String args[ ] ){
      if (args.length == 0 ){
         System.out.println("no arguments") ;
      }
      else{
         System.out.println( args.length + " arguments") ;
      }
   }
}
```

Select 1 option

A. The program will fail to compile.

B. The program will throw a `NullPointerException` when run with zero arguments.

C. The program will print `no arguments` and `1 arguments` when called with zero and one arguments.

D. The program will print `no arguments` and `2 arguments` when called with zero and one arguments.

E. The program will print `no arguments` and `3 arguments` when called with zero and one arguments.

Check Answer

06. QID - 2.1079

Consider the following two classes defined in two .java files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1  <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       System.out.println(X.LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Select 1 option

A. `import static X;`

B. `import static com.foo.*;`

C. `import static com.foo.X.*;`

D. `import com.foo.*;`

E. `import com.foo.X.LOGICID;`

Check Answer

07. QID - 2.943

Note: Option 4 of this question may be considered too advanced for this exam.

Which lines of code will not be acceptable to the compiler?

```
import java.*; //1
public abstract class InternalLogic //2
{
   float density = 20.0; //3
   public class Doer //4
   {
      void do() //5
      {
         //lot of valid code.
      }
   }
}
```

Select 2 options

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

08. QID - 2.867

Which of the following keywords may occur multiple times in a Java source file?

Select 4 options

A. import

B. class

C. private

D. package

E. public

Check Answer

09. QID - 2.928

Given the following set of member declarations, which of the following is true?

```
int a;     //  (1)
static int a;     //  (2)
int f( )    { return a; }     //  (3)
static int f( ) { return a; }     //  (4)
```

Select 2 options

A. Declarations (1) and (3) cannot occur in the same class definition.

B. Declarations (2) and (4) cannot occur in the same class definition.

C. Declarations (1) and (4) cannot occur in the same class definition.

D. Declarations (2) and (3) cannot occur in the same class definition.

E. Declarations (1) and (2) cannot occur in the same class definition.

Check Answer

10. QID - 2.1321

Consider the following class :

```
public class Parser{
   public static void main( String[] args){
       try{
           int i = 0;
           i =  Integer.parseInt( args[0] );
       }
       catch(NumberFormatException e){
          System.out.println("Problem in " + i );
       }
   }
}
```

What will happen if it is run with the following command line:

```
java Parser one
```

Select 1 option

A. It will print `Problem in 0`

B. It will throw an exception and end without printing anything.

C. It will not even compile.

D. It will not print anything if the argument is '1' instead of 'one'.

E. None of the above.

Check Answer

11. QID - 2.1341

What will be result of attempting to compile this class?

```
import java.util.*;
package test;
public class TestClass{
    public OtherClass oc = new OtherClass();
}
class OtherClass{
    int value;
}
```

Select 1 option

A. The class will fail to compile, since the class OtherClass is used before it is defined.

B. There is no problem with the code.

C. The class will fail to compile, since the class OtherClass must be defined in a file called OtherClass.java

D. The class will fail to compile .

E. None of the above.

Check Answer

12. QID - 2.1056

Which one of these is a proper definition of a class TestClass that cannot be sub-classed?

Select 1 option

A. `final class TestClass { }`

B. `abstract class TestClass { }`

C. `native class TestClass { }`

D. `static class TestClass { }`

E. `private class TestClass { }`

Check Answer

13. QID - 2.1085

Consider the following two classes defined in two java source files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1 <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       X x = new X();
       x.apply(LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Select 2 options

A. `import static X;`

B. `import static com.foo.*;`

C. `import static com.foo.X.*;`

D. `import com.foo.*;`

E. `import com.foo.X.LOGICID;`

Check Answer

14. QID - 2.1069

Which method declarations will enable a class to be run as a standalone program?

Select 2 options

A. `static void main(String args[ ])`

B. `public void static main(String args[ ])`

C. `public static main(String[ ] argv)`

D. `final public static void main(String [ ] array)`

E. `public static void main(String args[ ])`

Check Answer

15. QID - 2.1080

Which of the following are valid declarations of the standard main() method?

Select 2 options

A. `static void main(String args[ ]) { }`

B. `public static int main(String args[ ]) {}`

C. `public static void main (String args) { }`

D. `final static public void main (String[ ] arguments ) { }`

E. `public static void main (String[ ] args) { }`

Check Answer

16. QID - 2.1004

Which of these statements concerning the use of modifiers are true?

Select 1 option

A. By default (i.e. no modifier) the member is only accessible to classes in the same package and subclasses of the class.

B. You cannot specify visibility of local variables.

C. Local variable always have default accessibility.

D. Local variables can be declared as private.

E. Local variables can only be declared as public.

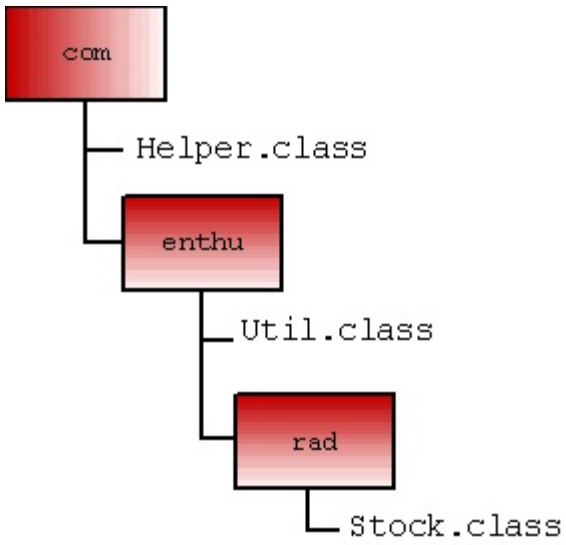
Check Answer

17. QID - 2.1066

Consider the following directory structure shown in Image 1 that displays available folders and classes and the code given below.

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements must be added to the above class?



Select 2 options

A. `import com.enthu.*;`

B. `import com.*.*;`

C. `import *.*.*;`

D. `import com.*;`

E. `import com.enthu.rad.*;`

F. `import all;`

Check Answer

18. QID - 2.1141

Which line(s) in the following code will cause a compilation error?

```
static import java.lang.System.*; //1
class $$ //2
{
    static public void main(String... _$_) //3
   {
       String _ = ""; //4
       for(int $=0; ++$ < _$_.length; )  //5
           _ += _$_[$];  //6
       out.println(_);  //7
    }
}
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. 5

F. 6

G. 7

H. None of the lines is invalid.

Check Answer

19. QID - 2.1163

The following is a valid member variable declaration:

```
private static final transient int i = 20;
```

Select 1 option

A. True

B. False

Check Answer

20. QID - 2.1364

Given the following class, which of these are valid ways of referring to the class from outside of the package com.enthu?

```
package com.enthu;
public class Base{
   // ....
   // lot of code...
}
```

Select 2 options

A. Base

B. By importing the package `com.*` and referring to the class as `enthu.Base`

C. importing `com.*` is illegal.

D. By importing `com.enthu.*` and referring to the class as `Base`.

E. By referring to the class as `com.enthu.Base`.

Check Answer

21. QID - 2.845

The options below contain the complete contents of a file (the name of the file is not specified).
Which of these options can be run with the following command line once compiled?

```
java main
```

Select 1 option

A.
```
//in file main.java
class main {
   public void main(String[] args) {
       System.out.println("hello");
   }
}
```

B.
```
//in file main.java
   public static void main4(String[] args) {
       System.out.println("hello");
   }
```

C.
```
//in file main.java
public class anotherone{
}
class main {
   public static void main(String[] args) {
       System.out.println("hello");
   }
}
```

D.
```
//in file main.java
class anothermain{
   public static void main(String[] args) {
       System.out.println("hello2");
```

```
    }
}
class main {
    public final static void main(String[] args) {
        System.out.println("hello");
    }
}
```

Check Answer

22. QID - 2.976

Consider the following two java files:

```
//in file SM.java
package x.y;
public class SM{
    public static void foo(){ };
}


//in file TestClass.java
//insert import statement here //1
public class TestClass{
   public static void main(String[] args){
      foo();
   }
}
```

What should be inserted at //1 so that TestClass will compile and run?

Select 2 options

A. `import static x.y.*;`

B. `import static x.y.SM;`

C. `import static x.y.SM.foo;`

D. `import static x.y.SM.foo();`

E. `import static x.y.SM.*;`

Check Answer

23. QID - 2.1272

Which of these are not legal declarations within a class?

Select 1 option

A. `static volatile int sa ;`

B. `final Object[ ] objArr = { null } ;`

C. `abstract int t ;`

D. `native void format( ) ;`

E. `final transient static private double PI = 3.14159265358979323846 ;`

Check Answer

24. QID - 2.1132

Which of the following is/are illegal Java identifier(s)?

Select 1 option

A. num

B. int123

C. 2Next

D. _interface

E. a$_123

Check Answer

25. QID - 2.1118

Which of the changes given in options can be done (independent of each other) to let the following code compile and run without errors?

```
class SomeClass{
   String s1 = "green mile";   // 0
   public void generateReport( int n ){
      String local;   // 1
      if( n > 0 ) local = "good";   //2
      System.out.println( s1+" = " + local );   //3
   }
}
```

Select 2 options

A. Insert after line 2 : `else local = "bad";`

B. Insert after line 2 : `if(n <= 0) local = "bad";`

C. Move line 1 and place it after line 0.

D. change line 1 to : `final String local = "rocky";`

E. The program already is without any errors.

Check Answer

26. QID - 2.944

Which of the following are valid declarations:

Select 3 options

A. int a = b = c = 100;

B. int a, b, c; a = b = c = 100;

C. int a, b, c=100;

D. int a=100, b, c;

E. int a= 100 = b = c;

Check Answer

27. QID - 2.1007

How can you declare a method someMethod() such that an instance of the class is not needed to access it and all the members of the same package have access to it.

Select 3 options

A. `public static void someMethod()`

B. `static void someMethod()`

C. `protected static void someMethod()`

D. `void someMethod()`

E. `protected void someMethod()`

F. `public abstract static void someMethod()`

Check Answer

28. QID - 2.1142

Which of the given options should be inserted at line 1 so that the following code can compile without any errors?

```
package objective1;
// 1
public class StaticImports{

    public StaticImports(){
    out.println(MAX_VALUE);
    }

}
```

(Assume that java.lang.Integer has a static field named MAX_VALUE)

Select 2 options

A. `import static java.lang.Integer.*;`

B. `static import java.lang.System.out;`

C. `static import Integer.MAX_VALUE;`

D. `import static java.lang.System.*;`

E. `static import java.lang.System.*;`

Check Answer

29. QID - 2.1333

Which of the following are not legal Java identifiers?

Select 1 option

A. goto

B. unsigned

C. String

D. _xyz

E. $_abc

F. iLikeVeryVeryVeryVeryVeryLongIdentifiersThatDontMakeAnySenseAtAll (65 characters)

Check Answer

30. QID - 2.1197

Which of the following are keywords in Java?

Select 4 options

A. default

B. NULL

C. String

D. throws

E. long

F. strictfp

Check Answer

31. QID - 2.1360

Given the following code, which statements can be placed at the indicated position without causing compile and run time errors?

```
public class Test{
   int i1;
   static int i2;
   public void method1(){
      int i;
      // ... insert statements here
   }
}
```

Select 3 options

A. `i = this.i1;`

B. `i = this.i2;`

C. `this = new Test( );`

D. `this.i = 4;`

E. `this.i1 = i2;`

Check Answer

32. QID - 2.1020

What does the zeroth element of the string array passed to the standard main method contain?

Select 1 option

A. The name of the class.

B. The string "java".

C. The number of arguments.

D. The first argument of the argument list, if present.

E. None of the above.

Check Answer

33. QID - 2.940

Which of the statements regarding the following code are correct?

```
public class TestClass{
   static int a;
   int b;
   public TestClass(){
      int c;
      c = a;
      a++;
      b += c;
   }
   public static void main(String args[]) {   new TestClass();   }
}
```

Select 1 option

A. The code will fail to compile because the constructor is trying to access static members.

B. The code will fail to compile because the constructor is trying to use static member variable a before it has been initialized.

C. The code will fail to compile because the constructor is trying to use member variable b before it has been initialized.

D. The code will fail to compile because the constructor is trying to use local variable c before it has been initialized.

E. The code will compile and run without any problem.

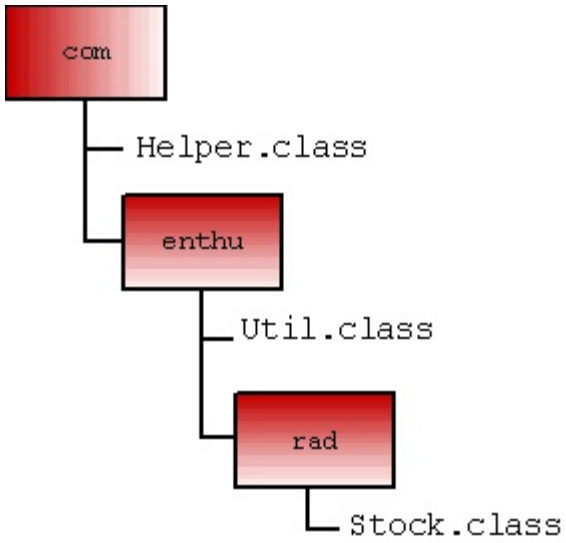
Check Answer

34. QID - 2.1063

Consider the directory structure shown in Image 1 that displays available folders and classes and the code given below:

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public void store() throws IOException{
    return Util.store(stock);
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements MUST be added to the above class?



Select 4 options

A. `package com.enthu.rad.*;`

B. `import com.enthu.*;`

C. `package com.enthu.rad;`

D. `import com.*;`

E. `import java.io.*;`

F. It is not required to import java.io.* or import java.io.IOException because java.io package is imported automatically.

Check Answer

35. QID - 2.1297

Which of the following are valid identifiers?

Select 2 options

A. class

B. $value$

C. angstrom

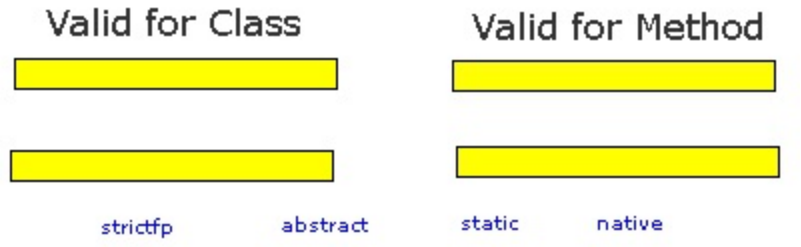
D. 2much

E. zer@

Check Answer

36. QID - 2.1139

Idenfity valid modifiers for a top level class and method declarations.
(Assume that the class is not declared inside another class.)



Check Answer

37. QID - 2.937

Given the following class, which statements can be inserted at line 1 without causing the code to fail compilation?

```
public class TestClass{
   int a;
   int b = 0;
   static int c;
   public void m(){
      int d;
      int e = 0;
      // Line 1
   }
}
```

Select 4 options

A. `a++;`

B. `b++;`

C. `c++;`

D. `d++;`

E. `e++;`

Check Answer

38. QID - 2.1298

Which of the following are valid declarations within a class?

Select 2 options

A. `volatile int k;`

B. `abstract boolean bool;`

C. `native float radix;`

D. `friendly int ArrayList;`

E. `int friendly, ArrayList;`

Check Answer

39. QID - 2.1162

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){}
}
```

Select 1 option

A. The class `TestClass` cannot be declared `abstract`.

B. The variable `j` cannot be declared `transient`.

C. The variable `k` cannot be declared `synchronized`.

D. The constructor `TestClass( )` cannot be declared final.

E. The method `f( )` cannot be declared static.

F. This code has no errors.

Check Answer

40. QID - 2.1258

Consider the following code:

```
class Test{
  public static void main(String[] args){
    for (int i = 0; i < args.length; i++)   System.out.print(i == 0
? args[i] : " " + args[i]);
  }
}
```

What will be the output when it is run using the following command:

java Test good bye friend!

Select 1 option

A. good bye friend!

B. good good good

C. goodgoodgood

D. good bye

E. None of the above.

Check Answer

41. QID - 2.1249

Consider the following class:

```
public class ArgsPrinter{
   public static void main(String args){
      for(int i=0; i<3; i++){
         System.out.print(args+" ");
      }
   }
}
```

What will be printed when the above class is run using the following command line:
`java ArgsPrinter 1 2 3 4`

Select 1 option

A. `1 2 3`

B. `ArgsPrinter 1 2`

C. `java ArgsPrinter 1 2`

D. `1 1 1`

E. None of these.

Check Answer

42. QID - 2.1338

What is the correct parameter specification for the standard main method?

Select 2 options

A. void

B. String[ ] args

C. Strings args[ ]

D. String args

E. String args[ ]

Check Answer

43. QID - 2.1110

Which of these statements are true?

Select 2 options

A. A static method can call other non-static methods in the same class by using the 'this' keyword.

B. A class may contain both static and non-static variables and both static and non-static methods.

C. Each object of a class has its own copy of each non-static member variable.

D. Instance methods may access local variables of static methods.

E. All methods in a class are implicitly passed a 'this' parameter when called.

Check Answer

44. QID - 2.955

An instance member ...

Select 2 options

A. can be a variable, a constant or a method.

B. is a variable or a constant.

C. belongs to the class.

D. belongs to an instance of the class.

E. is same as a local variable.

Check Answer

45. QID - 2.1242

Given:

```
public class TestClass{
     public static void main(String[] args){
     int i = Integer.parseInt(args[1]);
     System.out.println(args[i]);
   }
}
```

What will happen when you compile and run the above program using the following command line:

```
java TestClass 1 2
```

Select 1 option

A. It will print `1`

B. It will print `2`

C. It will print some junk value.

D. It will throw `ArrayIndexOutOfBoundsException`.

E. It will throw NumberFormatException

Check Answer

46. QID - 2.1234

Consider the following code:

```
public abstract class TestClass{
    public abstract void m1();
    public abstract void m2(){
        System.out.println("hello");
    }
}
```

Which of the following corrections can be applied to the above code (independently) so that it compiles without any error?

Select 2 options

A. Replace the method body of m2() with a ; (semi-colon).

B. Replace the ; at the end of m1() with a method body.

C. Remove abstract from m2().

D. Remove abstract from the class declaration.

Check Answer

47. QID - 2.894

The following are the complete contents of TestClass.java file. Which packages are automatically imported?

```
class TestClass{
   public static void main(String[] args){
     System.out.println("hello");
   }
}
```

Select 2 options

A. java.util

B. System

C. java.lang

D. java.io

E. String

F. The package with no name.

Check Answer

48. QID - 2.1170

Consider the classes shown below:

```
class A{
   public A() { }
   public A(int i) {   System.out.println(i );    }
}
class B{
   static A s1 = new A(1);
   A a = new A(2);
   public static void main(String[] args){
      B b = new B();
      A a = new A(3);
   }
   static A s2 = new A(4);
}
```

Which is the correct sequence of the digits that will be printed when B is run?

Select 1 option

A. 1 ,2 ,3 4.

B. 1 ,4, 2 ,3

C. 3, 1, 2, 4

D. 2, 1, 4, 3

E. 2, 3, 1, 4

Check Answer

49. QID - 2.915

Given the following contents of two java source files:

```
package util.log4j;
public class Logger  {
  public void log(String msg){
      System.out.println(msg);
  }
}
```

and

```
package util;
public class TestClass {
    public static void main(String[] args) throws Exception {
        Logger logger = new Logger();
        logger.log("hello");
    }
}
```

What changes, when made independently, will enable the code to compile and run?

Select 2 options

A. Replace `Logger logger = new Logger();` with:

```
log4j.Logger logger = new log4j.Logger();
```

B. Replace package `util.log4j;` with

```
package util;
```

C. Replace `Logger logger = new Logger();` with:

```
util.log4j.Logger logger = new util.log4j.Logger();
```

D. Remove package `util.log4j;` from Logger.

E. Add `import log4j;` to TestClass.

Check Answer

Java Basics (Answered)

01. QID - 2.1122 : Java Basics

Note: This question may be considered too advanced for this exam. What will the code shown below print when compiled and run with the following command line?

```
java WarZone self
```

```
interface XMen {
   void shoot(String a);
}

public class WarZone {
   public static void main(String[] args){
       XMen x = null;
       if(args.length() > 0){
          x = new XMen(){
             public void shoot(String s){
                for(int i=0; i<s.length; i++){
                  System.out.println("shot : "+s.charAt(i));
                }
             }
          };
       }

       if(x != null){
          x.shoot(args[0]);
       }
   }
}
```

Correct Option is : E

~~**A.**~~ It will not compile because interface XMen cannot be instantiated.

An anonymous inner class that implements XMen interface is being created.

~~**B.** It will print `shot :` 4 times, one at each line.~~

~~**C.** It will print "`shot : s`", "`shot : e`", "`shot : l`", "`shot : f`" one by one on 4 lines.~~

> This would be correct, if `args.length()` is changed to `args.length` and `s.length` to `s.length()`.

~~**D.** It will compile but will throw an exception at runtime.~~

E. None of these options is correct.

> Read the question carefully. 'args' is an array and length is an attribute (not a method) of arrays. 's' is a String and there is no attribute length in a String but a method length(). Expect questions that try to confuse by adding misleading statements.

Back to Question without Answer

02. QID - 2.1131 : Java Basics

What would be the result of attempting to compile and run the following program?

```
class TestClass{
   static TestClass ref;
   String[] arguments;
   public static void main(String args[]){
      ref = new TestClass();
      ref.func(args);
   }
   public void func(String[] args){
      ref.arguments = args;
   }
}
```

Correct Option is : E

~~**A.**~~ The program will fail to compile, since the static method `main` is trying to call the non-static method `func`.

> The concept here is that a non-static method (i.e. an instance method) can only be called on an instance of that class. Whether the caller itself is a static method or not, is immaterial.
>
> The main method is calling `ref.func();` - this means the main method is calling a non-static method on an actual instance of the class TestClass (referred to by 'ref'). Hence, it is valid.
> It is not trying calling it directly such as `func()` or `this.func()`, in which case, it would have been invalid.

~~**B.**~~ The program will fail to compile, since the non-static method `func` cannot access the static member variable `ref`.

> Non static methods can access static as well as non static methods of a class.

~~**C.**~~ The program will fail to compile, since the argument `args` passed to the static method `main` cannot be passed on to the non-static method `func`.

> Non sense!

~~**D.**~~ The program will fail to compile, since method `func` is trying to assign to the non-static member variable '`arguments`' through the static member variable `ref`.

E. The program will compile and run successfully.

Back to Question without Answer

03. QID - 2.1318 : Java Basics

Which of the following statements is correct?

Correct Option is : E

~~**A.**~~ new, delete and goto are keywords in the Java language

> 'delete' is not a keyword.

~~**B.**~~ try, catch and thrown are keywords in the Java language

> 'thrown' is not, though 'throws' is.

~~**C.**~~ static, unsigned and long are keywords in the Java language

> There is no 'unsigned'. All primitive types are 'signed' except 'char'.

~~**D.**~~ exit, class and while are keywords in the Java language

> 'exit' is a method name and is not a keyword.

E. return, goto and default are keywords in the Java language

> although not used, 'goto' is a reserved word.

Explanation:

Following is a list of Java keywords :

abstract continue for new switch
assert default goto package synchronized

boolean do if private this
break double implements protected throw
byte else import public throws
case enum instanceof return transient
catch extends int short try
char final interface static void
class finally long strictfp volatile
const float native super while

Technically 'true', 'false' and 'null' are literals but for the purpose of the exam consider them as keywords.

Back to Question without Answer

04. QID - 2.1232 : Java Basics

Which of the following lines can be inserted at line 1 to make the program run?

```
//line 1
public class TestClass{
  public static void main(String[] args){
     PrintWriter pw = new PrintWriter(System.out);
     OutputStreamWriter osw  =  new OutputStreamWriter( System.out
);
     pw.print("hello");
  }
}
```

Assume that `PrintWriter` and `OutputStreamWriter` are valid classes in `java.io` package.

Correct Option is : B

~~**A.**~~ `import java.lang.*;`

> Although you can import java.lang package explicitly, it is not required because this package is always imported by the compiler.

B. `import java.io.*;`

> This will make all the classes of java.io package available.

~~**C.**~~ `import java.io.OutputStreamWriter;`

> This will only make OutputStreamwriter available. PrintWriter will still be unavailable.

~~**D.**~~ `include java.io.*;`

include is not valid keyword in Java.

E. ~~`include java.lang.System;`~~

Back to Question without Answer

05. QID - 2.992 : Java Basics

Given the following program, which statement is true?

```
class SomeClass{
   public static void main( String args[ ] ){
      if (args.length == 0 ){
         System.out.println("no arguments") ;
      }
      else{
         System.out.println( args.length + " arguments") ;
      }
   }
}
```

Correct Option is : C

~~**A.**~~ The program will fail to compile.

~~**B.**~~ The program will throw a `NullPointerException` when run with zero arguments.

C. The program will print `no arguments` and `1 arguments` when called with zero and one arguments.

> The word java and class name are not a part of the argument list.

~~**D.**~~ The program will print `no arguments` and `2 arguments` when called with zero and one arguments.

~~**E.**~~ The program will print `no arguments` and `3 arguments` when called with zero and one arguments.

When the program is called with no arguments, the `args` array will be of length zero.

Explanation:

When the program is called with no arguments, the `args` array will be of length zero. Unlike in C/C++, args[0] is not the name of the program or class. This is because the name of the class is always the same that was defined in the Java file. So there is no need for passing its name as an argument to main method.

Back to Question without Answer

06. QID - 2.1079 : Java Basics

Consider the following two classes defined in two .java files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1  <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       System.out.println(X.LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Correct Option is : D

~~**A.**~~ `import static X;`

~~**B.**~~ `import static com.foo.*;`

> Bad syntax. Package import does not use static keyword.

~~**C.**~~ `import static com.foo.X.*;`

> This static import, although syntactically correct, will not help here because Y

is accessing class X in X.LOGICID.

D. `import com.foo.*;`

This is required because Y is accessing class X. static import of LOGICID is NOT required because Y is accessing LOGICID through X (X.LOGICID). Had it been just System.out.println(LOGICID), only one import statement: import static com.foo.X.*; would have worked.

E. `import com.foo.X.LOGICID;`

Bad Syntax. Syntax for importing static fields is: import static <package>.<classname>.*; or import static <package>.<classname>.<fieldname>;

Back to Question without Answer

07. QID - 2.943 : Java Basics

Note: Option 4 of this question may be considered too advanced for this exam.

Which lines of code will not be acceptable to the compiler?

```
import java.*; //1
public abstract class InternalLogic //2
{
   float density = 20.0; //3
   public class Doer //4
   {
      void do() //5
      {
         //lot of valid code.
      }
   }
}
```

Correct Options are : C E

~~**A.**~~ 1

Although it doesn't import anything but import java.*; is valid.

~~**B.**~~ 2

A class having no abstract method can still be abstract. But not vice-versa.

C. 3

20.0 is a double and needs a cast to be assigned to a float.

~~**D.**~~ 4

It is a valid inner class.

E. 5

do is a keyword so do() is not a valid method name.

Back to Question without Answer

08. QID - 2.867 : Java Basics

Which of the following keywords may occur multiple times in a Java source file?

Correct Options are : A B C E

A. import

B. class

C. private

~~**D.**~~ package

There can be at most one package statement in a Java source file and it must be the first statement in the file.

E. public

Back to Question without Answer

09. QID - 2.928 : Java Basics

Given the following set of member declarations, which of the following is true?

```
int a;     //  (1)
static int a;     //  (2)
int f( )    { return a; }     //  (3)
static int f( ) { return a; }     //  (4)
```

Correct Options are : C E

~~**A.**~~ Declarations (1) and (3) cannot occur in the same class definition.

~~**B.**~~ Declarations (2) and (4) cannot occur in the same class definition.

A static method can refer to a static field.

C. Declarations (1) and (4) cannot occur in the same class definition.

because method f() is static and a is not.

~~**D.**~~ Declarations (2) and (3) cannot occur in the same class definition.

E. Declarations (1) and (2) cannot occur in the same class definition.

variable names must be different.

Explanation:

Local variables can have same name as member variables. The local variables will simply shadow the member variables with the same names.

Declaration (4) defines a static method that tries to access a variable named 'a' which is not locally declared.
Since the method is static, this access will only be valid if variable 'a' is declared static within the class. Therefore declarations (1) and (4) cannot occur in the same definition.

Back to Question without Answer

10. QID - 2.1321 : Java Basics

Consider the following class :

```
public class Parser{
   public static void main( String[] args){
       try{
           int i = 0;
           i =  Integer.parseInt( args[0] );
       }
       catch(NumberFormatException e){
          System.out.println("Problem in " + i );
       }
   }
}
```

What will happen if it is run with the following command line:

```
java Parser one
```

Correct Option is : C

~~**A.**~~ It will print `Problem in 0`

~~**B.**~~ It will throw an exception and end without printing anything.

C. It will not even compile.

Because 'i' is defined in try block and so it is not visible in the catch block.

~~**D.**~~ It will not print anything if the argument is '1' instead of 'one'.

E. None of the above.

Back to Question without Answer

11. QID - 2.1341 : Java Basics

What will be result of attempting to compile this class?

```
import java.util.*;
package test;
public class TestClass{
    public OtherClass oc = new OtherClass();
}
class OtherClass{
    int value;
}
```

Correct Option is : D

~~**A.**~~ The class will fail to compile, since the class OtherClass is used before it is defined.

~~**B.**~~ There is no problem with the code.

~~**C.**~~ The class will fail to compile, since the class OtherClass must be defined in a file called OtherClass.java

> This is not needed because OtherClass is not public. The class & file name must match only if the class is public.

D. The class will fail to compile .

> The package declaration can never occur after an import statement.

~~**E.**~~ None of the above.

Explanation:

The order is:
package statement.
import statements
class/ interface definitions.
Important point to note here is YOU MUST READ THE QUESTIONS VERY CAREFULLY.

Back to Question without Answer

12. QID - 2.1056 : Java Basics

Which one of these is a proper definition of a class TestClass that cannot be sub-classed?

Correct Option is : A

A. `final class TestClass { }`

~~**B.**~~ `abstract class TestClass { }`

~~**C.**~~ `native class TestClass { }`

~~**D.**~~ `static class TestClass { }`

~~**E.**~~ `private class TestClass { }`

Explanation:

A final class cannot be subclassed.
Although declaring a method static usually implies that it is also final, this is not true for classes. An inner class can be declared static and still be extended.
Note that for classes, final means it cannot be extended, while for methods, final means it cannot be overridden in a subclass.
The native keyword can only be used on methods, not on classes and or variables.

Back to Question without Answer

13. QID - 2.1085 : Java Basics

Consider the following two classes defined in two java source files.

```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1 <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       X x = new X();
       x.apply(LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

Correct Options are : C D

~~**A.**~~ `import static X;`

~~**B.**~~ `import static com.foo.*;`

> Bad syntax. com.foo is a package and you cannot import a package statically. You can only import static members of a class statically.

C. `import static com.foo.X.*;`

This static import is required because of Y is accessing LOGICID directly without its class name (i.e. X.LOGICID).

D. `import com.foo.*;`

This is required because Y also accesses the class X : `X x = new X();` If Y had only one statement, `System.out.println(LOGICID); import static com.foo.X.*` would suffice.

E. `import com.foo.X.LOGICID;`

Syntax for importing static fields is: import static <package>.<classname>.*; or import static <package>.<classname>.<fieldname>;

Back to Question without Answer

14. QID - 2.1069 : Java Basics

Which method declarations will enable a class to be run as a standalone program?

Correct Options are : D E

~~**A.**~~ `static void main(String args[ ])`

> Surprisingly, it does work. Even if the class is defined in a package.

~~**B.**~~ `public void static main(String args[ ])`

> Remember, return type and method name are NEVER separated.

~~**C.**~~ `public static main(String[ ] argv)`

> There always has to be return type for a method. Only constructors don't have a return type.

D. `final public static void main(String [ ] array)`

> final only means that subclasses cannot shadow (in case of static methods) or override (in case of instance methods) it.

E. `public static void main(String args[ ])`

Explanation:

If you run the following program by changing the accessibility from `public` to `private` and `protected`, it may work on some versions of Java.

However, for the purpose of Java Certification exam, it should be assumed that for the

JVM to execute a class using the standard main method, the accessibility of the main method must be `public`.

```
package test;
public class TestClass{
    private static void main(String args[]){
         System.out.println("hello");
    }
}
```

Back to Question without Answer

15. QID - 2.1080 : Java Basics

Which of the following are valid declarations of the standard main() method?

Correct Options are : D E

A. ~~`static void main(String args[ ]) { }`~~

> Although practically correct but for the purpose of SCJP you should not select this option because the method is not declared public.

B. ~~`public static int main(String args[ ]) {}`~~

> This method returns an 'int' instead of 'void'.

C. ~~`public static void main (String args) { }`~~

> The method takes only one String instead of String[].

D. `final static public void main (String[ ] arguments ) { }`

E. `public static void main (String[ ] args) { }`

Explanation:

A valid declaration of the main() method must be 'public' and 'static', should return 'void', and should take a single array of Strings.
The order of the static and public keywords is irrelevant. But return type should always come just before the method name.
'final' does not change the method signature.
Also, in some of the JDK environments, even a private or protected main() method

may work however, for the purpose of Java Certification exam, it should be assumed that for the JVM to execute a class using the standard main method, the accessibility of the main method must be public.

Back to Question without Answer

16. QID - 2.1004 : Java Basics

Which of these statements concerning the use of modifiers are true?

Correct Option is : B

~~**A.**~~ By default (i.e. no modifier) the member is only accessible to classes in the same package and subclasses of the class.

> No, the member will be accessible only within the package.

B. You cannot specify visibility of local variables.

> They are always only accessible within the block in which they are declared.

~~**C.**~~ Local variable always have default accessibility.

> A local variable (aka automatic variable) means a variable declared in a method. They don't have any accessibility. They are accessible only from the block they are declared in.
> Remember, they are not initialized automatically. You have to initialize them explicitly.

~~**D.**~~ Local variables can be declared as private.

~~**E.**~~ Local variables can only be declared as public.

Explanation:

You cannot apply any modifier except final to a local variable. i.e. you cannot make them transient, volatile, static, public, and private.

But you can apply access modifiers (public private and protected) and final, transient, volatile, static to instance variables.
You cannot apply native and synchronized to any kind of variable.

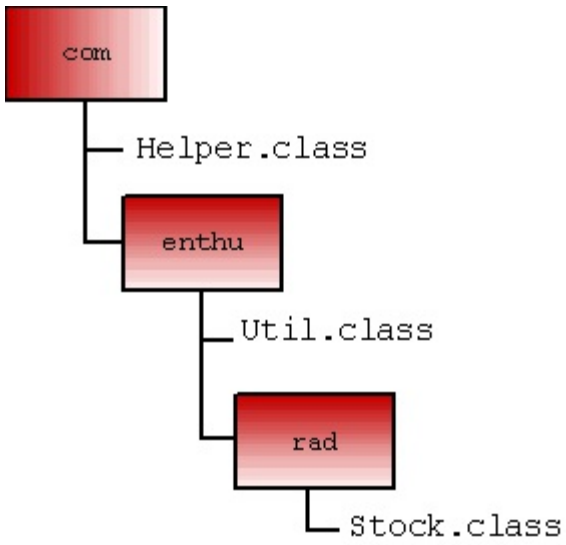
Back to Question without Answer

17. QID - 2.1066 : Java Basics

Consider the following directory structure shown in Image 1 that displays available folders and classes and the code given below.

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements must be added to the above class?



Correct Options are : D E

~~A.~~ `import com.enthu.*;`

This is not required because the code is not using any class from this package.

B. ~~`import com.*.*;`~~

> Bad Syntax. You can only import one package (i.e. all classes in that package) using a * or one class in an import statement.

C. ~~`import *.*.*;`~~

> Bad syntax.

D. `import com.*;`

> This is required because the code is using Helper.class, which exists in com package.

E. `import com.enthu.rad.*;`

> This is required because the code is using Stock.class, which exists in com.enthu.rad package.

F. ~~`import all;`~~

Explanation:

Since the given class does not have any package declaration, it belongs to the default package and therefore it must import com.Helper and com.enthu.rad.Stock classes.

Back to Question without Answer

18. QID - 2.1141 : Java Basics

Which line(s) in the following code will cause a compilation error?

```
static import java.lang.System.*; //1
class $$ //2
{
    static public void main(String... _$_) //3
   {
       String _ = ""; //4
       for(int $=0; ++$ < _$_.length; )  //5
           _ += _$_[$];  //6
       out.println(_);  //7
    }
}
```

Correct Option is : A

A. 1

> Read the question carefully. The order of static and import is invalid. It should have been "import static". Everything else is legal in the code!

~~B.~~ 2

~~C.~~ 3

~~D.~~ 4

~~E.~~ 5

~~F.~~ 6

~~G.~~ 7

~~H.~~ None of the lines is invalid.

Back to Question without Answer

19. QID - 2.1163 : Java Basics

The following is a valid member variable declaration:

```
private static final transient int i = 20;
```

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

Although it does not make sense to make static variable transient as static variables are not serialized anyway, it is valid to do so.
You can apply all the modifiers to member variables except `abstract, native` and `synchronized`.
For methods, you cannot apply `transient` and `volatile`.

Back to Question without Answer

20. QID - 2.1364 : Java Basics

Given the following class, which of these are valid ways of referring to the class from outside of the package com.enthu?

```
package com.enthu;
public class Base{
   // ....
   // lot of code...
}
```

Correct Options are : D E

~~**A.**~~ Base

> Only if you import the whole package containing the class or import the class first.

~~**B.**~~ By importing the package `com.*` and referring to the class as `enthu.Base`

> package 'com' does not contain Base.

~~**C.**~~ importing `com.*` is illegal.

> It is perfectly legal.

D. By importing `com.enthu.*` and referring to the class as `Base`.

E. By referring to the class as `com.enthu.Base`.

Explanation:

A class or interface can be referred to by using its fully qualified name or its simple name.
Using the fully qualified name will always work, but to use the simple name either the class must be in the same package or it has to be imported.
By importing `com.enthu.*` all the classes from the package will be imported and can be referred to using simple names.
Importing `com.*` will not import the subpackage `enthu`. It will only import the classes in package `com`.

Back to Question without Answer

21. QID - 2.845 : Java Basics

The options below contain the complete contents of a file (the name of the file is not specified).
Which of these options can be run with the following command line once compiled?

```
java main
```

Correct Option is : D

~~A.~~
```
//in file main.java
class main {
   public void main(String[] args) {
       System.out.println("hello");
   }
}
```

The main method should be static.

~~B.~~
```
//in file main.java
   public static void main4(String[] args) {
       System.out.println("hello");
   }
```

You cannot have a method on its own. It must be a part of a class.

~~C.~~
```
//in file main.java
public class anotherone{
}
class main {
   public static void main(String[] args) {
       System.out.println("hello");
   }
}
```

A public class must exist in a file by the same name. So this code is invalid

because anotherone is a public class but the name of the file is main. It would have been valid if the name of the file were anotherone.java.

A non public class may exist in any file. This implies that there can be only one public class in a file.

D.
```
//in file main.java
class anothermain{
   public static void main(String[] args) {
       System.out.println("hello2");
   }
}
class main {
   public final static void main(String[] args) {
       System.out.println("hello");
   }
}
```

class main's main method will be executed. final is a valid modifier for the standard main method.

Note that final means a method cannot be overridden. Although static methods can never be overridden. (they can be shadowed), making a static method final prevents the subclass from implementing the same static method.

Explanation:

Observe that the given code does not follow the standard Java naming convention. The class names should start with a capital letter.

There are questions in the exam that contain similar non-conventional and confusing names and that is why we have kept a few questions like that in this question bank.

Back to Question without Answer

22. QID - 2.976 : Java Basics

Consider the following two java files:

```
//in file SM.java
package x.y;
public class SM{
    public static void foo(){ };
}


//in file TestClass.java
//insert import statement here //1
public class TestClass{
   public static void main(String[] args){
      foo();
   }
}
```

What should be inserted at //1 so that TestClass will compile and run?

Correct Options are : C E

~~**A.**~~ `import static x.y.*;`

> x.y.* means all the classes in package x.y. Classes cannot be imported using "import static". You must do import x.y.* for importing class.
> Further, importing a class will not give you a direct access to the members of the class. You will need to do SM.foo(), if you import SM.

~~**B.**~~ `import static x.y.SM;`

> x.y.SM means you are trying to import class x.y.SM. A class cannot be imported statically.

C. `import static x.y.SM.foo;`

This is valid because this statement is importing the static member foo of class x.y.SM.

D. ~~`import static x.y.SM.foo();`~~

Even though foo is a method, you cannot put () in the import statement.

E. `import static x.y.SM.*;`

This is valid because this statement is importing all the static members of class x.y.SM.

Back to Question without Answer

23. QID - 2.1272 : Java Basics

Which of these are not legal declarations within a class?

Correct Option is : C

~~**A.**~~ `static volatile int sa ;`

~~**B.**~~ `final Object[ ] objArr = { null } ;`

Declares and defines an array of Objects of length 1.

C. `abstract int t ;`

Variables can't be declared abstract and native.

~~**D.**~~ `native void format( ) ;`

~~**E.**~~ `final transient static private double PI = 3.14159265358979323846 ;`

Explanation:

`static` and `final` are valid modifiers for both member field and method declarations within a class.
`transient` and `volatile` modifiers are only valid for member field declarations.
`abstract` and `native` are only valid for member methods.

Note: a class declaration can have only have `final`, `abstract` and `public` as modifiers, unless it is a nested class, in which case, it can be declared `private` or `protected` as well.

Within a method, a local variable may be declared as `final`.

Back to Question without Answer

24. QID - 2.1132 : Java Basics

Which of the following is/are illegal Java identifier(s)?

Correct Option is : C

~~**A.**~~ num

~~**B.**~~ int123

Can contain digits but cannot start with a digit.

C. 2Next

Cannot start with a digit.

~~**D.**~~ _interface

_ and $ are valid at any position.

~~**E.**~~ a$_123

_ and $ are valid at any position.

Explanation:

A valid java identifier is composed of a sequence of java letters and digits, the first of which must be a letter.
(According to JLS.)
It is not clear from the objectives whether this topic is included or not. Regardless, it is good to know how to declare a valid identifier.

Back to Question without Answer

25. QID - 2.1118 : Java Basics

Which of the changes given in options can be done (independent of each other) to let the following code compile and run without errors?

```
class SomeClass{
   String s1 = "green mile";   // 0
   public void generateReport( int n ){
      String local;   // 1
      if( n > 0 ) local = "good";   //2
      System.out.println( s1+" = " + local );   //3
   }
}
```

Correct Options are : A C

A. Insert after line 2 : `else local = "bad";`

~~**B.**~~ Insert after line 2 : `if(n <= 0) local = "bad";`

C. Move line 1 and place it after line 0.

~~**D.**~~ change line 1 to : `final String local = "rocky";`

> Making it final will not let //2 compile as it would then try to modify a final variable.

~~**E.**~~ The program already is without any errors.

Explanation:

The problem is that `local` is declared inside a method is therefore local to that

method. It is called a local variable (also known as automatic variable) and it cannot be used before initialized. Further, it will not be initialized unless you initialize it explicitly because local variables are not initialized by the JVM on its own. The compiler spots the usage of such uninitialized variables and ends with an error message.

1. Making it a member variable (choice "Move line 1 and place it after line 0.") will initialize it to `null`.
2. Putting an else part (choice "Insert after line 2 : `else local = "bad";`") will ensure that it is initialized to either '`good`' or '`bad`'. So this also works.
Choice "Insert after line 2 : `if(n <= 0) local = "bad";`" doesn't work because the second `'if'` will actually be another statement and is not considered as a part of first `'if'`. So, compiler doesn't realize that '`local`' will be initialized even though it does get initialized.

Back to Question without Answer

26. QID - 2.944 : Java Basics

Which of the following are valid declarations:

Correct Options are : B C D

~~**A.** int a = b = c = 100;~~

B. int a, b, c; a = b = c = 100;

C. int a, b, c=100;

D. int a=100, b, c;

~~**E.** int a= 100 = b = c;~~

Explanation:

Java does not allow chained initialization in declaration so option 1 and 5 are not valid.

Back to Question without Answer

27. QID - 2.1007 : Java Basics

How can you declare a method someMethod() such that an instance of the class is not needed to access it and all the members of the same package have access to it.

Correct Options are : A B C

A. `public static void someMethod()`

B. `static void someMethod()`

C. `protected static void someMethod()`

D. `void someMethod()`

E. `protected void someMethod()`

F. `public abstract static void someMethod()`

static methods can't be abstract.

Explanation:

Since the question says, "...an instance of the class is not needed...", the method has to be static.
Also, as the question does not say that other packages should not have access to the method so public or protected is also correct.

Back to Question without Answer

28. QID - 2.1142 : Java Basics

Which of the given options should be inserted at line 1 so that the following code can compile without any errors?

```
package objective1;
// 1
public class StaticImports{

    public StaticImports(){
    out.println(MAX_VALUE);
    }

}
```

(Assume that java.lang.Integer has a static field named MAX_VALUE)

Correct Options are : A D

A. `import static java.lang.Integer.*;`

~~B.~~ `static import java.lang.System.out;`

~~C.~~ `static import Integer.MAX_VALUE;`

D. `import static java.lang.System.*;`

~~E.~~ `static import java.lang.System.*;`

Explanation:

The order of keywords for a static import must be "`import static ...` ".
You can either import all the static member using `import static java.lang.Integer.*` or one specific member using `import static java.lang.Integer.MAX_VALUE;`
You must specify the full package name of the class that you are importing (just like the regular import statement). So, `import static Integer.*;` is wrong.

Back to Question without Answer

29. QID - 2.1333 : Java Basics

Which of the following are not legal Java identifiers?

Correct Option is : A

A. goto

> Even though it is not used anywhere in Java code, it has been kept as a reserved word to prevent its usage in any form.

~~**B.**~~ unsigned

> It is not a reserved word or keyword.

~~**C.**~~ String

> Yes, it is valid. This is a valid statement: String String = "String";

~~**D.**~~ _xyz

> An identifier can start with a _ sign or a $ sign.

~~**E.**~~ $_abc

> Can have _ or $.

~~**F.**~~ iLikeVeryVeryVeryVeryVeryLongIdentifiersThatDontMakeAnySenseAtAll (65 characters)

> There is no restriction on the length of an identifier.

Explanation:

A valid java identifier is composed of a sequence of java letters and digits, the first of which must be a letter. It cannot be same as any java Keywords or literals (i.e. true, false or null). Java letters include uppercase and lowercase ASCII latin letters and _ , $. (According to JLS.) . In fact, class names can serve as a valid identifier as in 'String' in one of the options above.

Back to Question without Answer

30. QID - 2.1197 : Java Basics

Which of the following are keywords in Java?

Correct Options are : A D E F

A. default

~~**B.**~~ NULL

null (all lowercase) is a keyword.

~~**C.**~~ String

It is a Java class.

D. throws

E. long

F. strictfp

Back to Question without Answer

31. QID - 2.1360 : Java Basics

Given the following code, which statements can be placed at the indicated position without causing compile and run time errors?

```
public class Test{
   int i1;
   static int i2;
   public void method1(){
      int i;
      // ... insert statements here
   }
}
```

Correct Options are : A B E

A. `i = this.i1;`

> As i1 is an instance variable, it is accessible through 'this'.

B. `i = this.i2;`

> Although 'this' is not needed to access `i2`, it is not an error to do so.

~~C.~~ `this = new Test( );`

> Nope, you can't change `this`.

~~D.~~ `this.i = 4;`

> You cannot do `this.i` as `i` is a local variable.

E. `this.i1 = i2;`

You are just assigning a static field's value to non-static field.

Back to Question without Answer

32. QID - 2.1020 : Java Basics

What does the zeroth element of the string array passed to the standard main method contain?

Correct Option is : D

~~**A.**~~ The name of the class.

~~**B.**~~ The string "java".

~~**C.**~~ The number of arguments.

D. The first argument of the argument list, if present.

~~**E.**~~ None of the above.

Explanation:

Note that if no argument is passed the args parameter is NOT null but a valid non-null String array of length zero.

Back to Question without Answer

33. QID - 2.940 : Java Basics

Which of the statements regarding the following code are correct?

```
public class TestClass{
   static int a;
   int b;
   public TestClass(){
      int c;
      c = a;
      a++;
      b += c;
   }
   public static void main(String args[]) {   new TestClass();   }
}
```

Correct Option is : E

~~**A.**~~ The code will fail to compile because the constructor is trying to access static members.

> A constructor (or any other method) can access static members.

~~**B.**~~ The code will fail to compile because the constructor is trying to use static member variable a before it has been initialized.

> static fields are always initialized automatically if you do not initialize them explicitly. So are instance fields.

~~**C.**~~ The code will fail to compile because the constructor is trying to use member variable b before it has been initialized.

~~**D.**~~ The code will fail to compile because the constructor is trying to use local variable

c before it has been initialized.

```
c is getting initialized at line 2: c = a;
```

E. The code will compile and run without any problem.

Explanation:

All the instance or static variables are given a default values if not explicitly initialized. All numeric variable are given a value of zero or equivalent to zero (i.e. 0.0 for double or float).
booleans are initialized to false and objects references are initialized to null.

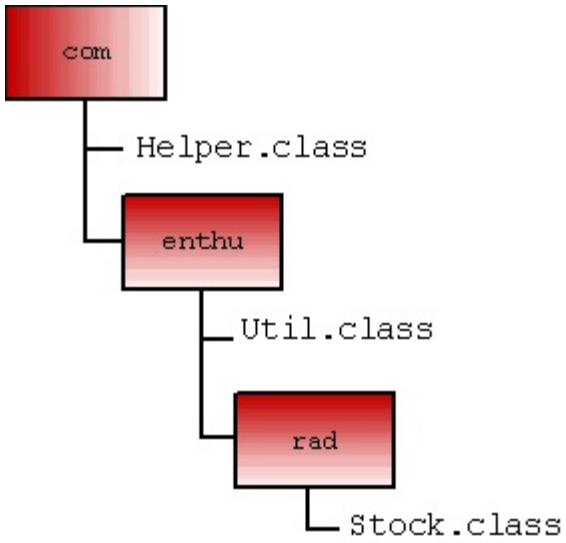
Back to Question without Answer

34. QID - 2.1063 : Java Basics

Consider the directory structure shown in Image 1 that displays available folders and classes and the code given below:

```
class StockQuote{
  Stock stock;
  public StockQuote(Stock s)  {
  }
  public void store() throws IOException{
    return Util.store(stock);
  }
  public double computePrice(){
    return Helper.getPricer(stock).price();
  }
}
```

Assuming that the code uses valid method calls, what statements MUST be added to the above class?



Correct Options are : B C D E

~~A.~~ `package com.enthu.rad.*;`

Bad syntax. A package statement can never have a *. It should specify the exact package name.

B. `import com.enthu.*;`

C. `package com.enthu.rad;`

Since there is no import statement available for com.enthu.rad package, you must put the given class in com.enthu.rad package so that it will be accessible. Classes of the same package are always available to each other.

D. `import com.*;`

E. `import java.io.*;`

F. It is not required to import java.io.* or import java.io.IOException because java.io package is imported automatically.

Since the code is using IOException, the java.io package (or just java.io.IOException class) must be imported. Only java.lang package is imported automatically.

Back to Question without Answer

35. QID - 2.1297 : Java Basics

Which of the following are valid identifiers?

Correct Options are : B C

~~**A.**~~ class

> It is a keyword.

B. $value$

> $ and _ are allowed at any place. Numerals (0 - 9) are also allowed but not at the first place.

C. angstrom

~~**D.**~~ 2much

> Cannot start with a number. But much2 is valid.

~~**E.**~~ zer@

> No special chars except $ and _ are allowed.

Explanation:

As per JLS section 3.8: A valid identifier must start with a Java letter followed by a Java letter or a digit.

A "Java letter" is a character for which the method Character.isJavaIdentifierStart(int) returns true.

A "Java letter-or-digit" is a character for which the method Character.isJavaIdentifierPart(int) returns true.

The "Java letters" include uppercase and lowercase ASCII Latin letters A-Z (\u0041-\u005a), and a-z (\u0061-\u007a), and, for historical reasons, the ASCII underscore (_, or \u005f) and dollar sign ($, or \u0024). The $ character should be used only in mechanically generated source code or, rarely, to access pre-existing names on legacy systems.

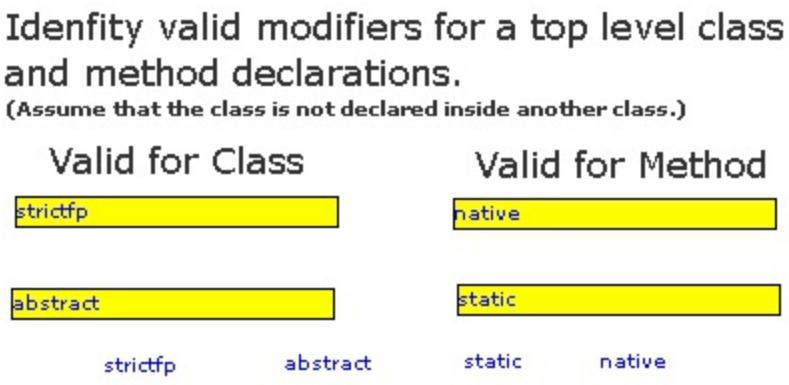
The "Java digits" include the ASCII digits 0-9 (\u0030-\u0039).

Letters and digits may be drawn from the entire Unicode character set, which supports most writing scripts in use in the world today, including the large sets for Chinese, Japanese, and Korean. This allows programmers to use identifiers in their programs that are written in their native languages.

An identifier cannot have the same spelling (Unicode character sequence) as a keyword (§3.9), boolean literal (§3.10.3), or the null literal (§3.10.7), or a compile-time error occurs.

Back to Question without Answer

36. QID - 2.1139 : Java Basics



Explanation:

Only methods can be 'native'.
A top level class (i.e. a class that is not nested inside any other class) cannot be declared as static.
Both - a class or a method can be declared as strictfp.

Back to Question without Answer

37. QID - 2.937 : Java Basics

Given the following class, which statements can be inserted at line 1 without causing the code to fail compilation?

```
public class TestClass{
   int a;
   int b = 0;
   static int c;
   public void m(){
      int d;
      int e = 0;
      // Line 1
   }
}
```

Correct Options are : A B C E

A. `a++;`

> Here, 'a' is an instance variable of type int. Therefore, it will be given a default value of Zero and so it need not be initialized explicitly.

B. `b++;`

C. `c++;`

> Here 'c' is a class variable (also called as static variable) of type int so it will be given a default value of Zero and so it need not be initialized explicitly.

~~D.~~ `d++;`

This will not compile because 'd' is not initialized. Note that automatic variables (also called as method local variables i.e. variables declared within a method) have to be explicitly initialized.

E. `e++;`

Explanation:

All the instance or static variables are given a default values if they are not explicitly initialized. All numeric variable are given a value of zero or equivalent to zero (i.e. 0 for integral types and 0.0 for double/float). Booleans are initialized to false and objects are initialized to null.

Back to Question without Answer

38. QID - 2.1298 : Java Basics

Which of the following are valid declarations within a class?

Correct Options are : A E

A. `volatile int k;`

~~B.~~ `abstract boolean bool;`

fields cannot be abstract.

~~C.~~ `native float radix;`

variables cannot be native.

~~D.~~ `friendly int ArrayList;`

Here friendly is being used as if it were a keyword, but friendly is not a valid keyword in Java.

E. `int friendly, ArrayList;`

Here, friendly and ArrayList are variable names. Note that any class name can be used as a variable name.

Back to Question without Answer

39. QID - 2.1162 : Java Basics

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){}
}
```

Correct Option is : C

~~**A.**~~ The class `TestClass` cannot be declared `abstract`.

> Any class can be declared `abstract`.

~~**B.**~~ The variable `j` cannot be declared `transient`.

C. The variable `k` cannot be declared `synchronized`.

> Variables cannot be declared `synchronized`. Only methods can be declared `synchronized`.

~~**D.**~~ The constructor `TestClass( )` cannot be declared final.

> It is not a constructor, it is a simple method. Notice `void` return type.

~~**E.**~~ The method `f( )` cannot be declared static.

~~**F.**~~ This code has no errors.

Back to Question without Answer

40. QID - 2.1258 : Java Basics

Consider the following code:

```
class Test{
  public static void main(String[] args){
    for (int i = 0; i < args.length; i++)   System.out.print(i == 0
? args[i] : " " + args[i]);
  }
}
```

What will be the output when it is run using the following command:

java Test good bye friend!

Correct Option is : A

A. good bye friend!

~~**B.**~~ good good good

~~**C.**~~ goodgoodgood

~~**D.**~~ good bye

~~**E.**~~ None of the above.

Explanation:

The arguments passed on the command line can be accessed using the args array. The first argument (i.e. good) is stored in args[0], second argument (i.e. bye) is stored in

args[1] and so on.

Here, we are passing 3 arguments. Therefore, args.length is 3 and the for loop will run 3 times. For the first iteration, i is 0 and so the first operand of the ternary operator (?) will be returned, which is args[i]. For the next two iterations, " "+args[i] will be returned. Hence, the program will print three strings: "good", " bye", and " friend" on the same line.

Notice that unlike in C++, program name is not the first parameter in the argument list. Java does not need to know the program name because the .class file name and the java class name are always same (for a public class). So the java code always knows the program name it is running in. So there is no need to pass the program name as the first parameter of the argument list.
Note that in C/C++, the binary file name may be anything so the code does not know what binary file it is going to end up in. That's why the program name is also sent (automatically) in parameter list.

Back to Question without Answer

41. QID - 2.1249 : Java Basics

Consider the following class:

```
public class ArgsPrinter{
   public static void main(String args){
      for(int i=0; i<3; i++){
         System.out.print(args+" ");
      }
   }
}
```

What will be printed when the above class is run using the following command line:
`java ArgsPrinter 1 2 3 4`

Correct Option is : E

~~**A.**~~ `1 2 3`

~~**B.**~~ `ArgsPrinter 1 2`

~~**C.**~~ `java ArgsPrinter 1 2`

~~**D.**~~ `1 1 1`

E. None of these.

Explanation:

To run a class from the command line, you need a `main(String[] )` method that takes an array of Strings array not just a String. Therefore, an exception will be thrown

at runtime saying no `main(String[] )` method found.

Back to Question without Answer

42. QID - 2.1338 : Java Basics

What is the correct parameter specification for the standard main method?

Correct Options are : B E

~~**A.** void~~

B. String[ ] args

~~**C.** Strings args[ ]~~

~~**D.** String args~~

E. String args[ ]

Explanation:

There is a no difference for `args` whether it is defined as `String[] args` or `String args[]`. However, there is an important difference in the way it is defined as illustrated by the following:

1. `String[] sa1, sa2;`
Here, both - `sa1` and `sa2` are String arrays.

2. `String sa1[], sa2;`
Here, only `sa1` is a String array. `sa2` is just a String.

Back to Question without Answer

43. QID - 2.1110 : Java Basics

Which of these statements are true?

Correct Options are : B C

~~**A.**~~ A static method can call other non-static methods in the same class by using the 'this' keyword.

> 'this' reference is not available within a static method.

B. A class may contain both static and non-static variables and both static and non-static methods.

C. Each object of a class has its own copy of each non-static member variable.

~~**D.**~~ Instance methods may access local variables of static methods.

> local variables can only be accessed in the method they are defined. So you cannot access them anywhere outside that method.

~~**E.**~~ All methods in a class are implicitly passed a 'this' parameter when called.

> All non-static/instance methods in a class are implicitly passed a 'this' parameter when called.

Explanation:

The keyword 'this' can only be used within non-static methods. static methods cannot access non static fields or methods.

Note : you can't do something like
this = new Object();

Back to Question without Answer

44. QID - 2.955 : Java Basics

An instance member ...

Correct Options are : A D

A. can be a variable, a constant or a method.

~~**B.**~~ is a variable or a constant.

~~**C.**~~ belongs to the class.

D. belongs to an instance of the class.

~~**E.**~~ is same as a local variable.

> variables defined in methods are called local variables (also known as automatic variables) where as instance members are defined in the class scope.

Explanation:

An instance member belongs to a single instance, not the class as a whole. An instance member is a member variable or a member method that belongs to a specific object instance. All non-static members are instance members.

Back to Question without Answer

45. QID - 2.1242 : Java Basics

Given:

```
public class TestClass{
     public static void main(String[] args){
     int i = Integer.parseInt(args[1]);
     System.out.println(args[i]);
   }
}
```

What will happen when you compile and run the above program using the following command line:

```
java TestClass 1 2
```

Correct Option is : D

~~**A.**~~ It will print `1`

~~**B.**~~ It will print `2`

~~**C.**~~ It will print some junk value.

D. It will throw `ArrayIndexOutOfBoundsException`.

~~**E.**~~ It will throw NumberFormatException

Explanation:

1. Arrays are indexed from 0.
2. In java, the name of the class is not the first element of args.

So, when the command line is : `java TestClass 1 2`, `args[0]` is 1 and `args[1]` is

2.

When you try to access `args[2]`, It will throw an `ArrayIndexOutofBoundsException` because the array length is only 2 so `args[2]` is out of bounds.

Back to Question without Answer

46. QID - 2.1234 : Java Basics

Consider the following code:

```
public abstract class TestClass{
    public abstract void m1();
    public abstract void m2(){
        System.out.println("hello");
    }
}
```

Which of the following corrections can be applied to the above code (independently) so that it compiles without any error?

Correct Options are : A C

A. Replace the method body of m2() with a ; (semi-colon).

~~**B.**~~ Replace the ; at the end of m1() with a method body.

C. Remove abstract from m2().

> A method that has a body cannot be abstract. In other words, an abstract method cannot have a body. So either remove the method body (as in m1()) or remove abstract keyword.

~~**D.**~~ Remove abstract from the class declaration.

Back to Question without Answer

47. QID - 2.894 : Java Basics

The following are the complete contents of TestClass.java file. Which packages are automatically imported?

```
class TestClass{
   public static void main(String[] args){
     System.out.println("hello");
   }
}
```

Correct Options are : C F

~~**A.**~~ java.util

~~**B.**~~ System

System is not a package. It is a class in java.lang package.

C. java.lang

~~**D.**~~ java.io

~~**E.**~~ String

String is a class in java.lang package.

F. The package with no name.

If there is no package statement in the source file, the class is assumed to be created in a default package that has no name.

Back to Question without Answer

48. QID - 2.1170 : Java Basics

Consider the classes shown below:

```
class A{
   public A() { }
   public A(int i) {   System.out.println(i );    }
}
class B{
   static A s1 = new A(1);
   A a = new A(2);
   public static void main(String[] args){
      B b = new B();
      A a = new A(3);
   }
   static A s2 = new A(4);
}
```

Which is the correct sequence of the digits that will be printed when B is run?

Correct Option is : B

~~**A.**~~ 1 ,2 ,3 4.

B. 1 ,4, 2 ,3

~~**C.**~~ 3, 1, 2, 4

~~**D.**~~ 2, 1, 4, 3

~~**E.**~~ 2, 3, 1, 4

Explanation:

The order of initialization of a class is:
1. All static constants, variables and blocks.(Among themselves the order is the order in which they appear in the code.)
2. All non static constants, variables and blocks.(Among themselves the order is the order in which they appear in the code.)
3. Constructor.

Back to Question without Answer

49. QID - 2.915 : Java Basics

Given the following contents of two java source files:

```
package util.log4j;
public class Logger  {
  public void log(String msg){
      System.out.println(msg);
  }
}
```

and

```
package util;
public class TestClass {
    public static void main(String[] args) throws Exception {
        Logger logger = new Logger();
        logger.log("hello");
    }
}
```

What changes, when made independently, will enable the code to compile and run?

Correct Options are : B C

~~**A.**~~ Replace `Logger logger = new Logger();` with:

```
log4j.Logger logger = new log4j.Logger();
```

> If you are not importing a class or the package of the class, you need to use the class's fully qualified name while using it. Here, you need to use `util.log4j.Logger` instead of just `log4j.Logger`:
>
> ```
> util.log4j.Logger logger = new util.log4j.Logger();
> ```

B. Replace package `util.log4j;` with

```
package util;
```

This will put both the classes in the same package and TestClass can then directly use Logger class without importing anything.

C. Replace `Logger logger = new Logger();` with:
`util.log4j.Logger logger = new util.log4j.Logger();`

Using a fully qualified class name always works irrespective of whether you import the package or not. In this case, you are importing package util but Logger is in util.log4j. Therefore, the use of fully qualified class name for Logger, which is util.log4j.Logger, makes it work.

~~**D.**~~ Remove package `util.log4j;` from Logger.

Remember that you can never access a class that is defined in the default package (i.e. the package with no name) from a class in any other package. So if you remove the package statement from Logger, you can't access it from util package, which is where TestClass is.

~~**E.**~~ Add `import log4j;` to TestClass.

This will not help because Logger is in `util.log4j` package and not in `log4j` package.

Back to Question without Answer

Java Basics - Garbage Collection

Exam Objectives -

Although not mentioned explicitly in the exam objectives, there are questions in the exam on this topic.

01. QID - 2.884

Consider the following code snippet:

```
public class Test{
  void test(){
      MyClass obj = new MyClass();
      obj.name = "jack";
      // 1 insert code here
  }
}

//In MyClass.java
public class MyClass{
  int value;
  String name;
}
```

What can be inserted at // 1, which will make the object referred to by obj eligible for garbage collection?

Select 1 option

A. `obj.destroy();`

B. `Runtime.getRuntime().gc();`

C. `obj = null;`

D. `obj.finalize()`

E. `obj.name = null; as well as obj = null;`

Check Answer

02. QID - 2.920

Which is the earliest line in the following code after which the object created on line // 1 can be garbage collected, assuming no compiler optimizations are done?

```
public class NewClass{
   private Object o;
   void doSomething(Object s){  o = s;   }

   public static void main(String args[]){
      Object obj = new Object(); // 1
      NewClass tc = new NewClass(); //2
      tc.doSomething(obj); //3
      obj = new Object();    //4
      obj = null;    //5
      tc.doSomething(obj); //6
   }
}
```

Select 1 option

A. Line 1

B. Line 2

C. Line 3

D. Line 4

E. Line 5

F. Line 6

Check Answer

03. QID - 2.921

Consider the following code:

```
class MyClass { }
public class TestClass{
   MyClass getMyClassObject(){
      MyClass mc = new MyClass(); //1
         return mc; //2
   }
   public static void main(String[] args){
      TestClass tc = new TestClass(); //3
      MyClass x = tc.getMyClassObject(); //4
      System.out.println("got myclass object"); //5
      x = new MyClass(); //6
      System.out.println("done"); //7
   }
}
```

After what line the MyClass object created at line 1 will be eligible for garbage collection?

Select 1 option

A. 2

B. 5

C. 6

D. 7

E. Never till the program ends.

Check Answer

04. QID - 2.922

In the following code, after which statement (earliest), the object originally held in s, may be garbage collected ?

```
1. public class TestClass{
2.   public static void main (String args[]){
3.      Student s = new Student("Vaishali", "930012");
4.      s.grade();
5.      System.out.println(s.getName());
6.      s = null;
7.      s = new Student("Vaishali", "930012");
8.      s.grade();
9.      System.out.println(s.getName());
10      s = null;
     }
   }

public class Student{
   private String name, rollNumber;

   public Student(String name, String rollNumber) {
      this.name = name;
      this.rollNumber = rollNumber;
   }

   //valid setter and getter for name and rollNumber follow

   public void grade() {
   }

}
```

Select 1 option

A. It will not be Garbage Collected till the end of the program.

B. Line 5

C. Line 6

D. Line 7

E. Line 10

Check Answer

05. QID - 2.923

Given the following code:

```
class M { }
class N{
   private M m = new M();
   public void makeItNull(M pM){
      pM = null;
   }
   public void makeThisNull(){
      makeItNull(m);
   }
   public static void main(String[] args){
      N n = new N();
      n.makeThisNull();
   }
}
```

Which of the following statements are correct?

Select 1 option

A. There are no instances of M to be garbage collected till the program ends.

B. A call to `n.makeThisNull()` marks the private instance of M for garbage collection.

C. Setting `pM = null;` in `makeItNull()`, marks the private instance of M for garbage collection.

D. `private` members of a class become eligible for garbage collection only when the instance of the class itself becomes eligible for garbage collection.

Check Answer

Java Basics - Garbage Collection (Answered)

01. QID - 2.884 : Java Basics - Garbage Collection

Consider the following code snippet:

```
public class Test{
  void test(){
      MyClass obj = new MyClass();
      obj.name = "jack";
      // 1 insert code here
  }
}

//In MyClass.java
public class MyClass{
  int value;
  String name;
}
```

What can be inserted at // 1, which will make the object referred to by obj eligible for garbage collection?

Correct Option is : C

~~**A.**~~ `obj.destroy();`

~~**B.**~~ `Runtime.getRuntime().gc();`

> Calling System.gc() or Runtime.getRuntime().gc() will not necessarily run the garbage collector. It only requests the JVM to perform garbage collection but there is no guarantee that the JVM will do it.
>
> By the way, System.gc() is equivalent to Runtime.getRuntime().gc().

C. `obj = null;`

This will make the object eligible for GC because there are no other references to it.

~~**D.**~~ `obj.finalize()`

~~**E.**~~ `obj.name = null; as well as obj = null;`

You don't need to do obj.name=null;

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

Nothing can ensure that an object will definitely be destroyed by the garbage collector. You can at most make an object eligible for GC by making sure that there are no references to it.

Back to Question without Answer

02. QID - 2.920 : Java Basics - Garbage Collection

Which is the earliest line in the following code after which the object created on line // 1 can be garbage collected, assuming no compiler optimizations are done?

```
public class NewClass{
   private Object o;
   void doSomething(Object s){  o = s;   }

   public static void main(String args[]){
      Object obj = new Object(); // 1
      NewClass tc = new NewClass(); //2
      tc.doSomething(obj); //3
      obj = new Object();    //4
      obj = null;    //5
      tc.doSomething(obj); //6
   }
}
```

Correct Option is : F

~~**A.**~~ Line 1

~~**B.**~~ Line 2

~~**C.**~~ Line 3

~~**D.**~~ Line 4

~~**E.**~~ Line 5

F. Line 6

Before this line the object is being pointed to by at least one variable.

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling `System.gc();`

Back to Question without Answer

03. QID - 2.921 : Java Basics - Garbage Collection

Consider the following code:

```
class MyClass { }
public class TestClass{
   MyClass getMyClassObject(){
      MyClass mc = new MyClass(); //1
         return mc; //2
   }
   public static void main(String[] args){
      TestClass tc = new TestClass(); //3
      MyClass x = tc.getMyClassObject(); //4
      System.out.println("got myclass object"); //5
      x = new MyClass(); //6
      System.out.println("done"); //7
   }
}
```

After what line the MyClass object created at line 1 will be eligible for garbage collection?

Correct Option is : C

~~**A.**~~ 2

~~**B.**~~ 5

C. 6

> At line 6, x starts pointing to a new MyClassObject and no reference to the original MyClass object is left.

~~**D.**~~ 7

~~**E.**~~ Never till the program ends.

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

Back to Question without Answer

04. QID - 2.922 : Java Basics - Garbage Collection

In the following code, after which statement (earliest), the object originally held in s, may be garbage collected ?

```
1. public class TestClass{
2.   public static void main (String args[]){
3.      Student s = new Student("Vaishali", "930012");
4.      s.grade();
5.      System.out.println(s.getName());
6.      s = null;
7.      s = new Student("Vaishali", "930012");
8.      s.grade();
9.      System.out.println(s.getName());
10      s = null;
     }
   }

public class Student{
   private String name, rollNumber;

   public Student(String name, String rollNumber) {
      this.name = name;
      this.rollNumber = rollNumber;
   }

   //valid setter and getter for name and rollNumber follow

   public void grade() {
   }

}
```

Correct Option is : C

~~A.~~ It will not be Garbage Collected till the end of the program.

~~**B.**~~ Line 5

C. Line 6

~~**D.**~~ Line 7

~~**E.**~~ Line 10

Explanation:

In this case, since there is only one reference to Student object, as soon as it is set to null, the object held by the reference is eligible for GC, here it is done at line 6.
Note that although an object is created at line 7 with same parameters, it is a different object and it will be eligible for GC after line 10.

Back to Question without Answer

05. QID - 2.923 : Java Basics - Garbage Collection

Given the following code:

```
class M { }
class N{
   private M m = new M();
   public void makeItNull(M pM){
      pM = null;
   }
   public void makeThisNull(){
      makeItNull(m);
   }
   public static void main(String[] args){
      N n = new N();
      n.makeThisNull();
   }
}
```

Which of the following statements are correct?

Correct Option is : A

A. There are no instances of M to be garbage collected till the program ends.

~~**B.**~~ A call to `n.makeThisNull()` marks the private instance of M for garbage collection.

~~**C.**~~ Setting `pM = null;` in `makeItNull()`, marks the private instance of M for garbage collection.

> `pM` is just a method parameter (a copy of the original reference) that is passed to `makeItNull()`. So setting it to null will not affect the original variable.

~~**D.**~~ `private` members of a class become eligible for garbage collection only when the instance of the class itself becomes eligible for garbage collection.

This is not true. Any instance can be made eligible by setting all its references to `null`. For example, in the following code, the Object instance referred to by 'o', can be made eligible for garbage collection by calling `setNull()`, even if the instance of X itself is not eligible for garbage collection.

```
class X{
  Object o = new Object();
  public void setNull(){ o = null; }
}
```

On the other hand, if the container object becomes eligible for GC and if there are no references to the contained objects outside of the container, the contained objects also become eligible for GC. For example, in the following code, both - the instance of X and the object instance contained inside X, will become eligible for garbage collection:

```
...
X x = new X();
x = null;
...
```

Explanation:

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.
2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

Back to Question without Answer

Overloading methods

Exam Objectives -

Create an overloaded method

01. QID - 2.1100

Consider the following class...

```
class TestClass{
    void probe(int... x) { System.out.println("In ..."); }  //1

    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(long x) { System.out.println("In long"); } //3

    void probe(Long x) { System.out.println("In LONG"); } //4

    public static void main(String[] args){
        Integer a = 4; new TestClass().probe(a); //5
        int b = 4; new TestClass().probe(b); //6
    }
}
```

What will it print when compiled and run?

Select 2 options

A. In Integer and In long

B. In ... and In LONG, if //2 and //3 are commented out.

C. In Integer and In ..., if //4 is commented out.

D. It will not compile, if //1, //2, and //3 are commented out.

E. In LONG and In long, if //1 and //2 are commented out.

Check Answer

02. QID - 2.939

Which of the following are true regarding overloading of a method?

Select 1 option

A. An overloading method must have a different parameter list and same return type as that of the overloaded method.

B. If there is another method with the same name but with a different number of arguments in a class then that method can be called as overloaded.

C. If there is another method with the same name and same number and type of arguments but with a different return type in a class then that method can be called as overloaded.

D. An overloaded method means a method with the same name and same number and type of arguments exists in the super class and sub class.

Check Answer

03. QID - 2.924

Consider the following class...

```
class TestClass{
    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(Object x) { System.out.println("In Object"); } //3

    void probe(Long x) { System.out.println("In Long"); } //4

    public static void main(String[] args){
        String a = "hello";
        new TestClass().probe(a);
    }
}
```

What will be printed?

Select 1 option

A. In Integer

B. In Object

C. In Long

D. It will not compile

Check Answer

04. QID - 2.1370

What will the following code print when run?

```
class Baap {
    public int h = 4;
    public int getH() {
        System.out.println("Baap " + h);
        return h;
    }
}

public class Beta extends Baap {
    public int h = 44;
    public int getH() {
        System.out.println("Beta " + h);
        return h;
    }
    public static void main(String[] args) {
        Baap b = new Beta();
        System.out.println(b.h + " " + b.getH());
        Beta bb = (Beta) b;
        System.out.println(bb.h + " " + bb.getH());
    }
}
```

Select 1 option

A.
```
Beta 44
4 44
Baap 44
44 44
```

B.
```
Baap 44
4 44
Beta 44
44 44
```

C.
```
Beta 44
4 44
Beta 44
4 44
```

D.
```
Beta 44
4 44
Beta 44
44 44
```

Check Answer

05. QID - 2.962

Consider the following method...

```
public int setVar(int a, int b, float c) { ...}
```

Which of the following methods correctly overload the above method?

Select 2 options

A.
```
public int setVar(int a, float b, int c){
  return (int)(a + b + c);
}
```

B.
```
public int setVar(int a, float b, int c){
  return this(a, c, b);
}
```

C.
```
public int setVar(int x, int y, float z){
  return x+y;
}
```

D.
```
public float setVar(int a, int b, float c){
  return c*a;
}
```

E.
```
public float setVar(int a){
  return a;
}
```

Check Answer

06. QID - 2.887

Given:

```
class OverloadingTest{

    void m1(int x){
        System.out.println("m1 int");
    }

    void m1(double x){
        System.out.println("m1 double");
    }

    void m1(String x){
        System.out.println("m1 String");
    }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        OverloadingTest ot = new OverloadingTest();
        ot.m1(1.0);
    }
}
```

What will be the output?

Select 1 option

A. It will fail to compile.

B. m1 int

C. ml double

D. ml String

Check Answer

07. QID - 2.1052

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10);        //1
        test1(10, 20); //2
   }

   public static void main(String[] args){
     new Varargs().test();
   }

   //insert method here.
}
```

Which of the following lines can be added independently to the above class so that it will run without any errors or exceptions?

Select 2 options

A. `public void test1(int i, int j){}`

B. `public void test1(int i, int... j){}`

C. `public void test1(int... i){}`

D. `public void test1(int i...){}`

E. `public void test1(int[] i){}`

Check Answer

08. QID - 2.1008

Given the following pairs of method declarations, which of the statements are true?

```
1.
void perform_work(int time){ }
int  perform_work(int time, int speed){ return time*speed ;}

2.
void perform_work(int time){ }
int  perform_work(int speed){return speed ;}

3.
void perform_work(int time){ }
void Perform_work(int time){ }
```

Select 2 options

A. The first pair of methods will compile correctly and overload the method 'perform_work'.

B. The second pair of methods will compile correctly and overload the method 'perform_work'.

C. The third pair of methods will compile correctly and overload the method 'perform_work'.

D. The second pair of methods will not compile correctly.

E. The third pair of methods will not compile correctly.

Check Answer

09. QID - 2.1340

Given the following code, which method declarations can be inserted at line 1 without any problems?

```
public class OverloadTest{
    public int sum(int i1, int i2) { return i1 + i2; }
    // 1
}
```

Select 3 options

A. `public int sum(int a, int b) { return a + b; }`

B. `public int sum(long i1, long i2) { return (int) i1; }`

C. `public int sum(int i1, long i2) { return (int) i2; }`

D. `public long sum(long i1, int i2) { return i1 + i2; }`

E. `public long sum(int i1, int i2) { return i1 + i2; }`

Check Answer

10. QID - 2.1058

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10, 20);  //1
   }

   public void test1(int i, int... j){ System.out.println("1"); }
   public void test1(int... i ){ System.out.println("2"); }
   public void test1(int i, int j){ System.out.println("3"); }

   public static void main(String[] args){
     new Varargs().test();
   }
}
```

What will the program print?

Select 1 option

A. 1

B. 2

C. 3

D. It will not compile.

E. Exception at runtime.

Check Answer

11. QID - 2.990

Consider the following classes...

```
class Teacher{
      void teach(String student){
          /* lots of code */
      }
}
class Prof extends Teacher{
        //1
}
```

Which of the following methods can be inserted at line //1 ?

Select 4 options

A. `public void teach() throws Exception`

B. `private void teach(int i) throws Exception`

C. `protected void teach(String s)`

D. `public final void teach(String s)`

E. `public abstract void teach(String s)`

Check Answer

12. QID - 2.1373

What should be placed in the two blanks so that the following code will compile without errors:

```
class XXX{
    public void m() throws Exception{
        throw new Exception();
    }
}
class YYY extends XXX{
    public void m() {
    }
}
public class TestClass {
    public static void main(String[] args) {
        ______ obj = new ______();
        obj.m();
    }
}
```

Select 1 option

A. XXX and YYY

B. XXX and XXX

C. YYY and YYY

D. YYY and XXX

E. Nothing will make the code compile.

Check Answer

Overloading methods (Answered)

01. QID - 2.1100 : Overloading methods

Consider the following class...

```
class TestClass{
    void probe(int... x) { System.out.println("In ..."); }  //1

    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(long x) { System.out.println("In long"); } //3

    void probe(Long x) { System.out.println("In LONG"); } //4

    public static void main(String[] args){
        Integer a = 4; new TestClass().probe(a); //5
        int b = 4; new TestClass().probe(b); //6
    }
}
```

What will it print when compiled and run?

Correct Options are : A D

A. In Integer and In long

~~**B.**~~ In ... and In LONG, if //2 and //3 are commented out.

~~**C.**~~ In Integer and In ..., if //4 is commented out.

D. It will not compile, if //1, //2, and //3 are commented out.

~~**E.**~~ In LONG and In long, if //1 and //2 are commented out.

Explanation:

To answer this type of questions, you need to know the following rules:

1. The compiler always tries to choose the most specific method available with least number of modifications to the arguments.

2. Java designers have decided that old code should work exactly as it used to work before boxing-unboxing functionality became available.

3. Widening is preferred to boxing/unboxing (because of rule 2), which in turn, is preferred over var-args.

Thus,
1.
`probe(Integer)` is bound to `probe(Integer)` (exact match), then with `probe(long)`, then with `probe(int...)` in that order of preference.
`probe(long)` is preferred over `probe(int...)` because unboxing an Integer gives an int and in pre 1.5 code `probe(long)` is compatible with an int (Rule 2).

It is never bound to `probe(Long )` because `Integer` and `Long` are different object types and there is no IS-A relation between them. (This holds true for any two wrapper classes).
It could, however, be bound to `probe(Object )` (if it existed), because `Integer` IS-A `Object`.

2.
`probe(int)` is bound to `probe(long)` (because of Rule 2) , then to `probe(Integer )` because boxing an `int` qives you an `Integer`, which matches exactly to `probe(Integer)`, and then to `probe(int...)`.

It is never bound to `probe(Long )` because int is not compatible with `Long`.

We advise you to run this program and try out various combinations. The exam has questions on this pattern but they are not this tough. If you have a basic understanding, you should be ok.

Back to Question without Answer

02. QID - 2.939 : Overloading methods

Which of the following are true regarding overloading of a method?

Correct Option is : B

~~**A.**~~ An overloading method must have a different parameter list and same return type as that of the overloaded method.

> There is no restriction on the return type. If the parameters are different then the methods are totally different (other than the name) so their return types can be anything.

B. If there is another method with the same name but with a different number of arguments in a class then that method can be called as overloaded.

~~**C.**~~ If there is another method with the same name and same number and type of arguments but with a different return type in a class then that method can be called as overloaded.

> For overloading a method, the "signature" of the overloaded methods must be different. In simple terms, a method signature includes method name and the number and type of arguments that it takes. So if the parameter list of the two methods with the same name are different either in terms of number or in terms of the types of the parameters, then they are overloaded.
>
> For example:
> Method m1 is overloaded if you have two methods : `void m1(int k); and void m1(double d);`
> `or if you have: void m1(int k); and void m1(int k, double d);`
>
> Note that return type is not considered a part of the method signature.

~~D.~~ An overloaded method means a method with the same name and same number and type of arguments exists in the super class and sub class.

This is called overriding and not overloading.

Back to Question without Answer

03. QID - 2.924 : Overloading methods

Consider the following class...

```
class TestClass{
    void probe(Integer x) { System.out.println("In Integer"); } //2

    void probe(Object x) { System.out.println("In Object"); } //3

    void probe(Long x) { System.out.println("In Long"); } //4

    public static void main(String[] args){
        String a = "hello";
        new TestClass().probe(a);
    }
}
```

What will be printed?

Correct Option is : B

~~**A.**~~ In Integer

B. In Object

~~**C.**~~ In Long

~~**D.**~~ It will not compile

Explanation:

Here, we have three overloaded probe methods but there is no probe method that takes

a String parameter. The only one that is able to accept a String is the one that takes Object as a parameter. So that method will be called.

A String cannot be assigned to a variable of class Integer or Long variable, but it can be assigned to a variable of class Object.

Back to Question without Answer

04. QID - 2.1370 : Overloading methods

What will the following code print when run?

```
class Baap {
    public int h = 4;
    public int getH() {
        System.out.println("Baap " + h);
        return h;
    }
}

public class Beta extends Baap {
    public int h = 44;
    public int getH() {
        System.out.println("Beta " + h);
        return h;
    }
    public static void main(String[] args) {
        Baap b = new Beta();
        System.out.println(b.h + " " + b.getH());
        Beta bb = (Beta) b;
        System.out.println(bb.h + " " + bb.getH());
    }
}
```

Correct Option is : D

~~**A.**~~
```
Beta 44
4 44
Baap 44
44 44
```

~~**B.**~~
```
Baap 44
4 44
Beta 44
44 44
```

~~**C.**~~
```
Beta 44
4 44
Beta 44
4 44
```

D.
```
Beta 44
4 44
Beta 44
44 44
```

Explanation:

Always remember: Methods are overridden and variables are shadowed.
Here, b refers to an object of class Beta so `b.getH()` will always call the overridden (subclass's method). However, the type of reference of b is Baap. so b.h will always refer to Baap's h.

Further, inside Beta's `getH()`, Beta's h will be accessed instead of Baap's h because you are accessing this.h ('this' is implicit) and the type of this is Beta.

Back to Question without Answer

05. QID - 2.962 : Overloading methods

Consider the following method...

```
public int setVar(int a, int b, float c) { ...}
```

Which of the following methods correctly overload the above method?

Correct Options are : A E

A.
```
public int setVar(int a, float b, int c){
  return (int)(a + b + c);
}
```

~~**B.**~~
```
public int setVar(int a, float b, int c){
  return this(a, c, b);
}
```

this(...) can only be called in a constructor and that too as a first statement.

~~**C.**~~
```
public int setVar(int x, int y, float z){
  return x+y;
}
```

It will not compile because it is same as the original method. The name of parameters do not matter.

~~**D.**~~
```
public float setVar(int a, int b, float c){
  return c*a;
}
```

It will not compile as it is same as the original method. The return type does not matter.

E.
```
public float setVar(int a){
  return a;
}
```

Explanation:

A method is said to be overloaded when the other method's name is same and parameters (either the number or their order) are different.
Option 2 is not valid Because of the line: `return this(a, c, b);` This is the syntax of calling a constructor and not a method. It should have been: `return this.setVar(a, c, b);`

Back to Question without Answer

06. QID - 2.887 : Overloading methods

Given:

```
class OverloadingTest{

    void m1(int x){
        System.out.println("m1 int");
    }

    void m1(double x){
        System.out.println("m1 double");
    }

    void m1(String x){
        System.out.println("m1 String");
    }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        OverloadingTest ot = new OverloadingTest();
        ot.m1(1.0);
    }
}
```

What will be the output?

Correct Option is : C

~~**A.**~~ It will fail to compile.

~~**B.**~~ m1 int

C. m1 double

~~**D.**~~ m1 String

Explanation:

Here, `m1()` is overloading for three different argument types. So when you call `ot.m1(1.0)`, the one with argument of type double will be invoked.

Back to Question without Answer

07. QID - 2.1052 : Overloading methods

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10);        //1
        test1(10, 20); //2
   }

   public static void main(String[] args){
     new Varargs().test();
   }

   //insert method here.
}
```

Which of the following lines can be added independently to the above class so that it will run without any errors or exceptions?

Correct Options are : B C

A. ~~`public void test1(int i, int j){}`~~

This will work only for //2.

B. `public void test1(int i, int... j){}`

The last parameter is a varargs of type int, which means, it can take any number of integers. Thus, it satisfies both //1 and //2.

C. `public void test1(int... i){}`

Since the only parameter is a varargs of type int, it can take any number of

integers. Thus, it satisfies both //1 and //2.

D. `public void test1(int i...){}`

This is not a correct syntax for a var-arg parameter.

E. `public void test1(int[] i){}`

Even though a var-arg parameter of type int is very similar to int[], they are not interchangeable. Therefore, int[] cannot be substituted for int... and it will not satisfy either of //1 or //2.

Explanation:

An interesting observation is that if you do javap on the following class, you will see the same signature for method m1 and m2. This shows that a var-arg parameter of type T is same as an array of type T.

```
class TestClass {
    String m1(int[] i) { return ""+i.length; }
    String m2(int... i) { return ""+i.length; }
}


javap TestClass produces this:

    java.lang.String m1(int[]);
    java.lang.String m2(int[]);


Conversely, the following code will not compile:

class TestClass {
    String m1(int[] i) { return ""+i.length; }
    String m1(int... i) { return ""+i.length; }  // Compiler determi
}
```

Back to Question without Answer

08. QID - 2.1008 : Overloading methods

Given the following pairs of method declarations, which of the statements are true?

```
1.
void perform_work(int time){ }
int  perform_work(int time, int speed){ return time*speed ;}

2.
void perform_work(int time){ }
int  perform_work(int speed){return speed ;}

3.
void perform_work(int time){ }
void Perform_work(int time){ }
```

Correct Options are : A D

A. The first pair of methods will compile correctly and overload the method 'perform_work'.

~~B.~~ The second pair of methods will compile correctly and overload the method 'perform_work'.

> You cannot have two methods with the same signature (i.e. same name and same parameter list) in the same class.
> Note that return type and names of the parameters don't matter while determining the signature.

~~C.~~ The third pair of methods will compile correctly and overload the method 'perform_work'.

D. The second pair of methods will not compile correctly.

~~**E.**~~ The third pair of methods will not compile correctly.

Both have different names (note the capital 'P') and so are different methods.

Explanation:

Overloading of a method occurs when the name of more than one methods is exactly same but the parameter lists are different.

The first and the third pairs of methods will compile correctly as they follow the above stated rule.

The second pair of methods will not compile correctly, since their method signatures are same and the compiler cannot differentiate between the two methods as it does not look for return type. Also, only name and input parameters are the part of method declaration . Names of the parameters don't matter.

Both methods in the first pair are named perform_work but have different parameter list so they overload this method name i.e. perform_work.

The method named 'perform_work' is distinct from the method named 'Perform_work', as identifiers in Java are case-sensitive.

Back to Question without Answer

09. QID - 2.1340 : Overloading methods

Given the following code, which method declarations can be inserted at line 1 without any problems?

```
public class OverloadTest{
    public int sum(int i1, int i2) { return i1 + i2; }
    // 1
}
```

Correct Options are : B C D

~~**A.**~~ `public int sum(int a, int b) { return a + b; }`

> Will cause duplicate method. Variable names don't matter. Only their types.

B. `public int sum(long i1, long i2) { return (int) i1; }`

C. `public int sum(int i1, long i2) { return (int) i2; }`

D. `public long sum(long i1, int i2) { return i1 + i2; }`

~~**E.**~~ `public long sum(int i1, int i2) { return i1 + i2; }`

> Only the return type is different so the compiler will complain about having duplicate method sum.

Explanation:

The rule is that you cannot have methods that create ambiguity for the compiler in a

class. It is illegal for a class to have two methods having same name and having same type of input parameters in the same order.
Name of the input variables and return type of the method are not looked into.
1. Option 1 is wrong because, then both the methods will be same (as their method name and the class/type and order of the input parameters will be same). So this amounts to duplicate method which is not allowed.
As mentioned, name of the input parameters does not matter. Only the type of parameters and their order matters.
2. 2 is valid because the type of input parameters are different. So this is a different method and does not amount to duplication.
3 and 4 are valid for the same reason
5 is not valid because it leads to duplicate method(as their method name and the class/type and order of the input parameters will be same). Note that as mentioned in the comments, return type does not matter.

Back to Question without Answer

10. QID - 2.1058 : Overloading methods

Consider the following code:

```
public class Varargs{
   public void test(){
        test1(10, 20);  //1
   }

   public void test1(int i, int... j){ System.out.println("1"); }
   public void test1(int... i ){ System.out.println("2"); }
   public void test1(int i, int j){ System.out.println("3"); }

   public static void main(String[] args){
     new Varargs().test();
   }
}
```

What will the program print?

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 2

C. 3

~~**D.**~~ It will not compile.

~~**E.**~~ Exception at runtime.

Explanation:

In cases where multiple methods are applicable, the compiler always calls the most specific one. In this case, the third one is the most specific one.

If no method is more specific than the other, then the compilation fails. For example, if the class did not have the third method test1(int i, int j), then the remaining two methods would have been equally applicable and equally specific. In that case, it would not compile.

Back to Question without Answer

11. QID - 2.990 : Overloading methods

Consider the following classes...

```
class Teacher{
      void teach(String student){
          /* lots of code */
      }
}
class Prof extends Teacher{
        //1
}
```

Which of the following methods can be inserted at line //1 ?

Correct Options are : A B C D

A. `public void teach() throws Exception`

> It overloads the `teach()` method instead of overriding it.

B. `private void teach(int i) throws Exception`

> It overloads the `teach()` method instead of overriding it.

C. `protected void teach(String s)`

> This overrides Teacher's teach method. The overriding method can have more visibility. (It cannot have less. Here, it cannot be private.)

D. `public final void teach(String s)`

> Overriding method may be made final.

E. ~~`public abstract void teach(String s)`~~

This is wrong because class Prof has not been declared as abstract. Note that otherwise it is OK to override a method by an abstract method.

Explanation:

Note that 'protected' is less restrictive than default 'no modifier'. So choice 3 is valid. "public abstract void teach(String s)" would have been valid if class Prof had been declared abstract.

Back to Question without Answer

12. QID - 2.1373 : Overloading methods

What should be placed in the two blanks so that the following code will compile without errors:

```
class XXX{
    public void m() throws Exception{
        throw new Exception();
    }
}
class YYY extends XXX{
    public void m() {
    }
}
public class TestClass {
    public static void main(String[] args) {
        ______ obj = new ______();
        obj.m();
    }
}
```

Correct Option is : C

~~**A.**~~ XXX and YYY

~~**B.**~~ XXX and XXX

C. YYY and YYY

~~**D.**~~ YYY and XXX

~~**E.**~~ Nothing will make the code compile.

Explanation:

1. The overriding method may choose to have no `throws` clause even if the overridden method has a `throws` clause.
2. Whether a call needs to be wrapped in a try/catch or whether the enclosing method requires a throws clause depends on the class of the reference and not the class of the actual object.

Here, if you define s of type XXX, the call `s.m()` will have to be wrapped into a try/catch because `main()` doesn't have a throws clause. But if you define s of class YYY, there is no need of try catch because YYY's `m()` does not throw an exception. Now, if the class of s is YYY, you cannot assign it an object of class XXX because XXX is a superclass of YYY. So the only option is to do:

```
YYY s = new YYY();
```

Back to Question without Answer

Using Loop Constructs

Exam Objectives -

Create and use while loops Create and use for loops including the enhanced for loop Create and use do/while loops Compare loop constructs Use break and continue

01. QID - 2.849

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {
        String[] sa = {"a", "b", "c"};
        for(String s :  sa){
            if("b".equals(s)) continue;
            System.out.println(s);
            if("b".equals(s)) break;
            System.out.println(s+" again");
        }
    }
}
```

Select 1 option

A.
```
a
a again
c
c again
```

B.
```
a
a again
b
```

C.
```
a
a again
b
b again
```

D.
```
c
c again
```

Check Answer

02. QID - 2.1280

In the following code what will be the output if 0 (integer value zero) is passed to loopTest()?

```
public class TestClass{
   public void loopTest(int x){
      loop: for (int i = 1; i < 5; i++){
         for (int j = 1; j < 5; j++){
            System.out.println(i);
            if (x == 0) {  continue loop;  }
            System.out.println(j);
         }
      }
   }
}
```

Select 1 option

A. The program will not compile.

B. It will print 1 2 3 4

C. It will print 1 1 2 3 4

D. It will print 1 1 2 2 3 3 4 4

E. Produces no output

Check Answer

03. QID - 2.1356

How many times will the line marked //1 be called in the following code?

```
int x = 10;
do{
 x--;
 System.out.println(x);  // 1
}while(x<10);
```

Select 1 option

A. 0

B. 1

C. 9

D. 10

E. None of these.

Check Answer

04. QID - 2.1245

Consider the following class :

```
class Test{
   public static void main(String[] args){
      for (int i = 0; i < 10; i++) System.out.print(i + " ");  //1
      for (int i = 10; i > 0; i--) System.out.print(i + " ");  //2
      int i = 20;                  //3
      System.out.print(i + " ");   //4
   }
}
```

Which of the following statements are true?

Select 4 options

A. As such the class will compile and print 20 at the end of its output.

B. It will not compile if line 3 is removed.

C. It will not compile if line 3 is removed and placed before line 1.

D. It will not compile if line 4 is removed and placed before line 3.

E. Only Option 2, 3, and 4 are correct.

Check Answer

05. QID - 2.916

Consider the following code written by a new developer:

```
while(true){
        //additional valid code
        if(isDone()) break;
}
```

What can be done to make this code more readable?

Select 1 option

A. Use a for loop

B. Use the enhanced for loop

C. use do-while instead of while.

D. Use continue instead of break.

Check Answer

06. QID - 2.1067

Which of these group of statements are valid?

Select 2 options

A. `{ { } }`

B. `{ continue ; }`

C. `block : { break block ; }`

D. `block : { continue block ; }`

E. The break keyword can only be used if there exists an enclosing loop construct (i.e. while, do-while or for).

Check Answer

07. QID - 2.1060

Identify the valid for loop constructs assuming the following declarations:

```
Object o = null;
Collection c = //valid collection object.
int[][] ia = //valid array
```

Select 2 options

A. `for(o : c){ }`

B. `for(final Object o2 :c){ }`

C. `for(int i : ia) { }`

D. `for(Iterator it : c.iterator()){ }`

E. `for(int i : ia[0]){ }`

Check Answer

08. QID - 2.1121

What will the following code print?

```
void crazyLoop(){
  int c = 0;
  JACK: while (c < 8){
    JILL: System.out.println(c);
    if (c > 3) break JACK; else c++;
  }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. It will print numbers from 0 to 8

D. It will print numbers from 0 to 3

E. It will print numbers from 0 to 4

Check Answer

09. QID - 2.1279

Which of the following code snippets will compile without any errors?
(Assume that the statement `int x = 0;` exists prior to the statements below.)

Select 3 options

A. `while (false) { x=3; }`

B. `if (false) { x=3; }`

C. `do{ x = 3; } while(false);`

D. `for( int i = 0; i< 0; i++) x = 3;`

Check Answer

10. QID - 2.1114

What will the following code print?

```
public class TestClass{
        int x = 5;
        int getX(){ return x; }

        public static void main(String args[]) throws Exception{
            TestClass tc = new TestClass();
            tc.looper();
            System.out.println(tc.x);
        }

        public void looper(){
            int x = 0;
            while( (x = getX()) != 0 ){
                for(int m = 10; m>=0; m--){
                    x = m;
                }
            }

       }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. It will print 0.

D. It will print 5.

E. None of these.

Check Answer

11. QID - 2.1013

Using a break in a while loop causes the loop to break the current iteration and start the next iteration of the loop.

Select 1 option

A. True

B. False

Check Answer

12. QID - 2.1227

Given the following code, which of these statements are true?

```
class TestClass{
   public static void main(String args[]){
      int k = 0;
      int m = 0;
      for ( int i = 0; i <= 3; i++){
         k++;
         if ( i == 2){
            // line 1
         }
         m++;
      }
      System.out.println( k + ", " + m );
   }
}
```

Select 3 options

A. It will print 3, 2 when line 1 is replaced by break;

B. It will print 3, 2 when line 1 is replaced by continue.

C. It will print 4, 3 when line 1 is replaced by continue.

D. It will print 4, 4 when line 1 is replaced by i = m++;

E. It will print 3, 3 when line 1 is replaced by i = 4;

Check Answer

13. QID - 2.1265

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int i=0, j=0;
        X1: for(i = 0; i < 3; i++){
            X2: for(j = 3; j > 0; j--){
                if(i < j) continue X1;
                else break X2;
            }
        }
        System.out.println(i+" "+j);
    }
}
```

Select 1 option

A. 0 3

B. 0 2

C. 3 0

D. 3 3

E. 2 2

Check Answer

14. QID - 2.891

What can you do to make the following code compile?

```
public class TestClass {
    public static void main(String[] args) {
        int[] values = { 10, 20, 30 };
        for( /* put code here */ ){
        }
    }
}
```

Select 2 options

A. int k : values

B. int k in values

C. int k; k<0; k++

D. ;;

E. ; k<values.length;k++

Check Answer

15. QID - 2.1010

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     for : for(int i = 0; i< 10; i++){
        for (int j = 0; j< 10; j++){
             if ( i+ j > 10 )  break for;
        }
        System.out.println( "hello");
     }
  }
}
```

Select 1 option

A. It will print `hello` 6 times.

B. It will not compile.

C. It will print `hello` 2 times.

D. It will print `hello` 5 times.

E. It will print `hello` 4 times.

Check Answer

16. QID - 2.1083

What will the following program print?

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

Select 1 option

A. 2

B. 3

C. 6

D. 7

E. 9

Check Answer

17. QID - 2.914

Consider the following code:

```
public static void main(String[] args) {
   int[] values = { 10, 30, 50 };
        for( int val : values ){
            int x = 0;
            while(x<values.length){
             System.out.println(x+" "+val);
              x++;
            }
        }
    }
```

How many times is 2 printed out?

Select 1 option

A. 0

B. 1

C. 2

D. 3

Check Answer

18. QID - 2.960

What will the following code print when compiled and run?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        A: for(int i = 0; i < 2; i++){
            B: for(int j = 0; j < 2; j++){
                C: for(int k = 0; k < 3; k++){
                    c++;
                    if(k>j) break;
                }
            }
        }
        System.out.println(c);
    }
}
```

Select 1 option

A. 7

B. 8

C. 9

D. 10

E. 11

Check Answer

19. QID - 2.1042

Consider the following method which is called with an argument of 7:

```
public void method1(int i){
   int j = (i*30 - 2)/100;

   POINT1 : for(;j<10; j++){
       boolean flag  = false;
       while(!flag){
 if(Math.random()>0.5) break POINT1;
       }
   }
  while(j>0){
     System.out.println(j--);
     if(j == 4) break POINT1;
   }
}
```

What will it print?
(Assume that Math.random() return a double between 0.0 and 1.0, not including 1.0)

Select 1 option

A. It will print `1` and `2`

B. It will print `1` to N where N is a random number.

C. It will not compile.

D. It will throw an exception at runtime.

Check Answer

20. QID - 2.1016

What will the following code snippet print?

```
int count = 0, sum = 0;
do{
        if(count % 3 == 0) continue;
        sum+=count;
}
while(count++ < 11);
System.out.println(sum);
```

Select 1 option

A. 49

B. 48

C. 37

D. 36

E. 38

Check Answer

21. QID - 2.1099

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int x  = 0;
      labelA:   for (int i=10; i<0; i--){
         int j = 0;
         labelB:
         while (j < 10){
            if (j > i) break labelB;
            if (i == j){
               x++;
               continue labelA;
            }
            j++;
         }
         x--;
      }
      System.out.println(x);
   }
}
```

Select 1 option

A. It will not compile.

B. It will go in infinite loop when run.

C. The program will write 10 to the standard output.

D. The program will write 0 to the standard output.

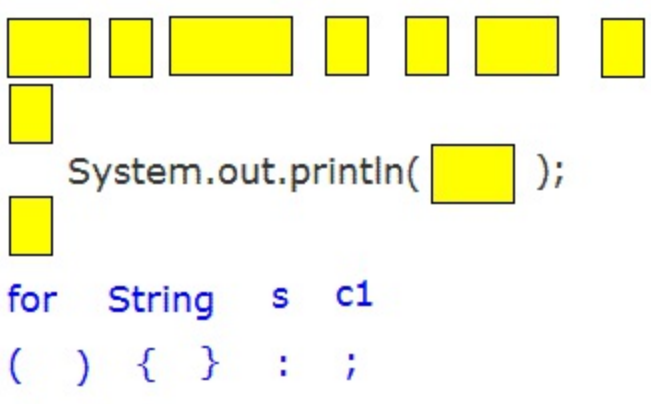
E. None of the above.

Check Answer

22. QID - 2.1043

Assuming the following declarations, write a for loop that prints each string in the collection using the given elements.

```
Collection<String> c1 = new HashSet<String>();
```



Check Answer

23. QID - 2.1044

What will be the output if you run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0 ; j < 1 ; ++j , i++){
         System.out.println( i + " " + j );
      }
      System.out.println( i + " " + j );
   }
}
```

Select 1 option

A. `0 0` will be printed twice.

B. `1 1` will be printed once.

C. `0 1` will be printed followed by `1 2`.

D. `0 0` will be printed followed by `1 1`.

E. It will print `0 0` and then `0 1`.

Check Answer

24. QID - 2.1233

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        boolean flag = true;
        for(int i = 0; i < 3; i++){
            while(flag){
                c++;
                if(i>c || c>5) flag = false;
            }
        }
        System.out.println(c);
    }
}
```

Select 1 option

A. 3

B. 4

C. 5

D. 6

E. 7

Check Answer

25. QID - 2.1299

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int var = 20, i=0;
      do{
         while(true){
         if( i++ > var) break;
         }
      }while(i<var--);
      System.out.println(var);
   }
}
```

Select 1 option

A. 19

B. 20

C. 21

D. 22

E. It will enter an infinite loop.

Check Answer

26. QID - 2.1050

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      for (i=1 ;  i<5  ; i++) continue;  //(1)
      for (i=0 ;       ; i++) break;       //(2)
      for (    ; i<5?false:true ;    );     //(3)
   }
}
```

Select 1 option

A. The code will compile without error and will terminate without problems when run.

B. The code will fail to compile, since the `continue` can't be used this way.

C. The code will fail to compile, since the `break` can't be used this way

D. The code will fail to compile, since the `for` statement in the line 2 is invalid.

E. The code will compile without error but will never terminate.

Check Answer

27. QID - 2.1193

Which of these statements are valid when occurring by themselves?

Select 3 options

A. `while ( ) break ;`

B. `do { break ; } while (true) ;`

C. `if (true) { break ; }` (When not inside a switch block or a loop)

D. `switch (1) { default : break; }`

E. `for ( ; true ; ) break ;`

Check Answer

28. QID - 2.1032

What will the following code print?

```
public class BreakTest{
  public static void main(String[] args){
    int i = 0, j = 5;
    lab1 : for( ; ; i++){
      for( ; ; --j)  if( i >j ) break lab1;
    }
    System.out.println(" i = "+i+", j = "+j);
  }
}
```

Select 1 option

A. `i = 1, j = -1`

B. `i = 1, j = 4`

C. `i = 0, j = 4`

D. `i = 0, j = -1`

E. It will not compile.

Check Answer

29. QID - 2.848

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        do{
            System.out.println(k);
        }while(--k>0);
    }
}
```

Select 1 option

A.
```
1
```

B.
```
1
0
```

C.
```
2
1
```

D.
```
2
1
0
```

E. It will keeping printing numbers in an infinite loop.

F. It will not compile.

Check Answer

30. QID - 2.963

Consider the following code snippet:

```
for(int i=INT1; i<INT2; i++){
    System.out.println(i);
}
```

INT1 and INT2 can be any two integers.

Which of the following will produce the same result?

Select 1 option

A. `for(int i=INT1; i<INT2; System.out.println(++i));`

B. `for(int i=INT1; i++<INT2; System.out.println(i));`

C. `int i=INT1; while(i++<INT2) { System.out.println(i); }`

D. `int i=INT1; do { System.out.println(i); }while(i++<INT2);`

E. None of these.

Check Answer

31. QID - 2.1281

Which of the following statements regarding 'break' and 'continue' are true?

Select 1 option

A. break without a label, can occur only in a switch, while, do, or for statement.

B. continue without a label, can occur only in a switch, while, do, or for statement.

C. break can never occur without a label.

D. continue can never occur WITH a label.

E. None of the above.

Check Answer

32. QID - 2.1120

What will the following program snippet print?

```
int i=0, j=11;
do{
   if(i > j) { break; }
   j--;
}while( ++i < 5);
System.out.println(i+"  "+j);
```

Select 1 option

A. 5 5

B. 5 6

C. 6 6

D. 6 5

E. 4 5

Check Answer

33. QID - 2.847

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        while(--k){
            System.out.println(k);
        }
    }
}
```

Select 1 option

A. 1

B. 1
0

C. 2
1

D. 2
1
0

E. It will keeping printing numbers in an infinite loop.

F. It will not compile.

Check Answer

34. QID - 2.1270

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      boolean b = false;
      int i = 1;
      do{
         i++ ;
      } while (b = !b);
      System.out.println( i );
   }
}
```

Select 1 option

A. The code will fail to compile, 'while' has an invalid condition expression.

B. It will compile but will throw an exception at runtime.

C. It will print 3.

D. It will go in an infinite loop.

E. It will print 1.

Check Answer

35. QID - 2.1327

What will be the output when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0; j < i; ++j, i++){
         System.out.println(i + " " + j);
      }
      System.out.println(i + " " + j);
   }
}
```

Select 1 option

A. 0 0 will be printed twice.

B. 0 0 will be printed once.

C. It will keep on printing 0 0

D. It will not compile.

E. It will print 0 0 and then 0 1.

Check Answer

36. QID - 2.1059

Identify valid for constructs...
Assume that Math.random() returns a double between 0.0 and 1.0 (not including 1.0).

Select 3 options

A.
```
for(;Math.random()<0.5;){
    System.out.println("true");
}
```

B.
```
for(;;Math.random()<0.5){
    System.out.println("true");
}
```

C.
```
for(;;Math.random()){
    System.out.println("true");
}
```

D.
```
for(;;){
    Math.random()<.05? break : continue;
}
```

E.
```
for(;;){
    if(Math.random()<.05) break;
}
```

Check Answer

37. QID - 2.1057

What will the following code print?

```
void crazyLoop(){
   int c = 0;
   JACK: while (c < 8){
       JILL: System.out.println(c);
       if (c > 3) break JILL; else c++;
   }
}
```

Select 1 option

A. It will not compile.

B. It will throw an exception at runtime.

C. It will print numbers from 0 to 8

D. It will print numbers from 0 to 3

E. It will print numbers from 0 to 4

Check Answer

Using Loop Constructs (Answered)

01. QID - 2.849 : Using Loop Constructs

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {
        String[] sa = {"a", "b", "c"};
        for(String s :  sa){
            if("b".equals(s)) continue;
            System.out.println(s);
            if("b".equals(s)) break;
            System.out.println(s+" again");
        }
    }
}
```

Correct Option is : A

A.
```
a
a again
c
c again
```

~~**B.**~~
```
a
a again
b
```

~~**C.**~~
```
a
a again
b
b again
```

~~**D.**~~
```
c
c again
```

Explanation:

To determine the output you have to run through the loop one iteration at a time in your mind:

Iteration 1: s is "a". It is not equal to "b" so, it will print "a", and then "a again".
Iteration 2: s is "b". It is equal to "b", so the first if will execute "continue", which mean the rest of the code in the loop will not be executed (thus b and b again will not be printed), and the next iteration will start. Note that the second if is not executed at all because of the continue in the first if.
Iteration 3: s is "c", both the if conditions are not satisfied. So "c" and "c again" will be printed.

Back to Question without Answer

02. QID - 2.1280 : Using Loop Constructs

In the following code what will be the output if 0 (integer value zero) is passed to loopTest()?

```
public class TestClass{
   public void loopTest(int x){
      loop: for (int i = 1; i < 5; i++){
         for (int j = 1; j < 5; j++){
            System.out.println(i);
            if (x == 0) {  continue loop;  }
            System.out.println(j);
         }
      }
   }
}
```

Correct Option is : B

~~**A.**~~ The program will not compile.

B. It will print 1 2 3 4

~~**C.**~~ It will print 1 1 2 3 4

~~**D.**~~ It will print 1 1 2 2 3 3 4 4

~~**E.**~~ Produces no output

Explanation:

When x is 0, the statement continue loop; is executed. Note that loop: is for the outer loop. So, only one iteration (that too not full) is performed for the inner loop.
So, the inner loop prints the value of i only once and then next iteration of outer loop starts. 'j' is never printed. So, it prints 1 2 3 4.

Back to Question without Answer

03. QID - 2.1356 : Using Loop Constructs

How many times will the line marked //1 be called in the following code?

```
int x = 10;
do{
 x--;
 System.out.println(x);  // 1
}while(x<10);
```

Correct Option is : E

~~**A.**~~ 0

~~**B.**~~ 1

~~**C.**~~ 9

~~**D.**~~ 10

E. None of these.

Explanation:

A do-while loop is always executed at least once. So in the first iteration, x is decremented and becomes 9. Now the while condition is tested, which returns true because 9 is less than 10. So the loop is executed again with x = 9. In the loop, x is decremented to 8 and the condition is tested again, which again returns true because 8 is less than 10.

As you can see, x keeps on decreasing by one in each iteration and every time the condition x<10 returns true. However, after x reaches -2147483648, which is its MIN_VALUE, it cannot decrease any further and at this time when x-- is executed, the value rolls over to 2147483647, which is Integer.MAX_VALUE. At this time, the condition x<10 fails and the loop terminates.

Back to Question without Answer

04. QID - 2.1245 : Using Loop Constructs

Consider the following class :

```
class Test{
   public static void main(String[] args){
      for (int i = 0; i < 10; i++) System.out.print(i + " ");  //1
      for (int i = 10; i > 0; i--) System.out.print(i + " ");  //2
      int i = 20;                  //3
      System.out.print(i + " ");   //4
   }
}
```

Which of the following statements are true?

Correct Options are : A B C D

A. As such the class will compile and print 20 at the end of its output.

B. It will not compile if line 3 is removed.

> If //3 is removed, 'i' will be undefined for //4

C. It will not compile if line 3 is removed and placed before line 1.

D. It will not compile if line 4 is removed and placed before line 3.

~~**E.**~~ Only Option 2, 3, and 4 are correct.

Explanation:

The scope of a local variable declared in 'for' statement is the rest of the 'for' statement, including its own initializer. So, when line 3 is placed before line 1, there is a redeclaration of i in the first for() which is not legal.
As such, the scope of i's declared in for() is just within the 'for' blocks. So placing line 4 before line 3 will not work since 'i' is not in scope there.

Back to Question without Answer

05. QID - 2.916 : Using Loop Constructs

Consider the following code written by a new developer:

```
while(true){
        //additional valid code
        if(isDone()) break;
}
```

What can be done to make this code more readable?

Correct Option is : C

~~**A.**~~ Use a for loop

> The following is how it can be done using a for loop:
> ```
> for(;!isDone();) {
> //additional valid code
> }
> ```

~~**B.**~~ Use the enhanced for loop

C. use do-while instead of while.

> In the current state, the actual loop breaker condition is coded far away from the while condition. This reduces readability because it isn't immediately known when the loop breaks. Therefore, it should be changed to:
> ```
> do{
> } while( !isDone() );
> ```

~~**D.**~~ Use continue instead of break.

Back to Question without Answer

06. QID - 2.1067 : Using Loop Constructs

Which of these group of statements are valid?

Correct Options are : A C

A. `{ { } }`

> See explanation.

~~B.~~ `{ continue ; }`

> continue can be used only inside a 'for', 'while' or 'do while' loop.

C. `block : { break block ; }`

> This is a valid example of breaking out of a labelled block.

~~D.~~ `block : { continue block ; }`

> continue can be used only inside a 'for', 'while' or 'do while' loop.

~~E.~~ The break keyword can only be used if there exists an enclosing loop construct (i.e. while, do-while or for).

> It can also be used to break out of a labeled block and in switch construct. For example, option 3.

Explanation:

The construct '{ }' is a compound statement. The compound statement can contain zero or more arbitrary statements.

Thus, { { } }, which is a compound statement containing one statement which is a compound statement containing no statement, is legal.

Back to Question without Answer

07. QID - 2.1060 : Using Loop Constructs

Identify the valid for loop constructs assuming the following declarations:

```
Object o = null;
Collection c = //valid collection object.
int[][] ia = //valid array
```

Correct Options are : B E

~~**A.**~~ `for(o : c){ }`

> Cannot use an existing/predefined variable in the variable declaration part.

B. `for(final Object o2 :c){ }`

> final is the only modifier (excluding annotations) that is allowed here.

~~**C.**~~ `for(int i : ia) { }`

> Each element of ia is itself an array. Thus, they cannot be assigned to an int.

~~**D.**~~ `for(Iterator it : c.iterator()){ }`

> `c.iterator()` does not return any Collection. Note that the following would have been valid:
>
> ```
> Collection<Iterator> c = //some collection that contains
> Iterator objects
> for(Iterator it : c){ }
> ```

E. `for(int i : ia[0]){ }`

Since `ia[0]` is an array of ints, this is valid. (It may throw a `NullPointerException` or `ArrayIndexOutOfBoundsException` at runtime if `ia` is not appropriately initialized.)

Back to Question without Answer

08. QID - 2.1121 : Using Loop Constructs

What will the following code print?

```
void crazyLoop(){
  int c = 0;
  JACK: while (c < 8){
    JILL: System.out.println(c);
    if (c > 3) break JACK; else c++;
  }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ It will print numbers from 0 to 8

~~**D.**~~ It will print numbers from 0 to 3

E. It will print numbers from 0 to 4

Explanation:

This is a straight forward loop that contains a labelled break statement. A labelled break breaks out of the loop that is marked with the given label. Therefore, a labelled break is used to break out from deeply nested loops to the outer loops. Here, there is only one nested loop so the break; and break JACK; are same, but consider the

following code:

```
public static void crazyLoop(){
  int c = 0;
  JACK: while (c < 8){
    JILL: System.out.println(c);
    for(int k = 0; k<c; k++){
        System.out.println(" k = "+k+" c = "+c);
        if (c > 3) break JACK;
    }
    c++;
  }
}
```

This code prints:

```
c = 0
c = 1
  k = 0 c = 1
c = 2
  k = 0 c = 2
  k = 1 c = 2
c = 3
  k = 0 c = 3
  k = 1 c = 3
  k = 2 c = 3
c = 4
  k = 0 c = 4
```

As you can see, in this case, break JACK; will break out from the outer most loop (the while loop). If break JACK; is replaced by break; it will print:

```
c = 0
c = 1
  k = 0 c = 1
c = 2
  k = 0 c = 2
  k = 1 c = 2
c = 3
  k = 0 c = 3
  k = 1 c = 3
```

```
   k = 2 c = 3
c = 4
   k = 0 c = 4
c = 5
   k = 0 c = 5
c = 6
   k = 0 c = 6
c = 7
   k = 0 c = 7
```

This shows that a break without a label only breaks out of the current loop.

Back to Question without Answer

09. QID - 2.1279 : Using Loop Constructs

Which of the following code snippets will compile without any errors?
(Assume that the statement `int x = 0;` exists prior to the statements below.)

Correct Options are : B C D

~~**A.**~~ `while (false) { x=3; }`

B. `if (false) { x=3; }`

C. `do{ x = 3; } while(false);`

> In a do- while, the block is ALWAYS executed at least once because the condition check is done after the block is executed. Unlike a while loop, where the condition is checked before the execution of the block.

D. `for( int i = 0; i< 0; i++) x = 3;`

Explanation:

while `(false) { x=3; }` is a compile-time error because the statement `x=3;` is not reachable;
Similarly, `for( int i = 0; false; i++) x = 3;` is also a compile time error because `x= 3` is unreachable.

In `if(false){ x=3; },` although the body of the condition is unreachable, this is not an error because the JLS explicitly defines this as an exception to the rule. It allows this construct to support optimizations through the conditional compilation. For example,

```
if(DEBUG){ System.out.println("beginning task 1"); }
```

Here, the DEBUG variable can be set to false in the code while generating the production version of the class file, which will allow the compiler to optimize the code by removing the whole if statement entirely from the class file.

Back to Question without Answer

10. QID - 2.1114 : Using Loop Constructs

What will the following code print?

```
public class TestClass{
        int x = 5;
        int getX(){ return x; }

        public static void main(String args[]) throws Exception{
            TestClass tc = new TestClass();
            tc.looper();
            System.out.println(tc.x);
        }

        public void looper(){
            int x = 0;
            while( (x = getX()) != 0 ){
                for(int m = 10; m>=0; m--){
                    x = m;
                }
            }

       }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ It will print 0.

~~**D.** It will print 5.~~

E. None of these.

This program will compile and run but will never terminate.

Explanation:

Note that looper() declares an automatic variable x, which shadows the instance variable x. So when x = m; is executed, it is the local variable x that is changed not the instance field x. So getX() never returns 0. If you remove int x = 0; from looper(), it will print 0 and end.

Back to Question without Answer

11. QID - 2.1013 : Using Loop Constructs

Using a break in a while loop causes the loop to break the current iteration and start the next iteration of the loop.

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

The break statement is to break out of any loop completely. So the current iteration and any other remaining iterations of the loop will not execute.
Control is transferred to the first statement after the loop.

Back to Question without Answer

12. QID - 2.1227 : Using Loop Constructs

Given the following code, which of these statements are true?

```
class TestClass{
   public static void main(String args[]){
      int k = 0;
      int m = 0;
      for ( int i = 0; i <= 3; i++){
         k++;
         if ( i == 2){
            // line 1
         }
         m++;
      }
      System.out.println( k + ", " + m );
   }
}
```

Correct Options are : A C E

A. It will print 3, 2 when line 1 is replaced by break;

B. It will print 3, 2 when line 1 is replaced by continue.

C. It will print 4, 3 when line 1 is replaced by continue.

D. It will print 4, 4 when line 1 is replaced by i = m++;

It will print 4, 5

E. It will print 3, 3 when line 1 is replaced by i = 4;

Explanation:

This is a simple loop. All you need to do is execute each statement in your head. For example, if line 1 is replaced by break:

```
1. k=0, m=0
2. iteration 1: i=0
    2.1 k = 1
    2.2 i == 2 is false
    2.3 m = 1
3. iteration 2: i = 1
    3.1 k=2
    3.2 i==2 is false
    3.3 m = 2
4. iteration 3: i = 2
    4.1 k=3
    4.2 i==2 is true
    4.3 break
5. print 3, 2
```

Back to Question without Answer

13. QID - 2.1265 : Using Loop Constructs

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int i=0, j=0;
        X1: for(i = 0; i < 3; i++){
            X2: for(j = 3; j > 0; j--){
                if(i < j) continue X1;
                else break X2;
            }
        }
        System.out.println(i+" "+j);
    }
}
```

Correct Option is : D

~~**A.**~~ 0 3

~~**B.**~~ 0 2

~~**C.**~~ 3 0

D. 3 3

~~**E.**~~ 2 2

Explanation:

The statement: `if(i < j) continue X1; else break X2;` only makes sure that the inner loop does not iterate more than once. i.e. for each iteration of i, j only takes the value of 3 and then the j loop terminates, either because of `continue X1;` or because of `break X2;`.
Now, the point to remember here is that when the loop `for(i = 0; i < 3; i++)` ends, the value of i is 3 and not 2.
Similarly, if there were no statement inside inner loop, the value of j after the end of the loop would have been 0 and not 1.

Back to Question without Answer

14. QID - 2.891 : Using Loop Constructs

What can you do to make the following code compile?

```
public class TestClass {
    public static void main(String[] args) {
        int[] values = { 10, 20, 30 };
        for( /* put code here */ ){
        }
    }
}
```

Correct Options are : A D

A. int k : values

~~**B.**~~ int k in values

~~**C.**~~ int k; k<0; k++

k must be initialized first. So it should be: int k=0; k<0; k++

D. ;;

It will cause an infinite loop, but it is valid.

~~**E.**~~ ; k<values.length;k++

k needs to be declared first.

Back to Question without Answer

15. QID - 2.1010 : Using Loop Constructs

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     for : for(int i = 0; i< 10; i++){
        for (int j = 0; j< 10; j++){
             if ( i+ j > 10 )  break for;
        }
        System.out.println( "hello");
     }
  }
}
```

Correct Option is : B

~~**A.**~~ It will print `hello` 6 times.

B. It will not compile.

~~**C.**~~ It will print `hello` 2 times.

~~**D.**~~ It will print `hello` 5 times.

~~**E.**~~ It will print `hello` 4 times.

Explanation:

Note that for is a keyword and so cannot be used as a label. But you can use any other

identifier as a label.
For example, The following code is valid even though String is a class name and String is also used as an identifier!

```
String String = "";   //This is valid.
String : for(int i = 0; i< 10; i++) //This is valid too!
{
   for (int j = 0; j< 10; j++){
        if ( i+ j > 10 )  break String;
   }
  System.out.println( "hello");
 }
```

It will print hello 2 times.

Back to Question without Answer

16. QID - 2.1083 : Using Loop Constructs

What will the following program print?

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

Correct Option is : B

~~**A.** 2~~

B. 3

~~**C.** 6~~

~~**D.** 7~~

~~E.~~ 9

Explanation:

To understand how this loop works let us put some extra print statements in the innermost loop:

```
System.out.println("i="+i+" j="+j+" k="+k);
if(k-j>0){
     System.out.println("breaking middle "+j);
     break middle;
}
counter++;
```

This is what it prints:

```
i=0 j=0 k=0
i=0 j=0 k=1
breaking middle 0
i=1 j=0 k=0
i=1 j=0 k=1
breaking middle 0
i=2 j=0 k=0
i=2 j=0 k=1
breaking middle 0
3
```

The key is that the middle loop is broken as soon as k-j becomes > 0. This happens on every second iteration of inner loop when k is 1 and j is 0. Now, when middle is broken inner cannot continue. So the next iteration of outer starts.

Back to Question without Answer

17. QID - 2.914 : Using Loop Constructs

Consider the following code:

```
public static void main(String[] args) {
   int[] values = { 10, 30, 50 };
        for( int val : values ){
            int x = 0;
            while(x<values.length){
             System.out.println(x+" "+val);
              x++;
            }
        }
    }
```

How many times is 2 printed out?

Correct Option is : D

~~**A.**~~ 0

~~**B.**~~ 1

~~**C.**~~ 2

D. 3

Explanation:

This is a simple while loop nested inside a for loop. The `for` loop loops three times - once for each value in `values` array.
Since, `values.length` is 3, `x` is incremented two times for each for loop iteration

before the condition `x<values.length` returns `false`.
Therefore, it prints:

```
0 10
1 10
2 10
0 30
1 30
2 30
0 50
1 50
2 50
```

Back to Question without Answer

18. QID - 2.960 : Using Loop Constructs

What will the following code print when compiled and run?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        A: for(int i = 0; i < 2; i++){
            B: for(int j = 0; j < 2; j++){
                C: for(int k = 0; k < 3; k++){
                    c++;
                    if(k>j) break;
                }
            }
        }
        System.out.println(c);
    }
}
```

Correct Option is : D

~~**A.**~~ 7

~~**B.**~~ 8

~~**C.**~~ 9

D. 10

~~**E.**~~ 11

Explanation:

The point to note here is that a break without any label breaks the innermost outer loop. So in this case, whenever k>j, the C loop breaks.
You should run the program and follow it step by step to understand how it progresses.

Back to Question without Answer

19. QID - 2.1042 : Using Loop Constructs

Consider the following method which is called with an argument of 7:

```
public void method1(int i){
   int j = (i*30 - 2)/100;

   POINT1 : for(;j<10; j++){
       boolean flag  = false;
       while(!flag){
 if(Math.random()>0.5) break POINT1;
       }
   }
  while(j>0){
     System.out.println(j--);
     if(j == 4) break POINT1;
   }
}
```

What will it print?
(Assume that Math.random() return a double between 0.0 and 1.0, not including 1.0)

Correct Option is : C

~~**A.**~~ It will print `1` and `2`

~~**B.**~~ It will print `1` to N where N is a random number.

C. It will not compile.

Remember that a labeled break or continue statement must always exist inside the loop where the label is declared. Here, `if(j == 4) break POINT1;` is a labelled break that is occurring in the second loop while the label POINT1 is

declared for the first loop.

D. It will throw an exception at runtime.

Back to Question without Answer

20. QID - 2.1016 : Using Loop Constructs

What will the following code snippet print?

```
int count = 0, sum = 0;
do{
        if(count % 3 == 0) continue;
        sum+=count;
}
while(count++ < 11);
System.out.println(sum);
```

Correct Option is : B

~~**A.** 49~~

B. 48

~~**C.** 37~~

~~**D.** 36~~

~~**E.** 38~~

Explanation:

1. The while condition uses post increment operator, which means count is first compared with 11 (and based on this comparison a decision is made whether to execute the loop again or not) and then incremented. So when count is 10, the condition 10<11 is true (that means the loop needs to be executed again) and count is

incremented to 11.

2. When count is completely divisible by 3, (i.e. when count is 0, 3, 6, 9) `sum+=count;` is not executed.

Thus, the result is the summation of:
1 2 4 5 7 8 10 11

Back to Question without Answer

21. QID - 2.1099 : Using Loop Constructs

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int x  = 0;
      labelA:   for (int i=10; i<0; i--){
         int j = 0;
         labelB:
         while (j < 10){
            if (j > i) break labelB;
            if (i == j){
               x++;
               continue labelA;
            }
            j++;
         }
         x--;
      }
      System.out.println(x);
   }
}
```

Correct Option is : D

~~**A.**~~ It will not compile.

~~**B.**~~ It will go in infinite loop when run.

~~**C.**~~ The program will write 10 to the standard output.

D. The program will write 0 to the standard output.

E. None of the above.

Explanation:

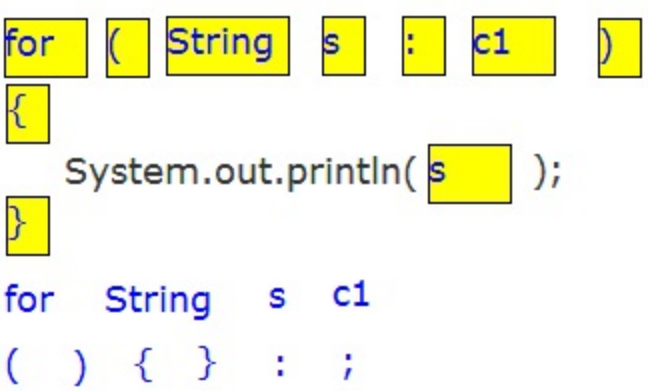
This is just a simple code that is meant to confuse you.
Notice the for statement: for `(int i=10; i<0; i--)`. i is being initialized to 10 and the test is i<0, which is false. Therefore, the control will never get inside the for loop, none of the weird code will be executed, and x will remain 0, which is what is printed.

Back to Question without Answer

22. QID - 2.1043 : Using Loop Constructs

Assuming the following declarations, write a for loop that prints each string in the collection using the given elements.

```
Collection<String> c1 = new HashSet<String>();
```



Explanation:

This illustrates the basic usage of an enhanced for loop.

Back to Question without Answer

23. QID - 2.1044 : Using Loop Constructs

What will be the output if you run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0 ; j < 1 ; ++j , i++){
         System.out.println( i + " " + j );
      }
      System.out.println( i + " " + j );
   }
}
```

Correct Option is : D

~~**A.**~~ `0 0` will be printed twice.

~~**B.**~~ `1 1` will be printed once.

~~**C.**~~ `0 1` will be printed followed by `1 2`.

D. `0 0` will be printed followed by `1 1`.

~~**E.**~~ It will print `0 0` and then `0 1`.

Explanation:

`j` will be less than `1` for only the first iteration. So, first it will print `0, 0`. Next, `i` and

`j` are incremented.

Because `j` is not less than `1` at the start of the loop, the condition fails and it comes out of the loop. Finally, it will print `1,1`.

Back to Question without Answer

24. QID - 2.1233 : Using Loop Constructs

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        boolean flag = true;
        for(int i = 0; i < 3; i++){
            while(flag){
                c++;
                if(i>c || c>5) flag = false;
            }
        }
        System.out.println(c);
    }
}
```

Correct Option is : D

~~**A.** 3~~

~~**B.** 4~~

~~**C.** 5~~

D. 6

~~**E.** 7~~

Explanation:

In the first iteration of for loop, the while loop keeps running till c becomes 6. Now, for all next for loop iteration, the while loop never runs as the flag is false. So final value of c is 6.

Back to Question without Answer

25. QID - 2.1299 : Using Loop Constructs

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int var = 20, i=0;
      do{
         while(true){
         if( i++ > var) break;
         }
      }while(i<var--);
      System.out.println(var);
   }
}
```

Correct Option is : A

A. 19

~~**B.**~~ 20

~~**C.**~~ 21

~~**D.**~~ 22

~~**E.**~~ It will enter an infinite loop.

Explanation:

When the first iteration of outer do-while loop starts, var is 20. Now, the inner loop

executes till i becomes 21.
Now, the condition for outer do-while is checked, while(22 < 20), [i is 22 because of the last i++>var check], thereby making var 19. And as the condition is false, the outer loop also ends.
So, 19 is printed.

Back to Question without Answer

26. QID - 2.1050 : Using Loop Constructs

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      for (i=1 ;  i<5  ; i++) continue;  //(1)
      for (i=0 ;       ; i++) break;       //(2)
      for (    ; i<5?false:true ;    );     //(3)
   }
}
```

Correct Option is : A

A. The code will compile without error and will terminate without problems when run.

~~B.~~ The code will fail to compile, since the `continue` can't be used this way.

~~C.~~ The code will fail to compile, since the `break` can't be used this way

~~D.~~ The code will fail to compile, since the `for` statement in the line 2 is invalid.

~~E.~~ The code will compile without error but will never terminate.

the condition part is 'false' so the control will never go inside the loop.

Explanation:

A continue statement can occur in and only in a for, while or do-while loop. A

continue statement means: Forget about the rest of the statements in the loop and start the next iteration.
So,
`for (i=1 ;  i<5  ; i++) continue;` just increments the value of i up to 5 because of i++.

`for (i=0 ;       ; i++) break;` iterates only once because of the break so the value of i becomes 0.

`for (    ; i<5?false:true ;    );` never iterates because i is less than 5 (it is 0 because of //2) and the condition expression is false!

At the end of the code, the value of i is 0.

Back to Question without Answer

27. QID - 2.1193 : Using Loop Constructs

Which of these statements are valid when occurring by themselves?

Correct Options are : B D E

~~**A.**~~ `while ( ) break ;`

> The condition expression in a while header is required.

B. `do { break ; } while (true) ;`

~~**C.**~~ `if (true) { break ; }` (When not inside a switch block or a loop)

> Cannot have break or continue in an 'if' or 'else' block.

D. `switch (1) { default : break; }`

> You can use a constant in `switch(...);`

E. `for ( ; true ; ) break ;`

Explanation:

It is not possible to break out of an if statement. But if the if statement is placed within a labelled block or a switch statement or a loop construct, the usage of break in option 3 would be valid.

Back to Question without Answer

28. QID - 2.1032 : Using Loop Constructs

What will the following code print?

```
public class BreakTest{
  public static void main(String[] args){
    int i = 0, j = 5;
    lab1 : for( ; ; i++){
      for( ; ; --j)  if( i >j ) break lab1;
    }
    System.out.println(" i = "+i+", j = "+j);
  }
}
```

Correct Option is : D

~~**A.**~~ `i = 1, j = -1`

~~**B.**~~ `i = 1, j = 4`

~~**C.**~~ `i = 0, j = 4`

D. `i = 0, j = -1`

~~**E.**~~ It will not compile.

Explanation:

The values of i and j in the inner most for loop change as follows:

```
i = 0 j = 5
i = 0 j = 4
```

```
i = 0 j = 3
i = 0 j = 2
i = 0 j = 1
i = 0 j = 0
i = 0 j = -1
```

Therefore, the final println prints `i = 0, j = -1`

Back to Question without Answer

29. QID - 2.848 : Using Loop Constructs

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        do{
            System.out.println(k);
        }while(--k>0);
    }
}
```

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 1
0

C. 2
1

> `--k>0` implies, decrement the value of k and then compare with 0. Therefore, the loop will only execute twice, printing 2 and 1.
>
> Had it been `k-->0`, it would imply, first compare k with 0, and then decrement k. In this case, the loop would execute thrice, printing 2, 1, and 0.

~~**D.**~~ 2
1

0

~~**E.**~~ It will keeping printing numbers in an infinite loop.

~~**F.**~~ It will not compile.

Back to Question without Answer

30. QID - 2.963 : Using Loop Constructs

Consider the following code snippet:

```
for(int i=INT1; i<INT2; i++){
    System.out.println(i);
}
```

INT1 and INT2 can be any two integers.

Which of the following will produce the same result?

Correct Option is : E

~~**A.**~~ `for(int i=INT1; i<INT2; System.out.println(++i));`

> Prints: 2 and 3

~~**B.**~~ `for(int i=INT1; i++<INT2; System.out.println(i));`

> Prints: 2 and 3

~~**C.**~~ `int i=INT1; while(i++<INT2) { System.out.println(i); }`

> Prints: 2 and 3

~~**D.**~~ `int i=INT1; do { System.out.println(i); }while(i++<INT2);`

> Prints: 1 2 and 3

E. None of these.

Explanation:

In such a question it is best to take a sample data such as INT1=1 and INT2=3 and execute the loops mentally. Eliminate the wrong options. In this case, the original loop will print:
=====ORIGINAL ====
1

2

Outputs of all the options are given above (Ignoring the line breaks).

Thus, none of them is same as the original.

Back to Question without Answer

31. QID - 2.1281 : Using Loop Constructs

Which of the following statements regarding 'break' and 'continue' are true?

Correct Option is : A

A. break without a label, can occur only in a switch, while, do, or for statement.

~~**B.**~~ continue without a label, can occur only in a switch, while, do, or for statement.

It cannot occur in a switch.

~~**C.**~~ break can never occur without a label.

~~**D.**~~ continue can never occur WITH a label.

~~**E.**~~ None of the above.

Explanation:

A break statement with no label attempts to transfer control to the innermost enclosing switch, while, do, or for statement; this statement, which is called the break target, then immediately completes normally. If no switch, while, do, or for statement encloses the break statement, a compile-time error occurs.

A break statement with label Identifier attempts to transfer control to the enclosing labeled statement that has the same Identifier as its label; this statement, which is called the break target, then immediately completes normally. In this case, the break target need not be a while, do, for, or switch statement.

A continue statement with no label attempts to transfer control to the innermost enclosing while, do, or for statement; this statement, which is called the continue target, then immediately ends the current iteration and begins a new one. If no while, do, or for statement encloses the continue statement, a compile-time error occurs.

A continue statement with label Identifier attempts to transfer control to the enclosing labelled statement that has the same Identifier as its label; that statement, which is called the continue target, then immediately ends the current iteration and begins a new one. The continue target must be a while, do, or for statement or a compile-time error occurs. If no labelled statement with Identifier as its label contains the continue statement, a compile-time error occurs.

Back to Question without Answer

32. QID - 2.1120 : Using Loop Constructs

What will the following program snippet print?

```
int i=0, j=11;
do{
   if(i > j) { break; }
   j--;
}while( ++i < 5);
System.out.println(i+"  "+j);
```

Correct Option is : B

~~**A.**~~ 5 5

B. 5 6

~~**C.**~~ 6 6

~~**D.**~~ 6 5

~~**E.**~~ 4 5

Explanation:

++i < 5 means, increment the value of i and then compare with 5.
Now, Try to work out the values of i and j at every iteration.
To start with, i=0 and j=11. At the time of evaluation of the while condition, i and j are as follows:

1. j = 10 and i=1 (loop will continue because i<5) (Remember that comparison will happen AFTER increment i because it is ++i and not i++.
2. j = 9 and i=2 (loop will continue because i<5).
3. j = 8 and i=3 (loop will continue because i<5).
4. j = 7 and i=4 (loop will continue because i<5).
5. j = 6 and i=5 (loop will NOT continue because i not <5).

So it will print 5 6. (It is print i first and then j).

Back to Question without Answer

33. QID - 2.847 : Using Loop Constructs

What will the following code print when compiled and run:

```
public class TestClass {

    public static void main(String[] args){
        int k = 2;
        while(--k){
            System.out.println(k);
        }
    }
}
```

Correct Option is : F

~~**A.**~~ 1

~~**B.**~~ 1
0

~~**C.**~~ 2
1

~~**D.**~~ 2
1
0

~~**E.**~~ It will keeping printing numbers in an infinite loop.

F. It will not compile.

In Java, a while or do/while construct takes an expression that returns a boolean. The expression `--i` is an integer, which is invalid and so the compilation fails.

You could change it to: `while( --i>0 ){ ... }`. In this case, `--i<0` is a boolean expression and is valid.

Back to Question without Answer

34. QID - 2.1270 : Using Loop Constructs

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      boolean b = false;
      int i = 1;
      do{
         i++ ;
      } while (b = !b);
      System.out.println( i );
   }
}
```

Correct Option is : C

~~**A.**~~ The code will fail to compile, 'while' has an invalid condition expression.

> It is perfectly valid because b = !b; returns a boolean, which is what is needed for while condition.

~~**B.**~~ It will compile but will throw an exception at runtime.

C. It will print 3.

> The loop body is executed twice and the program will print 3.

~~**D.**~~ It will go in an infinite loop.

~~**E.**~~ It will print 1.

Explanation:

Unlike the 'while(){}' loop, the 'do {} while()' loop executes at least once because the condition is checked after the iteration.

Back to Question without Answer

35. QID - 2.1327 : Using Loop Constructs

What will be the output when the following program is run?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0; j < i; ++j, i++){
         System.out.println(i + " " + j);
      }
      System.out.println(i + " " + j);
   }
}
```

Correct Option is : B

~~**A.**~~ 0 0 will be printed twice.

B. 0 0 will be printed once.

~~**C.**~~ It will keep on printing 0 0

~~**D.**~~ It will not compile.

~~**E.**~~ It will print 0 0 and then 0 1.

Explanation:

++j and i++ do not matter in this case.

The loop will not execute even once since j is not less than i at the start of the loop so the condition fails and the program will print 0 0 just once.

Back to Question without Answer

36. QID - 2.1059 : Using Loop Constructs

Identify valid for constructs...
Assume that Math.random() returns a double between 0.0 and 1.0 (not including 1.0).

Correct Options are : A C E

A.
```
for(;Math.random()<0.5;){
    System.out.println("true");
}
```

> The second expression in a for loop must return a boolean, which is happening here. So this is valid.

B.
```
for(;;Math.random()<0.5){
    System.out.println("true");
}
```

> Here, the first part (i.e. the init part) and the second part (i.e. the expression/condition part) part of the for loop are empty. Both are valid. (When the expression/condition part is emtpy, it is interpreted as true.)
>
> The third part (i.e. the update part) of the for loop does not allow every kind of statement. It allows only the following statements here: Assignment, PreIncrementExpression, PreDecrementExpression, PostIncrementExpression, PostDecrementExpression, MethodInvocation, and ClassInstanceCreationExpression. Thus, Math.random()<0.5 is not valid here, and so this will not compile.

C.
```
for(;;Math.random()){
    System.out.println("true");
}
```

This is a valid never ending loop that will keep printing true.

D.
```
for(;;){
    Math.random()<.05? break : continue;
}
```

This is an invalid use of ? : operator. Both sides of : should return the same type (excluding void). Here, break and continue do not return anything. However, the following would have been valid:
for(;Math.random()<.05? true : false;){ }

E.
```
for(;;){
    if(Math.random()<.05) break;
}
```

Explanation:

The three parts of a for loop are independent of each other. However, there are certain rules for each part. Please go through section 14.14.1 of JLS to understand it fully.

Back to Question without Answer

37. QID - 2.1057 : Using Loop Constructs

What will the following code print?

```
void crazyLoop(){
   int c = 0;
   JACK: while (c < 8){
       JILL: System.out.println(c);
       if (c > 3) break JILL; else c++;
   }
}
```

Correct Option is : A

A. It will not compile.

> Because `break JILL;` would be valid only when it is within the block of code under the scope of the label `JILL`.
> In this case, the scope of `JILL` extends only up till `System.out.println(c);` and `break JILL;` is out of the scope of the label.

~~**B.**~~ It will throw an exception at runtime.

~~**C.**~~ It will print numbers from 0 to 8

~~**D.**~~ It will print numbers from 0 to 3

~~**E.**~~ It will print numbers from 0 to 4

Back to Question without Answer

Using Operators and Decision Constructs

Exam Objectives -

Use Java operators Use parentheses to override operator precedence Test equality between strings and other objects using == and equals () Create if and if/else constructs Use a switch statement

01. QID - 2.1324

What will happen when the following program is compiled and run?

```
public class SM{
   public String checkIt(String s){
      if(s.length() == 0 || s == null){
         return "EMPTY";
      }
      else return "NOT EMPTY";
    }

    public static void main(String[] args){
      SM a = new SM();
      a.checkIt(null);
    }
}
```

Select 1 option

A. It will print `EMPTY`.

B. It will print `NOT EMPTY`.

C. It will throw `NullPointerException`.

D. It will print `EMPTY` if || is replaced with |.

Check Answer

02. QID - 2.1103

What will be printed by the following code if it is run with command line: java TestClass -0.50 ?

```
public class TestClass{
    public static double getSwitch(String str){
        return Double.parseDouble(str.substring(1, str.length()-1)
);
    }
    public static void main(String args []){
        switch(getSwitch(args[0])){
            case 0.0 : System.out.println("Hello");
            case 1.0 : System.out.println("World"); break;
            default : System.out.println("Good Bye");
        }
    }
}
```

Select 1 option

A. Hello

B. World

C. Hello World

D. Hello World Good Bye

E. None of the above.

Check Answer

03. QID - 2.1081

Which of the following code snippets will print exactly 10?

```
1.  Object t = new Integer(106);
    int k = ((Integer) t).intValue()/10;
    System.out.println(k);

2.  System.out.println(100/9.9);

3.  System.out.println(100/10.0);

4.  System.out.println(100/10);

5.  System.out.println(3 + 100/10*2-13);
```

Select 3 options

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

04. QID - 2.1014

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     Object obj1 = new Object();
     Object obj2 = obj1;
     if( obj1.equals(obj2) ) System.out.println("true");
     else  System.out.println("false");
  }
}
```

Select 1 option

A. true

B. false

C. It will not compile.

D. It will compile but throw an exception at run time.

E. None of the above.

Check Answer

05. QID - 2.1277

Which of the following statements are true?

Select 2 options

A. `System.out.println(1 + 2 + "3");` would print 33.

B. `System.out.println("1" + 2 + 3);` would print 15.

C. `System.out.println(4 + 1.0f);` would print 5.0

D. `System.out.println(5/4);` would print 1.25

E. `System.out.println('a' + 1 );` would print b.

Check Answer

06. QID - 2.1361

The following program will print

```
java.lang.ArithmeticException: / by zero
```

```
class Test{
   public static void main(String[] args){
      int d = 0;
      try{
         int i = 1 / (d* doIt());
      } catch (Exception e){
         System.out.println(e);
      }
   }
   public static int doIt() throws Exception{
      throw new Exception("Forget It");
   }
}
```

Select 1 option

A. True

B. False

Check Answer

07. QID - 2.1190

What will the following code snippet print?

```
Object t = new Integer(107);
int k = (Integer) t.intValue()/9;
System.out.println(k);
```

Select 1 option

A. 11

B. 12

C. It will not compile.

D. It will throw an exception at runtime.

Check Answer

08. QID - 2.1369

Given the following declarations, identify which statements will return `true`:

```
Integer i1 = 1;
Integer i2 = new Integer(1);
int i3 = 1;
Byte b1 = 1;
Long g1 = 1L;
```

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Select 2 options

A. `i1 == i2`

B. `i1 == i3`

C. `i1 == b1`

D. `i1.equals(i2)`

E. `i1.equals(g1)`

F. `i1.equals(b1)`

Check Answer

09. QID - 2.1108

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int k = 9, s = 5;
      switch(k){
         default :
         if( k == 10) { s = s*2; }
         else{
            s = s+4;
            break;
         }
         case 7 : s = s+3;
      }
      System.out.println(s);
   }
}
```

Select 1 option

A. 5

B. 9

C. 12

D. It will not compile.

Check Answer

10. QID - 2.1104

Consider the following lines of code:

```
Integer i = new Integer(42);
Long ln = new Long(42);
Double d = new Double(42.0);
```

Which of the following options are valid?

Select 3 options

A. `i == ln;`

B. `ln == d;`

C. `i.equals(d);`

D. `d.equals(ln);`

E. `ln.equals(42);`

Check Answer

11. QID - 2.1259

Note: Although Wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of Wrapper classes.

What will be the output of the following program?

```
public class EqualTest{
   public static void main(String args[]){
      Integer i = new Integer(1) ;
      Long m = new Long(1);
      if( i.equals(m)) System.out.println("equal");    // 1
      else System.out.println("not equal");
   }
}
```

Select 1 option

A. equal

B. not equal

C. Compile time error at //1

D. Runtime error at //1

E. None of the above.

Check Answer

12. QID - 2.1111

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 == false){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print `true`

C. It will print `false`

D. Runtime error.

E. It will print nothing.

Check Answer

13. QID - 2.1196

Consider the following code snippet:

```
XXXX m ;
  switch( m ){
     case 32  : System.out.println("32");   break;
     case 64  : System.out.println("64");   break;
     case 128 : System.out.println("128");  break;
  }
```

What type can 'm' be of so that the above code compiles and runs as expected ?

Select 3 options

A. `int m;`

B. `long m;`

C. `char m;`

D. `byte m;`

E. `short m;`

Check Answer

14. QID - 2.930

Expression `(s instanceof java.util.Date)` will return false if 's' was declared as a variable of class `java.lang.String`.

Select 1 option

A. True

B. False

Check Answer

15. QID - 2.1045

Which line, if any, will give a compile time error ?

```
void test(byte x){
   switch(x){
      case 'a':    // 1
      case 256:    // 2
      case 0:      // 3
      default :    // 4
      case 80:     // 5
   }
}
```

Select 1 option

A. Line 1 as 'a' is not compatible with byte.

B. Line 2 as 256 cannot fit into a byte.

C. No compile time error but a run time error at line 2.

D. Line 4 as the default label must be the last label in the switch statement.

E. There is nothing wrong with the code.

Check Answer

16. QID - 2.986

The following method will compile and run without any problems.

```
public void switchTest(byte x){
   switch(x){
      case 'b':   // 1
      default :   // 2
      case -2:    // 3
      case 80:    // 4
   }
}
```

Select 1 option

A. True

B. False

Check Answer

17. QID - 2.860

What will the following code print when run?

```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

Select 1 option

A. bat
big bat

B. big bat
none

C. big bat

D. bat

E. The code will not compile.

Check Answer

18. QID - 2.1180

The following code snippet will not compile:

```
int i = 10;
System.out.println( i<20 ? out1() : out2() );
```

Assume that out1 and out2 have method signature: public void out1(); and public void out2();

Select 1 option

A. True

B. False

Check Answer

19. QID - 2.1125

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Given the following declarations, identify which statements will return true or false.

```
Integer i1 = 1;
Integer i2 = new Integer(1);
int i3 = 1;
Byte b1 = 1;
Long g1 = 1L;
```

1. i1 == i2
2. i1 == i3
3. i1 == b1
4. i1.equals(i2)
5. i1.equals(g1)
6. i1.equals(b1)

true

false

Will not compile

Exception at runtime

Check Answer

20. QID - 2.1089

Drag and drop valid operators (shown in blue) in yellow boxes.
= operator can be used more than once.

`=` `+=` `*=`

```
public class TestClass
{
   public static void main(String[] args)
   {
     Short k = 9;     Integer i = 9;  Boolean b = false;
     char c = 'a';    String str = "123";

     i      [ ]      (int) k.shortValue();

     str    [ ]      b;

     b      [ ]      !b;

     c      [ ]      i;

   }
}
```

Check Answer

21. QID - 2.994

Assume that a, b, and c refer to instances of primitive wrapper classes. Which of the following statements are correct?

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Select 2 options

A. a.equals(a) will always return true.

B. b.equals(c) may return false even if c.equals(b) returns true.

C. a.equals(b) returns same as a == b.

D. a.equals(b) throws an exception if they refer to instances of different classes.

E. a.equals(b) returns false if they refer to instances of different classes.

Check Answer

22. QID - 2.1237

Given:

```
enum Season { SUMMER, WINTER, SPRING, FALL }
```

What will the following code print?

```
Season s = Season.SPRING;
    switch(s){
        case SUMMER : System.out.println("SUMMER");
        case default : System.out.println("SEASON");
        case WINTER :  System.out.println("WINTER");
    }
```

Select 1 option

A. `SEASON`

B. `SEASON`
`WINTER`

C. It will not compile.

D. It will not print anything.

Check Answer

23. QID - 2.927

What will the following code print ?

```
class Test{
   public static void main(String[] args){
      int k = 1;
      int[] a = { 1 };
      k += (k = 4) * (k + 2);
      a[0] += (a[0] = 4) * (a[0] + 2);
      System.out.println( k + " , " + a[0]);
   }
}
```

Select 1 option

A. It will not compile.

B. 4 , 4

C. 25 , 25

D. 13 , 13

E. None of the above.

Check Answer

24. QID - 2.1087

Which of the given lines can be inserted at //1 of the following program ?

```
public class TestClass{
   public static void main(String[] args){
     short s = 9;
     //1
   }
}
```

Select 2 options

A. `Short k = new Short(9); System.out.println(k instanceof Short);`

B. `System.out.println(s instanceof Short);`

C. `Short k = 9; System.out.println( k instanceof s);`

D. `int i = 9; System.out.println(s == i);`

E. `Boolean b = s instanceof Number;`

F. `Short k = 9; Integer i = 9; System.out.println(k == i);`

G. `Integer i = 9; System.out.println( s == i );`

Check Answer

25. QID - 2.1082

What will be the output when the following class is compiled and run?

```
class ScopeTest{
   static int x = 5;
   public static void main(String[] args){
      int x  = ( x=3 ) * 4;  // 1
      System.out.println(x);
   }
}
```

Select 1 option

A. It will not compile because line //1 cannot be parsed correctly.

B. It will not compile because x is used before initialization.

C. It will not compile because there is an ambiguous reference to x.

D. It will print 12.

E. It will print 3 .

Check Answer

26. QID - 2.859

What will the following code print when run?

```
public class TestClass {

    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "c" : System.out.println( "cat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("c");
    }
}
```

Select 1 option

A. apple
cat
none

B. apple
cat

C. cat
none

D. cat

Check Answer

27. QID - 2.1164

Which of the following statements will compile without any error?

Select 4 options

A. `System.out.println("a"+'b'+63);`

B. `System.out.println("a"+63);`

C. `System.out.println('b'+new Integer(63));`

D. `String s = 'b'+63+"a";`

E. `String s = 63 + new Integer(10);`

Check Answer

28. QID - 2.1061

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i, j, k;
      i = j = k = 9;
      System.out.println(i);
   }
}
```

Select 2 options

A. The code will not compile because unlike in c++, operator '=' cannot be chained i.e. a = b = c = d is invalid.

B. The code will not compile as 'j' is being used before getting initialized.

C. The code will compile correctly and will display '9' when run.

D. The code will not compile as 'j' and 'i' are being used before getting initialized.

E. All the variables will get a value of 9.

Check Answer

29. QID - 2.1039

What will the following method return if called with an argument of 7?

```
public int transformNumber(int n){
   int radix = 2;
   int output = 0;
   output += radix*n;
   radix = output/radix;
   if(output<14){
       return output;
   }
   else{
       output = output*radix/2;
       return output;
   }
   else {
       return output/2;
   }
}
```

Select 1 option

A. 7

B. 14

C. 49

D. Compilation fails.

Check Answer

30. QID - 2.1335

Consider the code shown below:

```
public class TestClass{
  public static int switchTest(int k){
     int j = 1;
     switch(k){
        case 1: j++;
        case 2: j++;
        case 3: j++;
        case 4: j++;
        case 5: j++;
        default : j++;
     }
     return j + k;
  }
  public static void main(String[] args){
     System.out.println( switchTest(4) );
  }
}
```

What will it print when compiled and run?

Select 1 option

A. 5

B. 6

C. 7

D. 8

E. 9

Check Answer

31. QID - 2.1156

What will the following class print when executed?

```
class Test{
    static boolean a;
    static boolean b;
    static boolean c;
    public static void main (String[] args){
        boolean bool = (a = true) || (b = true) && (c = true);
        System.out.print(a + ", " + b + ", " + c);
    }
}
```

Select 1 option

A. true, false, true

B. true, true, false

C. true, false, false

D. true, true, true

Check Answer

32. QID - 2.965

Consider the following program:

```
public class TestClass{
   public static void main(String[] args)  {     calculate(2);     }
   public static void calculate(int x){
      String val;
      switch(x){
         case 2:
         default:
         val = "def";
      }
      System.out.println(val);
   }
}
```

What will happen if you try to compile and run the program?

Select 2 options

A. It will not compile saying that variable `val` may not have been initialized..

B. It will compile and print `def`

C. As such it will not compile but it will compile if `calculate(2);` is replaced by `calculate(3);`

D. It will compile for any int values in `calculate(...);`

Check Answer

33. QID - 2.1105

Consider the following method...

```
public void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   System.out.println("True False");
   else         // 3
   System.out.println("True True");
   else         // 4
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Select 3 options

A. If run with an argument of 'false', it will print 'False False'

B. If run with an argument of 'false', it will print 'True True'

C. If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

E. It will not compile.

Check Answer

34. QID - 2.1238

What will the following code print when run without any arguments ...

```
public class TestClass {

    public static int m1(int i){
        return ++i;
    }

    public static void main(String[] args) {

        int k = m1(args.length);
        k += 3 + ++k;
        System.out.println(k);
    }

}
```

Select 1 option

A. It will throw `ArrayIndexOutofBoundsException`.

B. It will throw `NullPointerException`.

C. `6`

D. `5`

E. `7`

F. 2

G. None of these.

Check Answer

35. QID - 2.1266

Which of the following statements are true?

Select 3 options

A. The condition expression in an if statement can contain method calls.

B. If a and b are of type boolean, the expression (a = b) can be used as the condition expression of an if statement.

C. An if statement can have either an 'if' clause or an 'else' clause.

D. The statement : if (false) ; else ; is illegal.

E. Only expressions which evaluate to a boolean value can be used as the condition in an if statement.

Check Answer

36. QID - 2.1282

What, if anything, is wrong with the following code?

```
void test(int x){
   switch(x){
      case 1:
      case 2:
      case 0:
      default :
      case 4:
   }
}
```

Select 1 option

A. Data Type of 'x' is not valid to be used as an expression for the switch clause.

B. The case label 0 must precede case label 1.

C. Each case section must end with a break keyword.

D. The default label must be the last label in the switch statement.

E. There is nothing wrong with the code.

Check Answer

37. QID - 2.1269

Which of the following will not give any error at compile time and run time?

Select 4 options

A. `if (8 == 81) {}`

B. `if (x = 3) {}` // assume that x is an int

C. `if (true) {}`

D. `if (bool = false) {}` //assume that bool is declared as a boolean

E. `if (x == 10 ? true:false) { }` // assume that x is an int

Check Answer

38. QID - 2.1212

Which of the following expressions will evaluate to true if preceded by the following code?

```
String a = "java";
    char[] b = { 'j', 'a', 'v', 'a' };
    String c = new String(b);
    String d = a;
```

Select 3 options

A. `(a == d)`

B. `(b == d)`

C. `(a == "java")`

D. `a.equals(c)`

Check Answer

39. QID - 2.1119

Given:

```
public class Switcher{

   public static void main(String[] args){
       switch(Integer.parseInt(args[1]))  //1
       {
          case 0 :
             boolean b = false;
             break;

          case 1 :
             b = true; //2
             break;
       }

       if(b) System.out.println(args[2]);
   }
}
```

What will the above program print if compiled and run using the following command line:

```
java Switcher 1 2 3
```

Select 1 option

A. It will print 1

B. It will print 2

C. It will print 3

D. It will not print anything.

E. It will not compile because of //1.

F. It will not compile because of //2.

G. It will not compile for some other reason.

Check Answer

40. QID - 2.932

Which of the following statements are true?

Select 2 options

A. The modulus operator % can only be used with integer operands.

B. & can have integral as well as boolean operands.

C. The arithmetic operators *, / and % have the same level of precedence.

D. && can have integer as well as boolean operands.

E. ~ can have integer as well as boolean operands.

Check Answer

41. QID - 2.1342

Consider that `str` is a variable of class `java.lang.String`.
Which of the following lines of code may throw a `NullPointerException` in certain situations?

Or a tougher version of the question could be :
Which of the following lines of code are not an example of robust design ?

Select 3 options

A. `if ( (str != null) | ( i == str.length() ) )`

B. `if ( (str == null) | ( i == str.length() ) )`

C. `if ( (str != null) || (i == str.length() ) )`

D. `if ( (str == null) || (i == str.length() ) )`

Check Answer

42. QID - 2.1278

Which of the following four constructs are valid?

1.
```
   switch(5)
  {
      default :
   }
```

2.
```
   switch(5)
   {
      default : break;
   }
```

3.
```
  switch(8);
```

4.
```
 int x = 0;
 switch(x){
 }
```

Select 1 option

A. 1, 3

B. 1, 2, 3

C. 3, 4

D. 1, 2, 4

E. All are valid.

Check Answer

43. QID - 2.1038

What will be the output of the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true ;
      boolean bool2 = false;
      boolean bool  = false;
      bool = ( bool2 &  method1(i++) ); //1
      bool = ( bool2 && method1(i++) ); //2
      bool = ( bool1 |  method1(i++) ); //3
      bool = ( bool1 || method1(i++) ); //4
      System.out.println(i);
   }
   public static boolean method1(int i){
       return i>0 ? true : false;
   }
}
```

Select 1 option

A. It will print 1.

B. It will print 2.

C. It will print 3.

D. It will print 4.

E. It will print 0.

Check Answer

44. QID - 2.948

Which statements about the output of the following programs are true?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true;
      boolean bool2 = false;
      boolean bool  = false;
      bool = (bool2 &  method1("1"));  //1
      bool = (bool2 && method1("2"));  //2
      bool = (bool1 |  method1("3"));  //3
      bool = (bool1 || method1("4"));  //4
   }
   public static boolean method1(String str){
      System.out.println(str);
      return true;
   }
}
```

Select 2 options

A. 1 will be the part of the output.

B. 2 will be the part of the output.

C. 3 will be the part of the output.

D. 4 will be the part of the output.

E. None of the above

Check Answer

45. QID - 2.977

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 != b2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print true;

C. It will print false;

D. Runtime error.

E. It will print nothing.

Check Answer

46. QID - 2.1065

What will the following code print?

```
boolean flag = true;
if(flag = false){
   System.out.println("1");
}else if(flag){
   System.out.println("2");
}else if(!flag){
   System.out.println("3");
}else    System.out.println("4");
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. Compilation error.

Check Answer

47. QID - 2.1084

Which of the following are valid operators in Java?

Select 4 options

A. !

B. ~

C. &

D. %=

E. $

Check Answer

48. QID - 2.1239

What letters will be printed by this program?

```
public class ForSwitch{
    public static void main(String args[]){
        char i;
        LOOP: for (i=0;i<5;i++){
            switch(i++){
                case '0': System.out.println("A");
                case 1: System.out.println("B"); break LOOP;
                case 2: System.out.println("C"); break;
                case 3: System.out.println("D"); break;
                case 4: System.out.println("E");
                case 'E' : System.out.println("F");
            }
        }
    }
}
```

Select 2 options

A. A

B. B

C. C

D. D

E. F

Check Answer

49. QID - 2.1313

What is the result of executing the following code when the value of i is 5:

```
switch (i){
    default:
    case 1:
        System.out.println(1);
    case 0:
        System.out.println(0);
    case 2:
        System.out.println(2);
        break;
    case 3:
        System.out.println(3);
}
```

Select 1 option

A. It will print `1 0 2`

B. It will print `1 0 2 3`

C. It will print `1 0`

D. It will print `1`

E. Nothing will be printed.

Check Answer

50. QID - 2.1040

What will the following class print ?

```
class InitTest{
   public static void main(String[] args){
      int a = 10;
      int b = 20;
      a += (a = 4);
      b = b + (b = 5);
      System.out.println(a+ ",  "+b);
   }
}
```

Select 1 option

A. It will print 8, 25

B. It will print 4, 5

C. It will print 14, 5

D. It will print 4, 25

E. It will print 14, 25

Check Answer

51. QID - 2.958

Consider the following code:

```
public class Conversion{
   public static void main(String[] args){
     int i = 1234567890;
     float f = i;
     System.out.println(i - (int)f);
   }
}
```

What will it print when run?

Select 1 option

A. It will print 0.

B. It will not print 0.

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

52. QID - 2.1173

Consider the following class :

```
public class Test{
   public static void main(String[] args){
      if (args[0].equals("open"))
         if (args[1].equals("someone"))
            System.out.println("Hello!");
      else System.out.println("Go away "+ args[1]);
    }
}
```

Which of the following statements are true if the above program is run with the command line :
java Test closed

Select 1 option

A. It will throw `ArrayIndexOutOfBoundsException` at runtime.

B. It will end without exceptions and will print nothing.

C. It will print `Go away`

D. It will print `Go away` and then will throw `ArrayIndexOutOfBoundsException`.

E. None of the above.

Check Answer

53. QID - 2.1267

Which of the following implementations of a `max()` method will correctly return the largest value?

Select 1 option

A.
```
int max(int x, int y){
    return(  if(x > y){ x; } else{ y; }  );
}
```

B.
```
int max(int x, int y){
    return( if(x > y){ return x; }  else{ return y; } );
}
```

C.
```
int max(int x, int y){
    switch(x < y){
       case true:
              return y;
       default :
              return x;
    };
}
```

D.
```
int max(int x, int y){
     if (x > y)  return x;
     return y;
}
```

E. None of the above.

Check Answer

54. QID - 2.949

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2 = false;
if (b2 != b1 = !b2){
   System.out.println("true");
}
else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print `true`.

C. It will print `false`.

D. Runtime error.

E. It will print nothing.

Check Answer

55. QID - 2.1157

What will be the result of attempting to compile and run the following code?

```
class SwitchTest{
   public static void main(String args[]){
      for ( int i = 0 ; i < 3 ; i++){
         boolean flag  = false;
         switch (i){
            flag  = true;
         }
         if ( flag )  System.out.println( i );
      }
   }
}
```

Select 1 option

A. It will print 0, 1 and 2.

B. It will not print anything.

C. Compilation error.

D. Runtime error.

E. None of the above.

Check Answer

56. QID - 2.1240

Given:

```
byte b = 1;
char c = 1;
short s = 1;
int i = 1;
```

which of the following expressions are valid?

Select 3 options

A. `s = b * b ;`

B. `i = b << b ;`

C. `s <<= b ;`

D. `c = c + b ;`

E. `s += i ;`

Check Answer

57. QID - 2.1317

Which of the following are NOT valid operators in Java?

Select 4 options

A. sizeof

B. <<<

C. instanceof

D. mod

E. equals

Check Answer

58. QID - 2.1071

What will be the output of the following code snippet?

```
int a = 1;
int[] ia = new int[10];
int b = ia[a];
int c = b + a;
System.out.println(b = c);
```

Select 1 option

A. 0

B. 1

C. 2

D. true

E. false

Check Answer

59. QID - 2.1070

What is the result of executing the following fragment of code:

```
boolean b1 = false;
int i1 = 2;
int i2 = 3;
if (b1 = i1 == i2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Select 1 option

A. Compile time error.

B. It will print true

C. It will print false

D. Runtime error.

E. It will print nothing.

Check Answer

60. QID - 2.1271

What will be the result of attempting to compile and run the following code?

```
public class PromotionTest{
   public static void main(String args[]){
      int i = 5;
      float f = 5.5f;
      double d = 3.8;
      char c = 'a';
      if (i == f) c++;
      if (((int) (f + d)) == ((int) f + (int) d)) c += 2;
      System.out.println(c);
   }
}
```

Select 1 option

A. The code will fail to compile.

B. It will print d.

C. It will print c.

D. It will print b

E. It will print a.

Check Answer

61. QID - 2.1325

What will be the result of attempting to compile and run the following class?

```
public class IfTest{
   public static void main(String args[]){
      if (true)
      if (false)
      System.out.println("True False");
      else
      System.out.println("True True");
   }
}
```

Select 1 option

A. The code will fail to compile because the syntax of the `if` statement is not correct.

B. The code will fail to compile because the values in the condition bracket are invalid.

C. The code will compile correctly and will not display anything.

D. The code will compile correctly and will display `True True`.

E. The code will compile correctly but will display `True False`

Check Answer

62. QID - 2.1088

Consider:
`o1` and `o2` denote two object references to two different objects of same class.
Which of the following statements are true?

Select 2 options

A. `o1.equals(o2)` will always be false.

B. `o1.hashCode() == o2.hashCode()` will always be false.

C. `o1 == o2` will always be false.

D. Nothing can be said about `o1.equals(o2)` regarding what it will return based on the given information.

E. Nothing can be said about `o1 == o2`.

Check Answer

63. QID - 2.1353

Which of the lines will cause a compile time error in the following program?

```
public class MyClass{
   public static void main(String args[]){
      char c;
      int i;
      c = 'a';//1
      i = c;  //2
      i++;    //3
      c = i;  //4
      c++;    //5
   }
}
```

Select 1 option

A. line 1

B. line 2

C. line 3

D. line 4

E. line 5

Check Answer

64. QID - 2.1183

Given the following LOCs:

```
int rate = 10;
XXX amount = 1 - rate/100*1 - rate/100;
```

What can XXX be?

Select 1 option

A. only int or long

B. only long or double

C. only double

D. double or float

E. long or double but not int or float.

F. int, long, float or double

Check Answer

65. QID - 2.1179

The following code snippet will print 'true'.

```
short s = Short.MAX_VALUE;
char c = s;
System.out.println( c == Short.MAX_VALUE);
```

Select 1 option

A. True

B. False

Check Answer

66. QID - 2.1177

Which of the following statements are true?

Select 1 option

A. For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `true`.

B. For any non-null reference `o1`, the expression `(o1 instanceof Object)` will always yield `true`.

C. For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `false`.

D. For any non-null reference `o1`, the expression `(o1 instanceof Object)` may yield `false`.

E. None of the above.

Check Answer

67. QID - 2.973

Which operators will always evaluate all the operands?

Select 2 options

A. &&

B. |

C. ||

D. ? :

E. %

Check Answer

68. QID - 2.1310

Consider the following method...

```
public static void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   if (flag)    //3
   System.out.println("False True");
   else         //4
   System.out.println("True False");
   else         //5
   System.out.println("True True");
   else         //6
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Select 2 options

A. If run with an argument of 'false', it will print 'False False'

B. If run with an argument of 'false', it will print 'True True'

C. If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

E. It will not compile.

Check Answer

69. QID - 2.980

Given the following class definitions, the expression

```
(obj instanceof A) && ! (obj instanceof C) && ! (obj instanceof D)
```

correctly identifies whether the object referred to by obj was created by instantiating class B rather than classes A, C and D?

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}
```

Select 1 option

A. True

B. False

Check Answer

Using Operators and Decision Constructs (Answered)

01. QID - 2.1324 : Using Operators and Decision Constructs

What will happen when the following program is compiled and run?

```
public class SM{
   public String checkIt(String s){
      if(s.length() == 0 || s == null){
         return "EMPTY";
      }
      else return "NOT EMPTY";
    }

    public static void main(String[] args){
      SM a = new SM();
      a.checkIt(null);
    }
}
```

Correct Option is : C

~~**A.**~~ It will print `EMPTY`.

~~**B.**~~ It will print `NOT EMPTY`.

C. It will throw `NullPointerException`.

> Because the first part of the expression `(s.length() == 0)` is trying to call a method on s, which is `null`. The check `s == null` should be done before calling a method on the reference.

~~**D.**~~ It will print `EMPTY` if || is replaced with |.

> In this case, replacing || with | will not make any difference because `s.length()`

will anyway be called before checking whether `s` is `null` or not. The right expression would be:

```
if( s == null || s.length() == 0) { ... }
```

In this case, || being a short circuit expression, `s.length() == 0` will not be called if `s == null` returns `true`. Hence, no `NullPointerExpression` will be thrown.

Back to Question without Answer

02. QID - 2.1103 : Using Operators and Decision Constructs

What will be printed by the following code if it is run with command line: java TestClass -0.50 ?

```
public class TestClass{
    public static double getSwitch(String str){
        return Double.parseDouble(str.substring(1, str.length()-1)
);
    }
    public static void main(String args []){
        switch(getSwitch(args[0])){
            case 0.0 : System.out.println("Hello");
            case 1.0 : System.out.println("World"); break;
            default : System.out.println("Good Bye");
        }
    }
}
```

Correct Option is : E

~~**A.**~~ Hello

~~**B.**~~ World

~~**C.**~~ Hello World

~~**D.**~~ Hello World Good Bye

E. None of the above.

Explanation:

Observe that the method `getSwitch()` has been declared to return a double. Its return value is being used in the switch() statement. Therefore, the program will not even compile because double/float/long/boolean cannot be used in switch(...) statement.

Back to Question without Answer

03. QID - 2.1081 : Using Operators and Decision Constructs

Which of the following code snippets will print exactly 10?

```
1.  Object t = new Integer(106);
    int k = ((Integer) t).intValue()/10;
    System.out.println(k);

2.  System.out.println(100/9.9);

3.  System.out.println(100/10.0);

4.  System.out.println(100/10);

5.  System.out.println(3 + 100/10*2-13);
```

Correct Options are : A D E

A. 1

~~B.~~ 2

> Since one of the operands (9.9) is a double, it wil perform a real division and will print 10.1010101010101

~~C.~~ 3

> Since one of the operands (10.0) is a double, it will perform a real division and will print 10.0

D. 4

E. 5

Explanation:

1. `int k = ((Integer) t).intValue()/10;`
Since both the operands of / are ints, it is a integer division. This means the resulting value is truncated (and not rounded). Therefore, the above statement will print 10 and not 11.
5. `3 + 100/10*2-13` will be parsed as: `3 + (100/10)*2-13`. This is because the precedence of / and * is same (and is higher than + and -) and since the expression is evaluated from left to right, the operands are grouped on first come first served basis. [This is not the right terminology but you will be able to answer the questions if you remember this rule.]

Back to Question without Answer

04. QID - 2.1014 : Using Operators and Decision Constructs

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
     Object obj1 = new Object();
     Object obj2 = obj1;
     if( obj1.equals(obj2) ) System.out.println("true");
     else  System.out.println("false");
  }
}
```

Correct Option is : A

A. true

~~**B.**~~ false

~~**C.**~~ It will not compile.

~~**D.**~~ It will compile but throw an exception at run time.

~~**E.**~~ None of the above.

Explanation:

Object class's `equals()` method just checks whether the two references are pointing to the same location or not. In this case they really are pointing to the same location because of `obj2 = obj1;` so it returns `true`.

Back to Question without Answer

05. QID - 2.1277 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Options are : A C

A. `System.out.println(1 + 2 + "3");` would print 33.

> operator + is left associative so evaluation of (1 + 2 + "3") is as follows: (1 + 2) + "3" -> 3 + "3" -> "33".

~~B.~~ `System.out.println("1" + 2 + 3);` would print 15.

> evaluation of ("1" + 2 + 3) is as follows: ("1" + 2) + 3 -> "12" + 3 -> "123".

C. `System.out.println(4 + 1.0f);` would print 5.0

> (4 + 1.0f) evaluates as 4.0f + 1.0f ->5.0f -> 5.0

~~D.~~ `System.out.println(5/4);` would print 1.25

> (5/4) performs integer division because both 5 and 4 are integers, resulting in the value 1.

~~E.~~ `System.out.println('a' + 1 );` would print b.

> Both operands in the expression ('a' + 1) will be promoted to int => 97 + 1 = 98

Explanation:

All operands of type byte, char or short are promoted AT LEAST to an int before

performing mathematical operations. If one of the operands is larger than an int then the other one is promoted to the same type.
Note that `System.out.println((float)5/4);` will print `1.25`. If you remove the explicit cast (float), it will print 1.

Back to Question without Answer

06. QID - 2.1361 : Using Operators and Decision Constructs

The following program will print

```
java.lang.ArithmeticException: / by zero
```

```
class Test{
   public static void main(String[] args){
      int d = 0;
      try{
         int i = 1 / (d* doIt());
      } catch (Exception e){
         System.out.println(e);
      }
   }
   public static int doIt() throws Exception{
      throw new Exception("Forget It");
   }
}
```

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

It will print `Forget It` because before the division can take place `doIt()` will throw an exception.

Back to Question without Answer

07. QID - 2.1190 : Using Operators and Decision Constructs

What will the following code snippet print?

```
Object t = new Integer(107);
int k = (Integer) t.intValue()/9;
System.out.println(k);
```

Correct Option is : C

~~**A.**~~ 11

~~**B.**~~ 12

C. It will not compile.

~~**D.**~~ It will throw an exception at runtime.

Explanation:

Compiler will complain that the method intValue() is not available in Object. This is because the . operator has more precedence than the cast operator. So you have to write it like this:

```
int k = ((Integer) t).intValue()/9;
```

Now, since both the operands of / are ints, it is an integer division. This means the resulting value is truncated (and not rounded). Therefore, the above statement will print 11 and not 12.

Back to Question without Answer

08. QID - 2.1369 : Using Operators and Decision Constructs

Given the following declarations, identify which statements will return `true`:

```
Integer i1 = 1;
Integer i2 = new Integer(1);
int i3 = 1;
Byte b1 = 1;
Long g1 = 1L;
```

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Correct Options are : B D

~~**A.**~~ `i1 == i2`

> This will return false because both are pointing to different objects.

B. `i1 == i3`

> This will return true because one operand is a primitive int, so the other will be unboxed and then the value will be compared.

~~**C.**~~ `i1 == b1`

> This will not compile because type of i1 and b1 references are classes that are not in the same class hierarchy. So the compiler figures out at compile time itself these two references cannot ever point to the same object.

D. `i1.equals(i2)`

This will return true because both are Integer objects and both have the value 1.

E. `i1.equals(g1)`

This will return false because they are pointing to objects of different types.

Signature of equals method is : boolean equals(Object o);
Thus, it can take any object as a parameter and so there will be no compilation error.

Further, The equals method of all wrapper classes first checks if the two object are of same class or not. If not, they immediately return false.

F. `i1.equals(b1)`

This will return false because they are pointing to objects of different types.

Back to Question without Answer

09. QID - 2.1108 : Using Operators and Decision Constructs

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int k = 9, s = 5;
      switch(k){
         default :
         if( k == 10) { s = s*2; }
         else{
            s = s+4;
            break;
         }
         case 7 : s = s+3;
      }
      System.out.println(s);
   }
}
```

Correct Option is : B

~~**A.**~~ 5

B. 9

Since 9 does not match any of the case labels, it is accepted by default block. In this block, the else part is executed, which sets s to the value of s+4, i.e. 9. Since there is a break in the else block, case 7: is not executed.

~~**C.**~~ 12

~~**D.**~~ It will not compile.

Back to Question without Answer

10. QID - 2.1104 : Using Operators and Decision Constructs

Consider the following lines of code:

```
Integer i = new Integer(42);
Long ln = new Long(42);
Double d = new Double(42.0);
```

Which of the following options are valid?

Correct Options are : C D E

~~**A.**~~ `i == ln;`

> This will fail at compile time

~~**B.**~~ `ln == d;`

> This will fail at compile time

C. `i.equals(d);`

D. `d.equals(ln);`

E. `ln.equals(42);`

> Due to auto-boxing int 42 is converted into an Integer object containing 42. So this is valid. It will return false though because ln is a Long and 42 is boxed into an Integer.

Explanation:

The concept to understand here is as follows -
If the compiler can figure out that something can NEVER happen, then it flags an error. In this question, the compiler knows that ln, i or d can never point to the same object in any case because they are references to different classes of objects that have no relation (superclass/subclass) between themselves.

Back to Question without Answer

11. QID - 2.1259 : Using Operators and Decision Constructs

Note: Although Wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of Wrapper classes.

What will be the output of the following program?

```
public class EqualTest{
   public static void main(String args[]){
      Integer i = new Integer(1) ;
      Long m = new Long(1);
      if( i.equals(m)) System.out.println("equal");    // 1
      else System.out.println("not equal");
   }
}
```

Correct Option is : B

~~**A.** equal~~

B. not equal

~~**C.** Compile time error at //1~~

~~**D.** Runtime error at //1~~

~~**E.** None of the above.~~

Explanation:

Signature of equals method is : `boolean equals(Object o);` So it can take any

object.
The equals methods of all wrapper classes first check if the two object are of same class or not. If not, they immediately return `false`. Hence it will print `not equal`.

Back to Question without Answer

12. QID - 2.1111 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 == false){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Correct Option is : B

~~**A.**~~ Compile time error.

B. It will print `true`

~~**C.**~~ It will print `false`

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

All that `if()` needs is a `boolean`, now `b1 == false` returns `true`, which is a `boolean` and since `b2 = true` is an expression and every expression has a return value (which is the Left Hand Side of the expression), it returns `true`, which is again a `boolean`.

FYI: the return value of expression `i = 10;` is `10` (an int).

Back to Question without Answer

13. QID - 2.1196 : Using Operators and Decision Constructs

Consider the following code snippet:

```
XXXX m ;
  switch( m ){
     case 32  : System.out.println("32");   break;
     case 64  : System.out.println("64");   break;
     case 128 : System.out.println("128");  break;
  }
```

What type can 'm' be of so that the above code compiles and runs as expected ?

Correct Options are : A C E

A. `int m;`

> m can hold all the case values.

~~**B.**~~ `long m;`

> long, float, double, and boolean can never be used as a switch variable.

C. `char m;`

> m can hold all the case values.

~~**D.**~~ `byte m;`

> m will not be able to hold 128. a byte's range is -128 to 127.

E. `short m;`

m can hold all the case values.

Explanation:

Three rules must be remembered about switch statement:
1. Only byte, char, short, int, and enum values can be used as types of a switch variable. (Java 7 allows String as well.)
2. The switch variable must be big enough to hold all the case constants.
So, if switch variable is of type char, then none of the case constants can be greater than 65535 because char's range is from 0 to 65535.
3. All case labels should be COMPILE TIME CONSTANTS.

Back to Question without Answer

14. QID - 2.930 : Using Operators and Decision Constructs

Expression `(s instanceof java.util.Date)` will return false if 's' was declared as a variable of class `java.lang.String`.

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

It will not even compile because the compiler knows that 's' (which is declared as of class `String`) can NEVER refer to an object of class `java.util.Date`. So, it will not accept this code.
Had 's' been declared as a variable of type `Object`, this code would have compiled because compiler sees that at run time it is possible for s to refer to an object of class `Date`.

Back to Question without Answer

15. QID - 2.1045 : Using Operators and Decision Constructs

Which line, if any, will give a compile time error ?

```
void test(byte x){
   switch(x){
      case 'a':    // 1
      case 256:    // 2
      case 0:      // 3
      default :    // 4
      case 80:     // 5
   }
}
```

Correct Option is : B

~~**A.**~~ Line 1 as 'a' is not compatible with byte.

> int value of 'a' can easily fit into a byte.

B. Line 2 as 256 cannot fit into a byte.

~~**C.**~~ No compile time error but a run time error at line 2.

~~**D.**~~ Line 4 as the default label must be the last label in the switch statement.

> Any order of case statements is valid.

~~**E.**~~ There is nothing wrong with the code.

Explanation:

Every case constant expression in a switch block must be assignable to the type of switch expression. Meaning :

```
byte by = 10;
switch(by){
     300 :  //some code;
     56 :   //some code;
}
```

This will not compile as 300 is not assignable to 'by ' which can only hold values from -128 to 127. This gives compile time error as the compiler detects it while compiling. The use of break keyword is not mandatory, and without it the control will simply fall through the labels of the switch statement.

Back to Question without Answer

16. QID - 2.986 : Using Operators and Decision Constructs

The following method will compile and run without any problems.

```
public void switchTest(byte x){
   switch(x){
      case 'b':   // 1
      default :   // 2
      case -2:    // 3
      case 80:    // 4
   }
}
```

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

The following types can be used as a switch variable:

`byte`, `char`, `short`, `int`, `String`, and enums. Note that `long`, `float`, `double`, and `boolean` are not allowed.

All the case constants should be assignable to the switch variable type. i.e. had there been a case label of 128 (case 128 : //some code), it would not have compiled. Because the range of a byte is from -128 to 127 and so 128 is not assignable to 'x'.

The interal value of 'b' is 98, which is less than 127 so Line //1 is fine.

Note: Although it is not required for the exam to know the integral values of characters, it is good to know that all English letters (upper case as well as lower case) as well as 0-9 are below 127 and so are assignable to byte.

Back to Question without Answer

17. QID - 2.860 : Using Operators and Decision Constructs

What will the following code print when run?

```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

Correct Option is : B

~~**A.** bat~~
big bat

B. big bat
none

Since there is a case condition that matches the input string "B", that case statement will be executed directly. This prints "big bat". Since there is no break after this case statement and the next case statement, the control will fall through the next one (which is default :) and so "none" will be printed as well.

Note that "b" and "B" are different strings. "B" is not equal to "b".

~~C.~~ big bat

~~D.~~ bat

~~E.~~ The code will not compile.

Explanation:

As of JDK 7 release, you can use a String object in the expression of a switch statement:

```
public String getTypeOfDayWithSwitchStatement(String dayOfWeekArg) {
     String typeOfDay;
     switch (dayOfWeekArg) {
         case "Monday":
             typeOfDay = "Start of work week";
             break;
         case "Tuesday":
         case "Wednesday":
         case "Thursday":
             typeOfDay = "Midweek";
             break;
         case "Friday":
             typeOfDay = "End of work week";
             break;
         case "Saturday":
         case "Sunday":
             typeOfDay = "Weekend";
             break;
         default:
             throw new IllegalArgumentException("Invalid day of the
     }
     return typeOfDay;
}
```

The switch statement compares the String object in its expression with the expressions

associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

Back to Question without Answer

18. QID - 2.1180 : Using Operators and Decision Constructs

The following code snippet will not compile:

```
int i = 10;
System.out.println( i<20 ? out1() : out2() );
```

Assume that out1 and out2 have method signature: public void out1(); and public void out2();

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

Note that it is not permitted for either the second or the third operand expression of the ? operator to be an invocation of a void method.

If one of the operands is of type byte and the other is of type short, then the type of the conditional expression is short.
If one of the operands is of type T where T is byte, short, or char, and the other operand is a constant expression of type int whose value is representable in type T, then the type of the conditional expression is T.

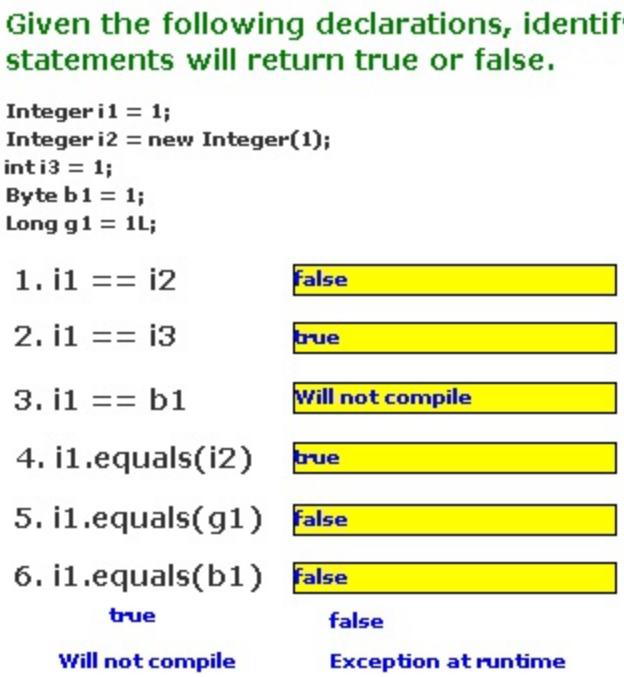
Otherwise, binary numeric promotion (5.6.2) is applied to the operand types, and the type of the conditional expression is the promoted type of the second and third operands.
If one of the second and third operands is of the null type and the type of the other is a reference type, then the type of the conditional expression is that reference type.

If the second and third operands are of different reference types, then it must be possible to convert one of the types to the other type (call this latter type T) by assignment conversion (5.2); the type of the conditional expression is T. It is a compile-time error if neither type is assignment compatible with the other type.

Back to Question without Answer

19. QID - 2.1125 : Using Operators and Decision Constructs

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.



Explanation:

`i1 == i2` will return false because both are pointing to different object.
`i1 == i3` will return true because one operand is a primitive int and so the other will be unboxed and then the value will be compared.
`i1 == b1` will not even compile because type of i1 and b1 references are classes that are not in the same class hierarchy. So == knows at compile time itself that they can't point to the same object.
`i1.equals(i2)` will return `true` because both are Integer objects and both have the value 1.
`i1.equals(b1)` and `i1.equals(g1)` will return false because they are pointing to objects of different types.

Signature of equals method is : `boolean equals(Object o);` So it can take any

object hence there will be no compilation error.
Further, The equals methods of all wrapper classes first check if the two object are of same class or not. If not, they immediately return `false`.

Back to Question without Answer

20. QID - 2.1089 : Using Operators and Decision Constructs

Drag and drop valid operators (shown in blue) in yellow boxes.
= operator can be used more than once.

`=` `+=` `*=`

```
public class TestClass
{
   public static void main(String[] args)
   {
     Short k = 9;     Integer i = 9;  Boolean b = false;
     char c = 'a';    String str = "123";

     i    [ = ]    (int) k.shortValue();
     str  [ += ]   b;
     b    [ = ]    !b;
     c    [ *= ]   i;

   }
}
```

Explanation:

1. `i = (int) k.shortValue();` --> You can use *= here but then you can't complete the 4th line.

2. `str += b;` --> You can't use =, or *= here. Only += is valid.

3. `b = !b;` --> You can't use anything other than = here.

4. `c *= i;` --> You can only use *= or +=. = is not valid. Further, if you use += here, you can't complete line 2.

Back to Question without Answer

21. QID - 2.994 : Using Operators and Decision Constructs

Assume that a, b, and c refer to instances of primitive wrapper classes. Which of the following statements are correct?

Note: Although primitive wrapper classes are not explicitly mentioned in the exam objectives, we have seen some candidates get questions on this aspect of wrapper classes.

Correct Options are : A E

A. a.equals(a) will always return true.

~~**B.**~~ b.equals(c) may return false even if c.equals(b) returns true.

~~**C.**~~ a.equals(b) returns same as a == b.

> The wrapper classes's equals() method overrides Object's equals() method to compare the actual value instead of the reference.

~~**D.**~~ a.equals(b) throws an exception if they refer to instances of different classes.

> It returns false in such a case.

E. a.equals(b) returns false if they refer to instances of different classes.

Explanation:

Equals method of a primitive wrapper class (e.g. java.lang.Integer, Double, Float etc) are
1. symmetric => a.equals(b) returns same as b.equals(a)

2. transitive => if a.equals(b) and b.equals(c) return true, then a.equals(c) returns true.
3. reflexive => a.equals(a) return true.

For example, the following code for the equals method on Integer explains how it works:

```
public boolean equals(Object obj) {
   if (obj instanceof Integer) {
       return value == ((Integer)obj).intValue();
   }
   return false;
}
```

Back to Question without Answer

22. QID - 2.1237 : Using Operators and Decision Constructs

Given:

```
enum Season { SUMMER, WINTER, SPRING, FALL }
```

What will the following code print?

```
Season s = Season.SPRING;
    switch(s){
        case SUMMER : System.out.println("SUMMER");
        case default : System.out.println("SEASON");
        case WINTER :  System.out.println("WINTER");
    }
```

Correct Option is : C

~~**A.**~~ `SEASON`

~~**B.**~~ `SEASON`
`WINTER`

C. It will not compile.

`case default : System.out.println("SEASON");` is syntactically wrong.
It should just be:
`default : System.out.println("SEASON");`

~~**D.**~~ It will not print anything.

Explanation:

If the code is changed to default : `System.out.println("SEASON");` it will print `SEASON`

```
WINTER.
```

Remember that since there is no `break` after the case statements, the control will go through the rest of the case statements without even checking the value of the cases.

Back to Question without Answer

23. QID - 2.927 : Using Operators and Decision Constructs

What will the following code print ?

```
class Test{
   public static void main(String[] args){
      int k = 1;
      int[] a = { 1 };
      k += (k = 4) * (k + 2);
      a[0] += (a[0] = 4) * (a[0] + 2);
      System.out.println( k + " , " + a[0]);
   }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ 4 , 4

C. 25 , 25

~~**D.**~~ 13 , 13

~~**E.**~~ None of the above.

Explanation:

The value `1` of `k` is saved by the compound assignment operator `+=` before its right-hand operand `(k = 4) * (k + 2)` is evaluated. Evaluation of this right-hand operand then assigns `4` to `k`, calculates the value `6` for `k + 2`, and then multiplies `4` by

`6` to get `24`. This is added to the saved value 1 to get 25, which is then stored into `k` by the `+=` operator. An identical analysis applies to the case that uses `a[0]`.

```
k += (k = 4) * (k + 2);
a[0] += (a[0] = 4) * (a[0] + 2);
k = k + (k = 4) * (k + 2);
a[0] = a[0] + (a[0] = 4) * (a[0] + 2);
```

Back to Question without Answer

24. QID - 2.1087 : Using Operators and Decision Constructs

Which of the given lines can be inserted at //1 of the following program ?

```
public class TestClass{
   public static void main(String[] args){
     short s = 9;
     //1
   }
}
```

Correct Options are : D G

~~**A.**~~ `Short k = new Short(9); System.out.println(k instanceof Short);`

> 9 is considered an int. This should be: `Short s = new Short( (short) 9 );`

~~**B.**~~ `System.out.println(s instanceof Short);`

> The left operand of instanceof MUST be an object and not a primitive.

~~**C.**~~ `Short k = 9; System.out.println( k instanceof s);`

> Right operand of instanceof MUST be a class name.

D. `int i = 9; System.out.println(s == i);`

> Any two integral primitives can be compared using == operator.

~~**E.**~~ `Boolean b = s instanceof Number;`

> Left operand of instanceof MUST be an object and not a primitive.

F. ~~`Short k = 9; Integer i = 9; System.out.println(k == i);`~~

This will not compile because k and i are referring to objects that have no IS-A relationship among themselves.

G. `Integer i = 9; System.out.println( s == i );`

Back to Question without Answer

25. QID - 2.1082 : Using Operators and Decision Constructs

What will be the output when the following class is compiled and run?

```
class ScopeTest{
   static int x = 5;
   public static void main(String[] args){
      int x  = ( x=3 ) * 4;  // 1
      System.out.println(x);
   }
}
```

Correct Option is : D

~~**A.**~~ It will not compile because line //1 cannot be parsed correctly.

~~**B.**~~ It will not compile because x is used before initialization.

> It is not.

~~**C.**~~ It will not compile because there is an ambiguous reference to x.

> There is no conflict for resolution of x. The local 'x' simply shadows the member variable 'x'.

D. It will print 12.

~~**E.**~~ It will print 3 .

Explanation:

x is first initialized by x = 3, then the value of this expression (i.e. "x = 3"), which is 3,

is multiplied by 4 and is again assigned to x. So it prints 12.

Back to Question without Answer

26. QID - 2.859 : Using Operators and Decision Constructs

What will the following code print when run?

```
public class TestClass {

    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "c" : System.out.println( "cat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("c");
    }
}
```

Correct Option is : C

~~**A.**~~ apple
cat
none

~~**B.**~~ apple
cat

C. cat
none

Since there is a case condition that matches the input string "c", that case statement will be executed directly. This prints "cat". Since there is no break after this case statement and the next case statement, the control will fall through the next one (which is default :) and so "none" will be printed as well.

~~**D.** cat~~

Explanation:

In the JDK 7 release, you can use a String object in the expression of a switch statement:

```
public String getTypeOfDayWithSwitchStatement(String dayOfWeekArg) {
     String typeOfDay;
     switch (dayOfWeekArg) {
         case "Monday":
             typeOfDay = "Start of work week";
             break;
         case "Tuesday":
         case "Wednesday":
         case "Thursday":
             typeOfDay = "Midweek";
             break;
         case "Friday":
             typeOfDay = "End of work week";
             break;
         case "Saturday":
         case "Sunday":
             typeOfDay = "Weekend";
             break;
         default:
             throw new IllegalArgumentException("Invalid day of the
week: " + dayOfWeekArg);
     }
     return typeOfDay;
}
```

The switch statement compares the String object in its expression with the expressions

associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

Back to Question without Answer

27. QID - 2.1164 : Using Operators and Decision Constructs

Which of the following statements will compile without any error?

Correct Options are : A B C D

A. `System.out.println("a"+'b'+63);`

> Since the first operand is a String all others (one by one) will be converted to String."ab" + 63 => "ab63"

B. `System.out.println("a"+63);`

> Since the first operand is a String all others (one by one) will be converted to String."a" + 'b' => "a63"

C. `System.out.println('b'+new Integer(63));`

> Since the first operand of + one is of numeric type, its numeric value of 98 will be used. Integer 63 will be unboxed and added to 98. Therefore, the final value will be int 161.

D. `String s = 'b'+63+"a";`

> Since the first one is numeric type so, 'b'+63 = 161, 161+"a" = 161a.

E. `String s = 63 + new Integer(10);`

> Since none of '+' the operands is a String, the + operator will not generate a String. However, due to auto-unboxing, it will generate an int value of 73.

Explanation:

+ is overloaded such that if any one of its two operands is a String then it will convert the other operand to a String and create a new string by concatenating the two.
Therefore, in 63+"a" and "a"+63, 63 is converted to "63" and 'b' +"a" and "a"+'b', 'b' is converted to "b".
Note that in 'b'+ 63 , 'b' is promoted to an int i.e. 98 giving 161.

Back to Question without Answer

28. QID - 2.1061 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following class?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i, j, k;
      i = j = k = 9;
      System.out.println(i);
   }
}
```

Correct Options are : C E

~~**A.**~~ The code will not compile because unlike in c++, operator '=' cannot be chained i.e. a = b = c = d is invalid.

> It can be chained. I.e. assuming all the variables are declared appropriately, `a = b = c = d;` is valid.

~~**B.**~~ The code will not compile as 'j' is being used before getting initialized.

> j is being initialize by the expression k = 9, which evaluates to 9.

C. The code will compile correctly and will display '9' when run.

~~**D.**~~ The code will not compile as 'j' and 'i' are being used before getting initialized.

E. All the variables will get a value of 9.

Explanation:

Every expression has a value, in this case the value of the expression is the value that is assigned to the Right Hand Side of the equation.
k has a value of 9 which is assigned to j and then to i.

Another implication of this is :

```
boolean b = false;
if( b = true) { System.out.println("TRUE");}
```

The above code is valid and will print TRUE. Because `b = true` has a boolean value, which is what an if statement expects.

Note that `if( i = 5) { ... }` is not valid because the value of the expression `i = 5` is an int (5) and not a boolean.

Back to Question without Answer

29. QID - 2.1039 : Using Operators and Decision Constructs

What will the following method return if called with an argument of 7?

```
public int transformNumber(int n){
   int radix = 2;
   int output = 0;
   output += radix*n;
   radix = output/radix;
   if(output<14){
       return output;
   }
   else{
       output = output*radix/2;
       return output;
   }
   else {
       return output/2;
   }
}
```

Correct Option is : D

~~**A.**~~ 7

~~**B.**~~ 14

~~**C.**~~ 49

D. Compilation fails.

The if-else-else is invalid. It should be if , else if, else.

Back to Question without Answer

30. QID - 2.1335 : Using Operators and Decision Constructs

Consider the code shown below:

```
public class TestClass{
  public static int switchTest(int k){
     int j = 1;
     switch(k){
        case 1: j++;
        case 2: j++;
        case 3: j++;
        case 4: j++;
        case 5: j++;
        default : j++;
     }
     return j + k;
  }
  public static void main(String[] args){
     System.out.println( switchTest(4) );
  }
}
```

What will it print when compiled and run?

Correct Option is : D

~~**A.**~~ 5

~~**B.**~~ 6

~~**C.**~~ 7

D. 8

~~E.~~ 9

Explanation:

The control in the case falls through till reaches the break statement.
Here, switch(4) will take the control to case 4:.
Now since there is no break statement, all the statements till the end will be executed.
So j will be incremented 3 time making it 4. finally 4 + 4 i.e. 8 will be returned.

Back to Question without Answer

31. QID - 2.1156 : Using Operators and Decision Constructs

What will the following class print when executed?

```
class Test{
    static boolean a;
    static boolean b;
    static boolean c;
    public static void main (String[] args){
        boolean bool = (a = true) || (b = true) && (c = true);
        System.out.print(a + ", " + b + ", " + c);
    }
}
```

Correct Option is : C

~~**A.**~~ true, false, true

~~**B.**~~ true, true, false

C. true, false, false

~~**D.**~~ true, true, true

Explanation:

Java parses the expression from left to right. Once it realizes that the left operand of a conditional "or" operator has evaluated to true, it does not even try to evaluate the right side expression.

Back to Question without Answer

32. QID - 2.965 : Using Operators and Decision Constructs

Consider the following program:

```
public class TestClass{
   public static void main(String[] args)  {     calculate(2);     }
   public static void calculate(int x){
      String val;
      switch(x){
         case 2:
         default:
         val = "def";
      }
      System.out.println(val);
   }
}
```

What will happen if you try to compile and run the program?

Correct Options are : B D

~~**A.**~~ It will not compile saying that variable `val` may not have been initialized..

B. It will compile and print `def`

~~**C.**~~ As such it will not compile but it will compile if `calculate(2);` is replaced by `calculate(3);`

D. It will compile for any int values in `calculate(...);`

Explanation:

When you try to access a local variable, the compiler makes sure that it is initialized in all the cases. If it finds that there is a case in which it may not be initialized then it flags an error. For example:

```
int i;
if( somecondition) i = 20;
int k = i;
```

Here, if some condition returns `false`, then i remains uninitialized hence the compiler flags an error.
In the given question:
As there is no `break` after `case 2`, `val` will always be initialized in the `switch` block. So it will compile and run fine.
Note that it will not compile if `break` is placed after `case 2` because the compiler will figure out that in certain cases `val` may be left uninitialized.

Back to Question without Answer

33. QID - 2.1105 : Using Operators and Decision Constructs

Consider the following method...

```
public void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   System.out.println("True False");
   else         // 3
   System.out.println("True True");
   else         // 4
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Correct Options are : A C D

A. If run with an argument of 'false', it will print 'False False'

~~**B.**~~ If run with an argument of 'false', it will print 'True True'

C. If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

~~**E.**~~ It will not compile.

Explanation:

Note that if and else do not cascade. They are like opening and closing braces.

```
if (flag)   //1
     if (flag)   //2
        System.out.println("True False");
     else        // 3 This closes //2
        System.out.println("True True");
 else        // 4 This closes //1
    System.out.println("False False");
```

So, else at //3 is associated with if at //2 and else at //4 is associated with if at //1

Back to Question without Answer

34. QID - 2.1238 : Using Operators and Decision Constructs

What will the following code print when run without any arguments ...

```
public class TestClass {

    public static int m1(int i){
        return ++i;
    }

    public static void main(String[] args) {

        int k = m1(args.length);
        k += 3 + ++k;
        System.out.println(k);
    }

}
```

Correct Option is : C

~~**A.**~~ It will throw `ArrayIndexOutofBoundsException`.

~~**B.**~~ It will throw `NullPointerException`.

C. 6

~~**D.**~~ 5

~~**E.**~~ 7

~~**F.**~~ 2

~~**G.**~~ None of these.

Explanation:

When the program is run without any arguments, `args` gets assigned a string array of size 0. So `NullPointerException` or `ArrayIndexOutofBoundsException` are out of question. Thus, the first call becomes :

```
int k = m1(0);
```

Follow through the code like this:
1. Method m1() uses pre-increment operation. Therefore, first i is incremented and then the new value of i is returned.
2. Thus, k gets the value of 1.

3. Expand the += operator as:

```
k = k + 3 + ++k;
```

This becomes (remember that k = 1 at this point):

`k = 1 + 3 + (++k)` i.e.

`k = 1 + 3 + 2;` (at this point value of k is 2 because of ++k). But the value of Right Hand Side has not yet been assigned to k.

`k = 6;` 6 is assigned to k thereby overwriting the value of 2.

Therefore, the final value of k is 6.

Back to Question without Answer

35. QID - 2.1266 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Options are : A B E

A. The condition expression in an if statement can contain method calls.

> Yes, as long as the method returns a boolean value.

B. If a and b are of type boolean, the expression (a = b) can be used as the condition expression of an if statement.

~~**C.**~~ An if statement can have either an 'if' clause or an 'else' clause.

> An if-statement must always have an 'if' clause. 'else' is optional.

~~**D.**~~ The statement : if (false) ; else ; is illegal.

> if-clause and the else-clause can have empty statements. Empty statement (i.e. just a semi-colon) is a valid statement.

E. Only expressions which evaluate to a boolean value can be used as the condition in an if statement.

> Unlike C/C++ where you can use integers as conditions, in java, only booleans are allowed.

Explanation:

The expression `(a = b)` does not compare the variables a and b, but rather assigns the value of b to the variable a. The result of the expression is the value being assigned.

Since a and b are `boolean` variables, the value returned by the expression is also `boolean`. This allows the expressions to be used as the condition for an if-statement. if-clause and the else-clause can have empty statements. Empty statement (i.e. just ;) is a valid statement.
But this is illegal :

```
if (true) else;
```

because the if part doesn't contain any valid statement. (A statement cannot start with an else!)
So, the following is valid.

```
if(true) if(false);
```

because `if(false);` is a valid statement.

Back to Question without Answer

36. QID - 2.1282 : Using Operators and Decision Constructs

What, if anything, is wrong with the following code?

```
void test(int x){
   switch(x){
      case 1:
      case 2:
      case 0:
      default :
      case 4:
   }
}
```

Correct Option is : E

~~**A.**~~ Data Type of 'x' is not valid to be used as an expression for the switch clause.

> x is an int and int is perfectly valid. long, double, boolean, and float are not valid.

~~**B.**~~ The case label 0 must precede case label 1.

> While ordering may be important for the logic being implemented in the code, technically, any order is valid.

~~**C.**~~ Each case section must end with a break keyword.

> This is not necessary. If there is no break at the end of a case section, the control will fall through to the next case section (even if the case label doesn't match).

~~**D.**~~ The default label must be the last label in the switch statement.

> Any order of case statements is valid.

E. There is nothing wrong with the code.

Back to Question without Answer

37. QID - 2.1269 : Using Operators and Decision Constructs

Which of the following will not give any error at compile time and run time?

Correct Options are : A C D E

A. `if (8 == 81) {}`

> `8 == 81` is a valid expression that returns false.

B. ~~`if (x = 3) {}`~~ // assume that x is an int

> Because the exp. x = 3 does not return a boolean.

C. `if (true) {}`

D. `if (bool = false) {}` //assume that bool is declared as a boolean

> Because the expression 'bool = false' returns a boolean (which happens to be false)

E. `if (x == 10 ? true:false) { }` // assume that x is an int

Explanation:

All an `if(...)` needs is a `boolean`.
`x = 3` is not valid because the return value of this expression is 3 which is not a `boolean`.

Back to Question without Answer

38. QID - 2.1212 : Using Operators and Decision Constructs

Which of the following expressions will evaluate to true if preceded by the following code?

```
String a = "java";
    char[] b = { 'j', 'a', 'v', 'a' };
    String c = new String(b);
    String d = a;
```

Correct Options are : A C D

A. `(a == d)`

~~B.~~ `(b == d)`

b and d can not even be compared because they are of different types.

C. `(a == "java")`

D. `a.equals(c)`

Note that a == c will be false because doing 'new' creates an entirely new object.

Back to Question without Answer

39. QID - 2.1119 : Using Operators and Decision Constructs

Given:

```
public class Switcher{

   public static void main(String[] args){
       switch(Integer.parseInt(args[1]))  //1
       {
          case 0 :
             boolean b = false;
             break;

          case 1 :
             b = true; //2
             break;
       }

       if(b) System.out.println(args[2]);
   }
}
```

What will the above program print if compiled and run using the following command line:

```
java Switcher 1 2 3
```

Correct Option is : G

~~**A.**~~ It will print 1

~~**B.**~~ It will print 2

~~**C.**~~ It will print 3

~~**D.**~~ It will not print anything.

~~**E.**~~ It will not compile because of //1.

> There is no problem here because Integer.parseInt() returns an int.

~~**F.**~~ It will not compile because of //2.

> There is no problem here. b is in scope for the rest of the switch block.

G. It will not compile for some other reason.

> It will not compile because of `if(b)` because `b` is declared in the switch block and it is out of scope after the switch block ends. Pay close attention to question text. It may seem to test you on one concept but actually it could be testing something entirely different.

Back to Question without Answer

40. QID - 2.932 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Options are : B C

~~**A.**~~ The modulus operator % can only be used with integer operands.

> It can be used on floating points operands also. For example, 5.5 % 3 = 2.5

B. & can have integral as well as boolean operands.

> unlike &&, & will not "short circuit" the expression if used on boolean parameters.

C. The arithmetic operators *, / and % have the same level of precedence.

~~**D.**~~ && can have integer as well as boolean operands.

> !, && and || operate only on booleans.

~~**E.**~~ ~ can have integer as well as boolean operands.

> ~ Operates only on integral types

Explanation:

Note :
integral types means byte, short, int, long, and char

Reference: http://java.sun.com/docs/books/jls/third_edition/html/typesValues.html

"The types of the Java programming language are divided into two categories: primitive types and reference types. The primitive types (§4.2) are the boolean type and the numeric types. The numeric types are the integral types byte, short, int, long, and char, and the floating-point types float and double. The reference types (§4.3) are class types, interface types, and array types. There is also a special null type. An object (§4.3.1) is a dynamically created instance of a class type or a dynamically created array. The values of a reference type are references to objects. All objects, including arrays, support the methods of class Object (§4.3.2). String literals are represented by String objects (§4.3.3)."

Back to Question without Answer

41. QID - 2.1342 : Using Operators and Decision Constructs

Consider that `str` is a variable of class `java.lang.String`.
Which of the following lines of code may throw a `NullPointerException` in certain situations?

Or a tougher version of the question could be :
Which of the following lines of code are not an example of robust design ?

Correct Options are : A B C

A. `if ( (str != null) | ( i == str.length() ) )`

> `(i == str.length())` will always be executed so if 'str' is null, then str.length() will throw a NullPointerException.

B. `if ( (str == null) | ( i == str.length() ) )`

> `(i == str.length())` will always be executed so if 'str' is null, then str.length() will throw a NullPointerException.

C. `if ( (str != null) || (i == str.length() ) )`

> `(i == str.length())` will only be evaluated if (str != null) is false, and (str != null) will be false if 'str' is null. So it will also throw a NullPointerException.

~~D.~~ ~~`if ( (str == null) || (i == str.length() ) )`~~

> `(i == str.length())` will only be evaluated if (str == null) is false, and (str == null) will be false if 'str' is NOT null. So it will NEVER throw a NullPointerException.

Explanation:

The concept is : `||` and `&&` are short circuiting operation i.e. if the value of the expression can be known by just seeing the first part then the remaining part is not evaluated while `|` and `&` will always let all the parts evaluates.
Let's break this down in two cases:

1. Say `str = null;`

for a, the first part is false and `str.length()` throws `NullPointerException` because str is null.

for b, the first part of it is true but it will still evaluate the second part and as str is null, str.length() throws NullPointerException. Had it been || instead of |, the second part would not have been evaluated and no exception would have been thrown.

for c, the first part of it is false and it will also evaluate the second part which will throw a NullPointerException as str is null.

for d, the first part is true, so the second part is not evaluated.

2. Say, `str = "somestring";` //i.e. str is not null.

for a, the first part is true, so is the second part. No exception is thrown. Note that second part will still be evaluated although by looking at the first part itself we can tell that the whole expression will return true.

for b, the first part is false, and the second part is also true. No exception is thrown.

for c, first part is true, so second part is not evaluated at all. No exception is thrown.

for d, first part is false, so it will evaluate second part. No exception is thrown as str

is not null.

It would be nice if you try to run the following program to understand the concept : (Uncomment only one of the commented lines one by one).

```
public class TestClass {
   public static void main(String[] args) {
      int i = 0;
      String s = "";

      //s = null;
      if ((s != null) | ( i==s.length())) {}
      System.out.println("A");

      //s = null;
      if ((s == null) | ( i==s.length())) {}
      System.out.println("B");

      //s = null;
      if ((s != null) || (i==s.length())) {}
      System.out.println("C");

      s = null;
      if ((s == null) || (i==s.length())) {}
      System.out.println("D");
   }
}
```

Back to Question without Answer

42. QID - 2.1278 : Using Operators and Decision Constructs

Which of the following four constructs are valid?

1.
```
   switch(5)
  {
      default :
   }
```

2.
```
   switch(5)
   {
      default : break;
   }
```

3.
```
  switch(8);
```

4.
```
 int x = 0;
 switch(x){
 }
```

Correct Option is : D

~~**A.**~~ 1, 3

~~**B.**~~ 1, 2, 3

~~**C.**~~ 3, 4

D. 1, 2, 4

Code 3 is invalid because a switch statement must have a body. The body may even be empty as shown in Code 4.

E. All are valid.

Back to Question without Answer

43. QID - 2.1038 : Using Operators and Decision Constructs

What will be the output of the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true ;
      boolean bool2 = false;
      boolean bool  = false;
      bool = ( bool2 &  method1(i++) ); //1
      bool = ( bool2 && method1(i++) ); //2
      bool = ( bool1 |  method1(i++) ); //3
      bool = ( bool1 || method1(i++) ); //4
      System.out.println(i);
   }
   public static boolean method1(int i){
       return i>0 ? true : false;
   }
}
```

Correct Option is : B

~~**A.**~~ It will print 1.

B. It will print 2.

~~**C.**~~ It will print 3.

~~**D.**~~ It will print 4.

~~**E.**~~ It will print 0.

Explanation:

& and | do not short circuit the expression but && and || do.
As the value of all the expressions (1 through 4) can be determined just by looking at the first part, && and || do not evaluate the rest of the expression, so method1() is not called for 2 and 4.
Hence the value of i is incremented only twice.

Back to Question without Answer

44. QID - 2.948 : Using Operators and Decision Constructs

Which statements about the output of the following programs are true?

```
public class TestClass{
   public static void main(String args[ ] ){
      int i = 0 ;
      boolean bool1 = true;
      boolean bool2 = false;
      boolean bool  = false;
      bool = (bool2 &  method1("1"));  //1
      bool = (bool2 && method1("2"));  //2
      bool = (bool1 |  method1("3"));  //3
      bool = (bool1 || method1("4"));  //4
   }
   public static boolean method1(String str){
      System.out.println(str);
      return true;
   }
}
```

Correct Options are : A C

A. 1 will be the part of the output.

> & (unlike &&), when used as a logical operator, does not short circuit the expression, which means it always evaluates both the operands even if the result of the whole expression can be known by just evaluating the left operand.

B. 2 will be the part of the output.

C. 3 will be the part of the output.

> & and | (unlike && and ||), when used as logicals operators, do not short circuit

the expression, which means they always evaluate both the operands even if the result of the whole expression can be known by just evaluating the left operand.

~~**D.**~~ 4 will be the part of the output.

~~**E.**~~ None of the above

Explanation:

`&` and `|` do not short circuit the expression. The value of all the expressions (1 through 4) can be determined just by looking at the first part.
`&&` and `||` do not evaluate the rest of the expression if the result of the whole expression can be known by just evaluating the left operand, so `method1()` is not called for 2 and 4.

Back to Question without Answer

45. QID - 2.977 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2  = false;
if (b2 = b1 != b2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Correct Option is : C

~~**A.**~~ Compile time error.

~~**B.**~~ It will print true;

C. It will print false;

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

All an `if()` needs is a `boolean`. Now, `b1 != b2` returns `false` which is a boolean and so the expression becomes `b2 = false`. It returns `false` which is again a `boolean`. So there is no error and it prints `false`.
Remember that every expression has a return value (which is actually the Left Hand Side of the expression). For example, The value of the expressing `i = 10` , is `10` (an

int).

Back to Question without Answer

46. QID - 2.1065 : Using Operators and Decision Constructs

What will the following code print?

```
boolean flag = true;
if(flag = false){
   System.out.println("1");
}else if(flag){
   System.out.println("2");
}else if(!flag){
   System.out.println("3");
}else    System.out.println("4");
```

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 2

C. 3

~~**D.**~~ 4

~~**E.**~~ Compilation error.

Explanation:

At the beginning, flag is true. In the first if, we do flag = false. Notice that it is not flag == false. It is a single =, which assigns false to flag. Thus, flag becomes false and the condition becomes false therefore 1 is not printed. In the first 'else if', again since flag is false, 2 is not printed. In second 'else if', !flag implies !false, which is true, so 3 is

printed. Finally, since an else-if condition has been satisfied, the last else is not executed.

Back to Question without Answer

47. QID - 2.1084 : Using Operators and Decision Constructs

Which of the following are valid operators in Java?

Correct Options are : A B C D

A. !

operates only on booleans

B. ~

bitwise negation. Operates only on integral types.

C. &

bitwise AND

D. %=

similar to += or /=

~~E.~~ $

It is not an operator!

Back to Question without Answer

48. QID - 2.1239 : Using Operators and Decision Constructs

What letters will be printed by this program?

```
public class ForSwitch{
    public static void main(String args[]){
        char i;
        LOOP: for (i=0;i<5;i++){
            switch(i++){
                case '0': System.out.println("A");
                case 1: System.out.println("B"); break LOOP;
                case 2: System.out.println("C"); break;
                case 3: System.out.println("D"); break;
                case 4: System.out.println("E");
                case 'E' : System.out.println("F");
            }
        }
    }
}
```

Correct Options are : C E

~~**A.**~~ A

~~**B.**~~ B

C. C

~~**D.**~~ D

E. F

Explanation:

1. Defining i as char doesn't mean that it can only hold characters (a, b, c etc). It is an integral data type which can take any +ive integer value from 0 to 2^16 -1.
2. Integer 0 or 1, 2 etc. is not same as char '0', '1' or '2' etc.
so when i is equal to 0, nothing gets printed and i is incremented to 1 (due to i++ in the switch).
i is then incremented again by the for loop for next iteration. so i becomes 2.
when i = 2, "C" is printed and i is incremented to 3 (due to i++ in the switch) and then i is incremented to 4 by the for loop so i becomes 4.
when i = 4, "E" is printed and since there is no break, it falls through to case '5' and "F" is printed.
i is incremented to 5 and the for loop ends.

Back to Question without Answer

49. QID - 2.1313 : Using Operators and Decision Constructs

What is the result of executing the following code when the value of i is 5:

```
switch (i){
    default:
    case 1:
        System.out.println(1);
    case 0:
        System.out.println(0);
    case 2:
        System.out.println(2);
        break;
    case 3:
        System.out.println(3);
}
```

Correct Option is : A

A. It will print `1 0 2`

~~**B.**~~ It will print `1 0 2 3`

~~**C.**~~ It will print `1 0`

~~**D.**~~ It will print `1`

~~**E.**~~ Nothing will be printed.

Explanation:

Here are the rules:

The type of the Expression must be char, byte, short, or int or a compile-time error occurs. Java 7 allows String as well.
The class of the switch variable may also be a Wrapper class i.e. Character, Byte, Short, or Integer.

All of the following must be true, or a compile-time error will result:
1.Every case constant expression associated with a switch statement must be assignable (5.2) to the type of the switch Expression.
2.No two of the case constant expressions associated with a switch statement may have the same value.
3.At most one default label may be associated with the same switch statement.
Basically it looks for a matching case or if no match is found it goes to default. (If default is also not found it does nothing)
Then it executes the statements till it reaches a break or end of the switch statement.
Here it goes to default and executes till it reaches first break. So it prints 1 0 2.

Note that the switch statement compares the String object in its expression with the expressions associated with each case label as if it were using the String.equals method; consequently, the comparison of String objects in switch statements is case sensitive. The Java compiler generates generally more efficient bytecode from switch statements that use String objects than from chained if-then-else statements.

Back to Question without Answer

50. QID - 2.1040 : Using Operators and Decision Constructs

What will the following class print ?

```
class InitTest{
   public static void main(String[] args){
      int a = 10;
      int b = 20;
      a += (a = 4);
      b = b + (b = 5);
      System.out.println(a+ ",  "+b);
   }
}
```

Correct Option is : E

~~**A.**~~ It will print 8, 25

~~**B.**~~ It will print 4, 5

~~**C.**~~ It will print 14, 5

~~**D.**~~ It will print 4, 25

E. It will print 14, 25

Explanation:

`a += (a =4)` is same as `a = a + (a=4).`
First, a's value of 10 is kept aside and (a=4) is evaluated. The statement (a=4) assigns 4 to a and the whole statement returns the value 4. Thus, 10 and 4 are added and

assigned back to a.

Same logic applies to `b = b + (b = 5);` as well.

Back to Question without Answer

51. QID - 2.958 : Using Operators and Decision Constructs

Consider the following code:

```
public class Conversion{
   public static void main(String[] args){
     int i = 1234567890;
     float f = i;
     System.out.println(i - (int)f);
   }
}
```

What will it print when run?

Correct Option is : B

~~**A.**~~ It will print 0.

B. It will not print 0.

~~**C.**~~ It will not compile.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

Actually it prints `-46`. This is because the information was lost during the conversion from type int to type float as values of type float are not precise to nine significant digits.

Note: You are not required to know the number of significant digits that can be stored by a float for the exam. However, it is good to know about loss of precision while using float and double.

Back to Question without Answer

52. QID - 2.1173 : Using Operators and Decision Constructs

Consider the following class :

```
public class Test{
   public static void main(String[] args){
      if (args[0].equals("open"))
         if (args[1].equals("someone"))
            System.out.println("Hello!");
      else System.out.println("Go away "+ args[1]);
    }
}
```

Which of the following statements are true if the above program is run with the command line :
java Test closed

Correct Option is : B

~~**A.**~~ It will throw `ArrayIndexOutOfBoundsException` at runtime.

B. It will end without exceptions and will print nothing.

~~**C.**~~ It will print `Go away`

~~**D.**~~ It will print `Go away` and then will throw `ArrayIndexOutOfBoundsException`.

~~**E.**~~ None of the above.

Explanation:

As in C and C++, the Java if statement suffers from the so-called "dangling else problem," The problem is that both the outer if statement and the inner if statement might conceivably own the else clause.
In this example, one might be tempted to assume that the programmer intended the else clause to belong to the outer if statement.

The Java language, like C and C++ and many languages before them, arbitrarily decree that an else clause belongs to the innermost if so as the first if() condition fails (args[0] not being "open") there is no else associated to execute. So, the program does nothing. The else actually is associated with the second if. So had the command line been :
`java Test open`, it would have executed the second if and thrown `ArrayIndexOutOfBoundsException`.
If the command line had been:
`java Test open xyz`, it would execute the else part(which is associated with the second if) and would have printed `"Go away xyz"`.

Back to Question without Answer

53. QID - 2.1267 : Using Operators and Decision Constructs

Which of the following implementations of a `max()` method will correctly return the largest value?

Correct Option is : D

~~**A.**~~
```
int max(int x, int y){
    return(  if(x > y){ x; } else{ y; }  );
}
```

The if statement does not return any value so it can not be used the way it is used in (1).

~~**B.**~~
```
int max(int x, int y){
    return( if(x > y){ return x; }  else{ return y; } );
}
```

It would work if the first return and the corresponding brackets is removed.

~~**C.**~~
```
int max(int x, int y){
    switch(x < y){
       case true:
              return y;
       default :
              return x;
    };
}
```

Neither the switch expression nor the case labels can be of type boolean.

D.
```
int max(int x, int y){
     if (x > y)  return x;
     return y;
}
```

E. None of the above.

Back to Question without Answer

54. QID - 2.949 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2 = false;
if (b2 != b1 = !b2){
   System.out.println("true");
}
else{
   System.out.println("false");
}
```

Correct Option is : A

A. Compile time error.

~~**B.**~~ It will print `true`.

~~**C.**~~ It will print `false`.

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

Note that boolean operators have more precedence than =. (In fact, = has least precedence of all operators.)
so, in (b2 != b1 = !b2) first b2 != b1 is evaluated which returns a value 'false'. So the expression becomes false = !b2. And this is illegal because false is a value and not a

variable!

Had it been something like (b2 = b1 != b2) then it is valid because it will boil down to : b2 = false.
Because all an if() needs is a boolean, now b1 != b2 returns false which is a boolean and as b2 = false is an expression and every expression has a return value (which is actually the Left Hand Side of the expression). Here, it returns false, which is again a boolean.

Note that return value of expression : i = 10 , where i is an int, is 10 (an int).

Back to Question without Answer

55. QID - 2.1157 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following code?

```
class SwitchTest{
   public static void main(String args[]){
      for ( int i = 0 ; i < 3 ; i++){
         boolean flag  = false;
         switch (i){
            flag  = true;
         }
         if ( flag )  System.out.println( i );
      }
   }
}
```

Correct Option is : C

~~**A.**~~ It will print 0, 1 and 2.

~~**B.**~~ It will not print anything.

C. Compilation error.

It will say 'case', 'default' or '}' expected at compile time.

~~**D.**~~ Runtime error.

~~**E.**~~ None of the above.

Explanation:

You cannot have unlabeled block of code inside a `switch` block. Any code block must succeed a case label (or default label). Since there is no `case` statement in this switch block, there is no way the line `flag = true;` can be reached! Therefore, it will not compile.

Back to Question without Answer

56. QID - 2.1240 : Using Operators and Decision Constructs

Given:

```
byte b = 1;
char c = 1;
short s = 1;
int i = 1;
```

which of the following expressions are valid?

Correct Options are : B C E

~~**A.**~~ `s = b * b ;`

b * b returns an int.

B. `i = b << b ;`

C. `s <<= b ;`

All compound assignment operators internally do an explicit cast.

~~**D.**~~ `c = c + b ;`

c + b returns an int

E. `s += i ;`

All compound assignment operators internally do an explicit cast.

Explanation:

Remember these rules for primitive types:
1. Anything bigger than an int can NEVER be assigned to an int or anything smaller than int (byte, char, or short) without explicit cast.
2. CONSTANT values up to int can be assigned (without cast) to variables of lesser size (for example, short to byte) if the value is representable by the variable.(that is, if it fits into the size of the variable).
3. operands of mathematical operators are ALWAYS promoted to AT LEAST int. (i.e. for byte * byte both bytes will be first promoted to int.) and the return value will be AT LEAST int.
4. Compound assignment operators (+=, *= etc) have strange ways so read this carefully:

A compound assignment expression of the form E1 op= E2 is equivalent to `E1 = (T) ((E1) op (E2))`, where T is the type of E1, except that E1 is evaluated only once. Note that the implied cast to type T may be either an identity conversion or a narrowing primitive conversion.
For example, the following code is correct:

```
short x = 3;
x += 4.6;
```

and results in x having the value 7 because it is equivalent to:

```
short x = 3;
x = (short)(x + 4.6);
```

Back to Question without Answer

57. QID - 2.1317 : Using Operators and Decision Constructs

Which of the following are NOT valid operators in Java?

Correct Options are : A B D E

A. sizeof

> It is in C++ but not in java because size of everything is known at compile time and is not machine dependent.

B. <<<

> For left shifts there is no difference between shifting signed and unsigned values so there is only one leftshift '<<' in java.

~~**C.**~~ instanceof

D. mod

> No such thing.

E. equals

> boolean equals(Object o) is a method in java.lang.Object. It is not an operator.

Back to Question without Answer

58. QID - 2.1071 : Using Operators and Decision Constructs

What will be the output of the following code snippet?

```
int a = 1;
int[] ia = new int[10];
int b = ia[a];
int c = b + a;
System.out.println(b = c);
```

Correct Option is : B

~~**A.**~~ 0

B. 1

~~**C.**~~ 2

~~**D.**~~ true

~~**E.**~~ false

Explanation:

1. All the elements of an array of primitives are automatically initialized by default values, which is 0 for numeric types and false for boolean.
Therefore, ia[1] is 0.
2. = is not same as ==. The statement b = c assigns c (whose value is 1) to b. which is then printed.

Back to Question without Answer

59. QID - 2.1070 : Using Operators and Decision Constructs

What is the result of executing the following fragment of code:

```
boolean b1 = false;
int i1 = 2;
int i2 = 3;
if (b1 = i1 == i2){
   System.out.println("true");
} else{
   System.out.println("false");
}
```

Correct Option is : C

~~**A.**~~ Compile time error.

~~**B.**~~ It will print true

C. It will print false

~~**D.**~~ Runtime error.

~~**E.**~~ It will print nothing.

Explanation:

All an if statement needs is a boolean. Now i1 == i2 returns false which is a boolean and since b1 = false is an expression and every expression has a return value (which is actually the Left Hand Side of the expression), it returns false which is again a boolean. Therefore, in this case, the else condition will be executed.

Back to Question without Answer

60. QID - 2.1271 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following code?

```
public class PromotionTest{
   public static void main(String args[]){
      int i = 5;
      float f = 5.5f;
      double d = 3.8;
      char c = 'a';
      if (i == f) c++;
      if (((int) (f + d)) == ((int) f + (int) d)) c += 2;
      System.out.println(c);
   }
}
```

Correct Option is : E

~~**A.**~~ The code will fail to compile.

~~**B.**~~ It will print d.

~~**C.**~~ It will print c.

~~**D.**~~ It will print b

E. It will print a.

Explanation:

In the case of i == f, value of i will be promoted to a float i.e. 5.0, and so it returns false.
(int)f+(int)d = (int)5.5 + (int) 3.8 => 5 + 3 = 8

(int)(f + d) => (int) (5.5 + 3.8) => (int)(9.3) => 9, so this also return false.
So, c is not incremented at all. Hence c remains 'a'.

Back to Question without Answer

61. QID - 2.1325 : Using Operators and Decision Constructs

What will be the result of attempting to compile and run the following class?

```
public class IfTest{
   public static void main(String args[]){
      if (true)
      if (false)
      System.out.println("True False");
      else
      System.out.println("True True");
   }
}
```

Correct Option is : D

~~**A.**~~ The code will fail to compile because the syntax of the `if` statement is not correct.

> It is perfectly valid.

~~**B.**~~ The code will fail to compile because the values in the condition bracket are invalid.

> Any expression that returns a boolean is valid. false and true are valid expressions that return boolean.

~~**C.**~~ The code will compile correctly and will not display anything.

D. The code will compile correctly and will display `True True`.

~~**E.**~~ The code will compile correctly but will display `True False`

Explanation:

This code can be rewritten as follows:

```
public class IfTest{
    public static void main(String args[]) {
        if (true) {
            if (false) {
                System.out.println("True False");
            } else {
                System.out.println("True True");
            }
        }
    }
}
```

Notice how the last "else" is associated with the last "if" and not the first "if". Now, the first if condition returns true so the next 'if' will be executed. In the second 'if' the condition returns false so the else part will be evaluated which prints 'True True'.

Back to Question without Answer

62. QID - 2.1088 : Using Operators and Decision Constructs

Consider:
`o1` and `o2` denote two object references to two different objects of same class.
Which of the following statements are true?

Correct Options are : C D

~~**A.**~~ `o1.equals(o2)` will always be false.

> It depends on how the equals method is overridden. If it is not overridden, then it will return false.

~~**B.**~~ `o1.hashCode() == o2.hashCode()` will always be false.

> `hashCode()` can be overridden and so the given statements is not true.

C. `o1 == o2` will always be false.

> The == operator compares whether the two references are pointing to the same object or not. Here, they are not, so it returns false.

D. Nothing can be said about `o1.equals(o2)` regarding what it will return based on the given information.

> It depends on how the class implements this method.

~~**E.**~~ Nothing can be said about `o1 == o2`.

> It will always return false if references are to two different objects.

Explanation:

Note that both `equals()` and `hashCode()` methods can be overridden by the programmer so you can't say anything about what they will return without looking at the code.

Back to Question without Answer

63. QID - 2.1353 : Using Operators and Decision Constructs

Which of the lines will cause a compile time error in the following program?

```
public class MyClass{
   public static void main(String args[]){
      char c;
      int i;
      c = 'a';//1
      i = c;  //2
      i++;    //3
      c = i;  //4
      c++;    //5
   }
}
```

Correct Option is : D

~~**A.**~~ line 1

~~**B.**~~ line 2

~~**C.**~~ line 3

D. line 4

~~**E.**~~ line 5

Explanation:

1. A char value can ALWAYS be assigned to an int variable, since the int type is

wider than the char type. So line 2 is valid.

2. Line 4 will not compile because it is trying to assign an int to a char. Although the value of i can be held by the char but since 'i' is not a constant but a variable, implicit narrowing will not occur.

Here is the rule given in JLS:
A narrowing primitive conversion may be used if all of the following conditions are satisfied:
The expression is a constant expression of type int.
The type of the variable is byte, short, or char.
The value of the expression (which is known at compile time, because it is a constant expression) is representable in the type of the variable.
Note that narrowing conversion does not apply to long or double.
so, char ch = 30L; will fail although 30 is representable by a char.

Back to Question without Answer

64. QID - 2.1183 : Using Operators and Decision Constructs

Given the following LOCs:

```
int rate = 10;
XXX amount = 1 - rate/100*1 - rate/100;
```

What can XXX be?

Correct Option is : F

~~**A.**~~ only int or long

~~**B.**~~ only long or double

~~**C.**~~ only double

~~**D.**~~ double or float

~~**E.**~~ long or double but not int or float.

F. int, long, float or double

Explanation:

Note that none of the terms in the expression `1 - rate/100*1 - rate/100;` is double or float. They are all ints. So the result of the expression will be an int.
Since an int can be assigned to a variable of type int, long, float or double, amount can be int, long, float or double.

Back to Question without Answer

65. QID - 2.1179 : Using Operators and Decision Constructs

The following code snippet will print 'true'.

```
short s = Short.MAX_VALUE;
char c = s;
System.out.println( c == Short.MAX_VALUE);
```

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

This will not compile because a short VARIABLE can NEVER be assigned to a char without explicit casting. A short CONSTANT can be assigned to a char only if the value fits into a char.

short s = 1; byte b = s; => this will also not compile because although value is small enough to be held by a byte but the Right Hand Side i.e. s is a variable and not a constant.
final short s = 1; byte b = s; => This is fine because s is a constant and the value fits into a byte.
final short s = 200; byte b = s; => This is invalid because although s is a constant but the value does not fit into a byte.

Implicit narrowing occurs only for byte, char, short, and int. Remember that it does not occur for long, float, or double. So, this will not compile: int i = 129L;

Back to Question without Answer

66. QID - 2.1177 : Using Operators and Decision Constructs

Which of the following statements are true?

Correct Option is : B

~~**A.**~~ For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `true`.

> `instanceof` operator does not accept objects as the right hand side operand. The operand at the right hand side should be a class. Therefore, this expression will not compile.

B. For any non-null reference `o1`, the expression `(o1 instanceof Object)` will always yield `true`.

> Because all objects in Java derive from Object class.

~~**C.**~~ For any non-null reference `o1`, the expression `(o1 instanceof o1)` will always yield `false`.

> It is wrong for the same reason as option 1.

~~**D.**~~ For any non-null reference `o1`, the expression `(o1 instanceof Object)` may yield `false`.

> Since all objects in Java derive from Object class, there is no way it will ever return `false`.

~~**E.**~~ None of the above.

Explanation:

You should understand here that `instanceof` operator returns true even if the Right Hand Side is a super class.
For example,

```
class Animal {}
class Dog extends Animal { }
Dog d = new Dog();
```

Now, `d instanceof Animal` will be true because even though `d` is actually an instance of `Dog`, since `Dog` is a subclass of `Animal`, `Dog` IS-A `Animal`.

Back to Question without Answer

67. QID - 2.973 : Using Operators and Decision Constructs

Which operators will always evaluate all the operands?

Correct Options are : B E

~~**A.** &&~~

B. |

~~**C.** ||~~

~~**D.** ? :~~

> If the condition before ? returns true, only the first operand will be evaluated, otherwise only the second operand is evaluated.

E. %

> All mathematical operators evaluate all the operands.

Explanation:

`||` and `&&` are also known as short circuit operators since they do not evaluate the rest of the expression if the value of the expression can be determined by just evaluating part of the expression for example `( true || bool = false)` will not assign `false` to `bool` because the value of the expression can be told just by seeing the first part i.e. `true`. But `( true | bool = false)` will assign `false` to `bool`.

Back to Question without Answer

68. QID - 2.1310 : Using Operators and Decision Constructs

Consider the following method...

```
public static void ifTest(boolean flag){
   if (flag)    //1
   if (flag)    //2
   if (flag)    //3
   System.out.println("False True");
   else         //4
   System.out.println("True False");
   else         //5
   System.out.println("True True");
   else         //6
   System.out.println("False False");
}
```

Which of the following statements are correct ?

Correct Options are : A D

A. If run with an argument of 'false', it will print 'False False'

~~**B.**~~ If run with an argument of 'false', it will print 'True True'

~~**C.**~~ If run with an argument of 'true', it will print 'True False'

D. It will never print 'True True'

~~**E.**~~ It will not compile.

Explanation:

Look at it like this:

```
if (flag)       //1
 {
     if (flag)        // 2
     {
          if (flag)         //3
          {
                System.out.println("False True");
          }
          else             //4
          {
                System.out.println("True False");
          }
     }
     else            //5
     {
           System.out.println("True True");
     }
 }
else            //6
{
      System.out.println("False False");
 }
```

Note that if and else do not cascade. They are like opening an closing brackets. So, else at //4 is associated with if at //3 and else at //5 is associated with if at //2

Back to Question without Answer

69. QID - 2.980 : Using Operators and Decision Constructs

Given the following class definitions, the expression

```
(obj instanceof A) && ! (obj instanceof C) && ! (obj instanceof D)
```

correctly identifies whether the object referred to by obj was created by instantiating class B rather than classes A, C and D?

```
class A {}
class B extends A {}
class C extends B {}
class D extends C {}
```

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

The given expression will not be able to distinguish between an object of class A and an object of class B. It will return true in both the cases. Also, The last part `!(obj instanceof D)` of the given expression is redundant because anything which is not instance of C cannot be an instanceof D either!

Correct expression would be `(obj instanceof B) && ! (obj instanceof C)`. This will return true only if obj points to an object of class B and not of A, C, or D.

Back to Question without Answer

Working with Inheritance

Exam Objectives -

Implement inheritance Develop code that demonstrates the use of polymorphism Differentiate between the type of a reference and the type of an object Determine when casting is necessary Use super and this to access objects and constructors Use abstract classes and interfaces

01. QID - 2.1095

What will be the result of compiling and running the following code?

```
class Base{
   public short getValue(){ return 1; } //1
}
class Base2 extends Base{
   public byte getValue(){ return 2; } //2
}
public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Select 1 option

A. It will print 1

B. It will print 2.

C. Compile time error at //1

D. Compile time error at //2

E. Compile time error at //3

Check Answer

02. QID - 2.1273

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      A o1 = new C( );
      B o2 = (B) o1;
      System.out.println(o1.m1( ) );
      System.out.println(o2.i );
   }
}
class A { int i = 10;  int m1( ) { return i; } }
class B extends A { int i = 20;  int m1() { return i; } }
class C extends B { int i = 30;  int m1() { return i; } }
```

Select 1 option

A. The program will fail to compile.

B. Class cast exception at runtime.

C. It will print 30, 20.

D. It will print 30, 30.

E. It will print 20, 20.

Check Answer

03. QID - 2.1145

Assume the following declarations:

```
class A{ }
class B extends A{ }
class C extends B{ }

class X{
   B getB(){ return new B(); }
}

class Y extends X{
  // method declaration here
}
```

Which of the following methods can be inserted in class Y?

Select 2 options

A. `public C getB(){ return new B(); }`

B. `protected B getB(){ return new C(); }`

C. `C getB(){ return new C(); }`

D. `A getB(){ return new A(); }`

Check Answer

04. QID - 2.1135

Which of the following statements is/are true?

Select 1 option

A. Subclasses must define all the abstract methods that the superclass defines.

B. A class implementing an interface must define all the methods of that interface.

C. A class cannot override the super class's constructor.

D. It is possible for two classes to be the superclass of each other.

E. An interface can implement multiple interfaces.

Check Answer

05. QID - 2.1337

Consider the following code:

```
class A{
   A() {  print();   }
   void print() { System.out.println("A"); }
}
class B extends A{
   int i =   Math.round(3.5f);
   public static void main(String[] args){
      A a = new B();
      a.print();
   }
   void print() { System.out.println(i); }
}
```

What will be the output when class B is run ?

Select 1 option

A. It will print A, 4.

B. It will print A, A

C. It will print 0, 4

D. It will print 4, 4

E. None of the above.

Check Answer

06. QID - 2.1300

An abstract method cannot be overridden.

Select 1 option

A. True

B. False

Check Answer

07. QID - 2.836

Which of the following statements are correct?

Select 3 options

A. An abstract class can be extended by an abstract or a concrete class.

B. A concrete class can be extended by an abstract or a concrete class.

C. An interface can be extended by another interface.

D. An interface can be extended by an abstract class.

E. An interface can be extended by a concrete class.

F. An abstract class cannot implement an interface.

Check Answer

08. QID - 2.1181

What would be the result of attempting to compile and run the following code?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      B c = new C();
      System.out.println(c.max(10, 20));
   }
}
class A{
   int max(int x, int y)  { if (x>y) return x; else return y; }
}
class B extends A{
  int max(int x, int y)  {  return 2 * super.max(x, y) ; }
}
class C extends B{
  int max(int x, int y)  {  return super.max( 2*x, 2*y); }
}
```

Select 1 option

A. The code will fail to compile.

B. Runtime error.

C. The code will compile without errors and will print 80 when run.

D. The code will compile without errors and will print 40 when run.

E. The code will compile without errors and will print 20 when run.

Check Answer

09. QID - 2.1086

What will be the result of compiling and running the following code?

```
class Base{
   public Object getValue(){ return new Object(); } //1
}

class Base2 extends Base{
   public String getValue(){ return "hello"; } //2
}

public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Select 1 option

A. It will print the hash code of the object.

B. It will print `hello`.

C. Compile time error at `//1`.

D. Compile time error at `//2`.

E. Compile time error at `//3`.

Check Answer

10. QID - 2.1143

Which of the given statements are correct for a method that overrides the following method:

```
public Set getSet(int a) {...}
```

Select 3 options

A. Its return type must be declared as `Set`.

B. It may return HashSet.

C. It can declare any Exception in throws clause

D. It can declare any RuntimeException in throws clause.

E. It can be abstract.

Check Answer

11. QID - 2.1256

Which of these statements are true?

Select 2 options

A. A `super( )` or `this( )` call must always be provided explicitly as the first statement in the body of the constructor.

B. If a subclass does not have any declared constructors, the implicit default constructor of the subclass will have a call to `super( )`.

C. If neither `super( )` or `this( )` is declared as the first statement of the body of a constructor, then `this( )` will implicitly be inserted as the first statement.

D. `super(...)` can only be called in the first line of the constructor but `this(...)` can be called from anywhere.

E. You can either call `super(...)` or `this(...)` but not both.

Check Answer

12. QID - 2.1231

Given the following interface definition, which definitions are valid?

```
interface I1{
   void setValue(String s);
   String getValue();
}
```

Select 2 options

A.
```
class A extends I1{
   String s;
   void setValue(String val) { s = val; }
   String getValue() { return s; }
}
```

B.
```
interface I2 extends I1{
   void analyse();
}
```

C.
```
abstract class B implements I1{
   int getValue(int i) { return 0; }
}
```

D.
```
interface I3 implements I1{
   void perform_work();
}
```

Check Answer

13. QID - 2.1328

Consider the following classes :

```
interface I{
}
class A implements I{
}

class B extends A {
}

class C extends B{
}
```

And the following declarations:

```
A a = new A();
B b = new B();
```

Identify options that will compile and run without error.

Select 1 option

A. `a = (B)(I)b;`

B. `b = (B)(I) a;`

C. `a = (I) b;`

D. `I i = (C) a;`

Check Answer

14. QID - 2.905

Given the following line of code:

```
List students = new ArrayList();
```

Identify the correct statement:

Select 1 option

A. The reference type is List and the object type is ArrayList.

B. The reference type is ArrayList and the object type is ArrayList.

C. The reference type is ArrayList and the object type is List.

D. The reference type is List and the object type is List.

Check Answer

15. QID - 2.945

Which of the following statements are true?

Select 2 options

A. Private methods cannot be overridden in subclasses.

B. A subclass can override any method in a non-final superclass.

C. An overriding method can declare that it throws a wider spectrum of checked exceptions than the method it is overriding.

D. The parameter list of an overriding method must be a subset of the parameter list of the method that it is overriding.

E. The overriding method may opt not to declare any throws clause even if the original method has a throws clause.

Check Answer

16. QID - 2.1136

Which of the given statements are correct about the following code?

```
//Filename: TestClass.java
class TestClass{
   public static void main(String[] args){
      A a = new A();
      B b = new B();
   };
}
class A implements T1, T2{}
class B extends A implements T1{}
interface T1 { }
interface T2 { }
```

Select 4 options

A. `(a instanceof T1)` will return true.

B. `(a instanceof T2)` will return true.

C. `(b instanceof T1)` will return true.

D. `(b instanceof T2)` will return true.

E. `(b instanceof A)` will return false.

Check Answer

17. QID - 2.1107

Which of the following are correct about "encapsulation"?

Select 2 options

A. Encapsulation is same as polymorphism.

B. It helps make sure that clients have no accidental dependence on the choice of representation

C. It helps avoiding name clashes as internal variables are not visible outside.

D. Encapsulation makes sure that messages are sent to the right object at run time.

E. Encapsulation helps you inherit the properties of another class.

Check Answer

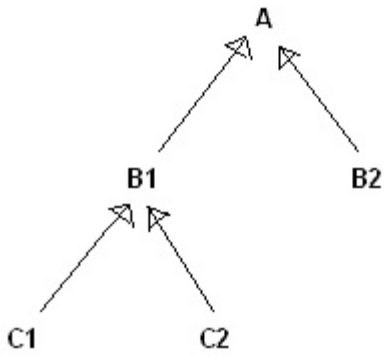
18. QID - 2.1287

Consider the following class hierarchy shown in the image. (B1 and B2 are subclasses of A and C1, C2 are subclasses of B1)

Assume that method `public void m1(){ ... }` is defined in all of these classes EXCEPT B1 and C1.

Assume that `"objectOfXX"` means a variable that points to an object of class XX. So, `objectOfC1` means a reference variable that is pointing to an object of class C1.

Which of the following statements are correct?



Select 1 option

A. `objectOfC1.m1();` will cause a compilation error.

B. `objectOfC2.m1();` will cause A's `m1()` to be called.

C. `objectOfC1.m1();` will cause A's `m1()` to be called.

D. `objectOfB1.m1();` will cause an exception at runtime.

E. `objectOfB2.m1();` will cause an exception at runtime.

Check Answer

19. QID - 2.931

Consider the following classes :

```
class A{
    public static void sM1() {  System.out.println("In base static"),
}
class B extends A{
Line 1 :   // public static void sM1() {  System.out.println("In sub
Line 2 :   // public  void sM1() {  System.out.println("In sub non-s
}
```

Which of the following statements are true?

Select 2 options

A. class B will not compile if line 1 is uncommented.

B. class B will not compile if line 2 is uncommented.

C. class B will not compile if line 1 and 2 are both uncommented.

D. Only Option 2 is correct.

E. Only Option 3 is correct.

Check Answer

20. QID - 2.936

Consider the following code:

```
class Super{
  static{ System.out.print("super "); }
}
class One{
  static {  System.out.print("one "); }
}
class Two extends Super{
  static {  System.out.print("two "); }
}
class Test{
  public static void main(String[] args){
     One o = null;
     Two t = new Two();
  }
}
```

What will be the output when class Test is run ?

Select 1 option

A. It will print `one, super` and `two`.

B. It will print `one, two` and `super`.

C. It will print `super` and `two`.

D. It will print `two` and `super`

E. None of the above.

Check Answer

21. QID - 2.872

Consider the following classes:

```
class A {
      public int getCode(){ return 2;}
}

class AA extends A {
  public void doStuff() {
  }
}
```

Given the following two declarations, which of the options will compile?

```
A a = null;
AA aa = null;
```

Select 4 options

A. `a = (AA)aa;`

B. `a = new AA();`

C. `aa = new A();`

D. `aa = (AA) a;`

E. `aa = a;`

F. `((AA)a).doStuff();`

Check Answer

22. QID - 2.1292

Given the definitions of I and Klass, complete the definition of SubClass so that it extends from Klass and implements I.
Use minimum number of elements.

```
interface I
{
    void m1();
}

abstract class Klass
{
    void m1() { };
}

class SubClass [ ] [ ] [ ] [ ]
{
    [ ] [ ] [ ] [ ] [ ] [ ]
}
```

extends implements Klass

I public void m1(){ }

Check Answer

23. QID - 2.1064

Consider this code:

```
interface X1{ }
interface X2{ }
class A { }
class B extends A implements X1{ }
class C extends B implements X2{
   D d = new D();
}
class D { }
```

Which of the following statements are true?

Select 3 options

A. D is-a B.

B. B has-a D.

C. C is-a A

D. C is-a X1

E. C is-a X2

Check Answer

24. QID - 2.1015

Which statement regarding the following code is correct?

```
class A{
   public int i = 10;
   private int j = 20;

}

class B extends A{
   private int i = 30; //1
   public int k = 40;

}

class C extends B{
}

public class TestClass{
   public static void main(String args[]){
      C c = new C();
      System.out.println(c.i); //2
      System.out.println(c.j); //3
      System.out.println(c.k);
   }
}
```

Select 1 option

A. It will print `10` and `40` if //3 is commented.

B. It will print `40` if //2 and //3 are commented.

C. It will not compile because of `//1`.

D. It will compile if `//2` is commented.

E. None of these.

Check Answer

25. QID - 2.1208

Given the following class definition:

```
class A{
  protected int i;
  A(int i) {    this.i = i;    }

}
// 1 : Insert code here
```

Which of the following would be a valid class that can be inserted at //1 ?

Select 2 options

A. `class B {}`

B. `class B extends A {}`

C. `class B extends A {  B()  {  System.out.println("i = " + i); }  }`

D. `class B { B() {} }`

Check Answer

26. QID - 2.1091

What will the following code print when TestClass is run?

```
class Employee {

  static int i = 10;  {
      i = 15;
      System.out.print(" Employee "+i);
  }
  static { System.out.print(" Employee static "+i); }
}

class Manager extends Employee {
  static {
    i = 45;
    System.out.print(" Manager static ");
  }{
    i = 30;
    System.out.print(" Manager "+i);
  }
}

class Owner extends Manager{
  static { System.out.println("Owner"); }
}

public class TestClass {
  public static void main(String[] args) {
      Manager m = new Manager();
  }
}
```

Select 1 option

A. `Employee static 10 Manager static Employee 10 Manager 30`

B. `Employee static 10 Manager static Owner Manager 30`

C. `Employee static 10 Manager static Employee 15 Manager 30`

D. `Employee static 10 Manager static 15 Manager 20`

E. It will not compile.

Check Answer

27. QID - 2.1320

What will the following code print when compiled and run?

```
class ABCD{
   int x = 10;
   static int y = 20;
}
class MNOP extends ABCD{
   int x = 30;
   static int y = 40;
}

public class TestClass {
   public static void main(String[] args) {
     System.out.println(new MNOP().x+", "+new MNOP().y);
   }
}
```

Select 1 option

A. 10, 40

B. 30, 20

C. 10, 20

D. 30, 40

E. 20, 30

F. Compilation error.

Check Answer

28. QID - 2.970

Consider the following classes...

```
class Car{
   public int gearRatio = 8;
   public String accelerate() {  return "Accelerate : Car";  }
}
class SportsCar extends Car{
   public int gearRatio = 9;
   public String accelerate() {  return  "Accelerate : SportsCar";
   public static void main(String[] args){
      Car c = new SportsCar();
      System.out.println( c.gearRatio+"  "+c.accelerate() );
   }
}
```

What will be printed when SportsCar is run?

Select 1 option

A. 8 Accelerate : Car

B. 9 Accelerate : Car

C. 8 Accelerate : SportsCar

D. 9 Accelerate : SportsCar

E. None of the above.

Check Answer

29. QID - 2.1171

Consider the following variable declaration within the definition of an interface:

```
int i = 10;
```

Which of the following declarations defined in a non-abstract class, is equivalent to the above?

Select 1 option

A. `public static int i = 10;`

B. `public final int i = 10;`

C. `public static final int i = 10;`

D. `public int i = 10;`

E. `final int i = 10;`

Check Answer

30. QID - 2.1049

Consider that you are writing a set of classes related to a new Data Transmission Protocol and have created your own exception hierarchy derived from java.lang.Exception as follows:

```
enthu.trans.ChannelException
              +-- enthu.trans.DataFloodingException,
                    enthu.trans.FrameCollisionException
```

You have a TransSocket class that has the following method:

```
long connect(String ipAddr) throws ChannelException
```

Now, you also want to write another "AdvancedTransSocket" class, derived from "TransSocket" which overrides the above mentioned method. Which of the following are valid declaration of the overriding method?

Select 2 options

A. `int connect(String ipAddr) throws DataFloodingException`

B. `int connect(String ipAddr) throws ChannelException`

C. `long connect(String ipAddr) throws FrameCollisionException`

D. `long connect(String ipAddr) throws Exception`

E. `long connect(String str)`

Check Answer

31. QID - 2.1003

Given the following code, which statements are true?

```
interface Automobile { String describe(); }

class FourWheeler implements Automobile{
   String name;
   public String describe(){ return " 4 Wheeler " + name; }
}

class TwoWheeler extends FourWheeler{
    String name;
    public String describe(){ return " 2 Wheeler " + name; }
}
```

Select 3 options

A. An instance of `TwoWheeler` is also an instance of `FourWheeler`.

B. An instance of `TwoWheeler` is a valid instance of `Automobile`.

C. The use of inheritance is not justified here because a `TwoWheeler` is not really a `FourWheeler`.

D. The code will compile only if `name` is removed from `TwoWheeler`.

E. The code will fail to compile.

Check Answer

32. QID - 2.993

Which is the first line that will cause compilation to fail in the following program?

```
// Filename: A.java
class A{
   public static void main(String args[]){
      A a = new A();
      B b = new B();
      a = b;  // 1
      b = a;  // 2
      a = (B) b; // 3
      b = (B) a; // 4
   }
}
class B extends A { }
```

Select 1 option

A. At Line 1.

B. At Line 2.

C. At Line 3.

D. At Line 4.

E. None of the above.

Check Answer

33. QID - 2.1035

Which of the following statements are true?

Select 2 options

A. The `extends` keyword is used to specify inheritance.

B. subclass of a non-abstract class cannot be declared `abstract`.

C. subclass of an abstract class can be declared `abstract`.

D. subclass of a final class cannot be `abstract`.

E. A class, in which all the members are declared `private`, cannot be declared `public`.

Check Answer

34. QID - 2.1009

Consider the following code:

```
class Super { static String ID = "QBANK"; }

class Sub extends Super{
   static { System.out.print("In Sub"); }
}
public class Test{
   public static void main(String[] args){
      System.out.println(Sub.ID);
   }
}
```

What will be the output when class Test is run?

Select 1 option

A. It will print `In Sub` and `QBANK`.

B. It will print `QBANK`.

C. Depends on the implementation of JVM.

D. It will not even compile.

E. None of the above.

Check Answer

35. QID - 2.1215

Consider the following classes :

```
class A{
   public void mA(){ };
}

class B extends A {
   public void mA(){ }
   public void mB() { }
}

class C extends B {
   public void mC(){ }
}
```

and the following declarations:

```
A x = new B(); B y = new B(); B z = new C();
```

Which of the following calls are polymorphic calls?

Select 3 options

A. `x.mA();`

B. `x.mB();`

C. `y.mA();`

D. `z.mC();`

E. `z.mB();`

Check Answer

36. QID - 2.1329

Which of the following method definitions will prevent overriding of that method?

Select 4 options

A. `public final void m1()`

B. `public static void m1()`

C. `public static final void m1()`

D. `public abstract void m1()`

E. `private void m1()`

Check Answer

37. QID - 2.838

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Flyer) System.out.println("f is a Flyer");
        if(e instanceof Bird) System.out.println("e is a Bird");
        if(b instanceof Bird) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Select 1 option

A. It will not compile.

B. It will throw an exception when run.

C. f is a Flyer
e is a Bird

D. f is a Flyer

E. e is a Bird

Check Answer

38. QID - 2.1150

Which of the following are valid declarations inside an interface?

Select 2 options

A. `void compute();`

B. `public void compute();`

C. `public final void compute();`

D. `static void compute();`

E. `protected void compute();`

Check Answer

39. QID - 2.952

Which of the following lines of code that, when inserted at line 1, will make the overriding method in SubClass invoke the overridden method in BaseClass on the current object with the same parameter.

```
class BaseClass{
   public void print(String s) {  System.out.println("BaseClass :"+s)
}
class SubClass extends BaseClass{
   public void print(String s){
      System.out.println("SubClass :"+s);
      // Line 1
   }
   public static void main(String args[]){
      SubClass sc = new SubClass();
      sc.print("location");
   }
}
```

Select 1 option

A. `this.print(s);`

B. `super.print(s);`

C. `print(s);`

D. `BaseClass.print(s);`

Check Answer

40. QID - 2.1117

Given the following classes, what will be the output of compiling and running the class Truck?

```
class Automobile{
   public void drive() {  System.out.println("Automobile: drive");
}

public class Truck extends Automobile{
   public void drive() {  System.out.println("Truck: drive");   }
   public static void main (String args [ ]){
      Automobile  a = new Automobile();
      Truck t  = new Truck();
      a.drive(); //1
      t.drive(); //2
      a = t;     //3
      a.drive(); //4
   }
}
```

Select 1 option

A. Compiler error at line 3.

B. Runtime error at line 3.

C. It will print:

```
Automobile: drive
Truck: drive
Automobile: drive
```

in that order.

D. It will print:

```
Automobile: drive
Truck: drive
Truck: drive
```

in that order.

E. It will print:

```
Automobile: drive
Automobile: drive
Automobile: drive
```

in that order.

Check Answer

41. QID - 2.995

Which of these statements about interfaces are true?

Select 3 options

A. Interfaces are abstract by default.

B. An interface can have static methods.

C. All methods in an interface are abstract although you need not declare them to be so.

D. Fields of an interface may be declared as transient or volatile but not synchronized.

E. interfaces cannot be final.

Check Answer

42. QID - 2.1113

Which of the following is a legal return type of a method overriding the given method:

```
public Object  myMethod() {...}
```
(Select the best option.)

Select 1 option

A. Object

B. String

C. Return type can be any object since all objects can be cast to Object.

D. void

E. None of the above.

Check Answer

43. QID - 2.983

Consider the following classes:

```
class A implements Runnable{ ...}
class B extends A implements Observer { ...}
```

(Assume that Observer has no relation to Runnable.)

and the declarations :

```
A a = new A() ;
B b = new B();
```

Which of the following Java code fragments will compile and execute without throwing exceptions?

Select 2 options

A. `Object o = a; Runnable r = o;`

B. `Object o = a; Runnable r = (Runnable) o;`

C. `Object o = a; Observer ob = (Observer) o ;`

D. `Object o = b; Observer o2 = o;`

E. `Object o = b; Runnable r = (Runnable) b;`

Check Answer

44. QID - 2.1072

What, if anything, is wrong with the following code?

```
// Filename: TestClass.java
class TestClass implements T1, T2{
   public void m1(){}
}
interface T1{
   int VALUE = 1;
   void m1();
}
interface T2{
   int VALUE = 2;
   void m1();
}
```

Select 1 option

A. `TestClass` cannot implement them both because it leads to ambiguity.

B. There is nothing wrong with the code.

C. The code will work fine only if `VALUE` is removed from one of the interfaces.

D. The code will work fine only if `m1()` is removed from one of the interfaces.

E. None of the above.

Check Answer

45. QID - 2.1205

Which of these statements about interfaces are true?

Select 3 options

A. Interfaces permit multiple implementation inheritance.

B. Unlike a class, an interface can extend from multiple interfaces.

C. Members of an interface are never static.

D. Members of an interface may be static.

E. Interfaces cannot be final.

Check Answer

46. QID - 2.1127

Consider the following class and interface definitions (in separate files):

```
public class Sample implements IInt{
 public static void main(String[] args){
  Sample s = new Sample();  //1
  int j = s.thevalue;       //2
  int k = IInt.thevalue;    //3
  int l = thevalue;         //4
 }
}
public interface IInt{
 int thevalue = 0;
}
```

What will happen when the above code is compiled and run?

Select 1 option

A. It will give an error at compile time at line //1.

B. It will give an error at compile time at line //2.

C. It will give an error at compile time at line //3

D. It will give an error at compile time at line //4.

E. It will compile and run without any problem.

Check Answer

47. QID - 2.1288

What will the following code print?

```
class Baap {
    public int h = 4;
    public int getH(){ System.out.println("Baap   "+h); return h; }
}

class Beta extends Baap {
    public int h = 44;
    public int getH(){ System.out.println("Beta   "+h); return h; }
    public static void main(String[] args) {
        Baap b = new Beta();
        System.out.println(b.h+"   "+b.getH());
        Beta bb = (Beta) b;
        System.out.println(bb.h+"   "+bb.getH());
    }
}
```

-------- Output ---------

Check Answer

48. QID - 2.1359

Consider :

```
class A {  public void perform_work(){}  }
class B extends A {  public void perform_work(){}  }
class C extends B {  public void perform_work(){}  }
```

How can you let `perform_work()` method of A to be called from an instance method in C?

Select 1 option

A. `( (A) this ).perform_work( );`

B. `super.perform_work( );`

C. `super.super.perform_work( );`

D. `this.super.perform_work( );`

E. It is not possible.

Check Answer

49. QID - 2.1019

You are modeling a class hierarchy for living things. You have a class `LivingThing` which has an abstract method `reproduce()`.
Now, you want to have 2 subclasses of `LivingThing` - `Plant` and `Animal`. Both do reproduce but the mechanisms are different. What would you do?

Select 1 option

A. Overload the `reproduce` method in Plant and Animal classes

B. Overload the `reproduce` method in LivingThing class.

C. Override the `reproduce` method in Plant and Animal classes

D. Either overload or override `reproduce` in `Plant` and `Animal` classes, it depends on the preference of the designer.

Check Answer

50. QID - 2.1130

You want to invoke the overridden method (the method in the base class) from the overriding method (the method in the derived class) named `m()`.
Which of the following constructs which will let you do that?

Select 1 option

A. `super.m();`

B. `super.this();`

C. `base.m();`

D. `parent.m();`

E. `super();`

Check Answer

51. QID - 2.1367

What should be inserted in the code given below at line marked //10:

```
class MyClass{
}

class MyComparable implements Comparable<MyClass>{
   public int compareTo(  *INSERT CODE HERE*  x ){ //10
         return 0;
   }
}
```

Select 1 option

A. `Object`

B. `MyClass`

C. `Object<MyClass>`

D. `Comparable<MyClass>`

E. Comparable

Check Answer

52. QID - 2.1165

Drag blue items xxx and yyy on yellow targets to complete the code.

xxx yyy

```
class XXX {
    public void m() throws Exception {
        throw new Exception();
    }
}
class YYY extends XXX{
    public void m(){    }
}

public class TestClass {

    public static void main(String[] args)
    {
        [          ]  s  = new [          ]();

        s.m();
    }
}
```

Check Answer

53. QID - 2.1261

Consider the following code:

```
public class SubClass extends SuperClass{
     int i, j, k;
     public SubClass( int m, int n )     {  i = m ;  j = m ;  } //1
     public SubClass( int m )  {   super(m );   } //2
 }
```

Which of the following constructors MUST exist in SuperClass for SubClass to compile correctly?

Select 2 options

A. It is ok even if no explicit constructor is defined in SuperClass

B. `public SuperClass(int a, int b)`

C. `public SuperClass(int a)`

D. `public SuperClass()`

E. only `public SuperClass(int a)` is required.

Check Answer

54. QID - 2.1129

Consider the following class:

```
public class PortConnector{
   public PortConnector(int port) throws IOException{
   ...lot of valid code.
   }
   ...other valid code.
}
```

You want to write another class `CleanConnector` that extends from `PortConnector`. Which of the following statements should hold true for `CleanConnector` class?

Select 1 option

A. It is not possible to define `CleanConnector` that does not throw `IOException` at instantiation.

B. `PortConnector` class itself is not valid because you cannot throw any exception from a constructor.

C. `CleanConnector`'s constructor cannot throw any exception other than `IOException`.

D. `CleanConnector`'s constructor cannot throw any exception other than subclass of `IOException`.

E. `CleanConnector`'s constructor cannot throw any exception other than superclass of `IOException`.

F. None of these.

Check Answer

55. QID - 2.1315

Note: This question may be considered too advanced for this exam.

Given the declaration

```
interface Worker { void perform_work(); }
```

which of the following methods/classes are valid?

Select 2 options

A.
```
Worker getWorker(int i){
   return new Worker(){   public void perform_work() {
        System.out.println(i); }    };
}
```

B.
```
Worker getWorker(final int i){
   return new Worker() {   public void perform_work() {
   System.out.println(i); }    };
}
```

C.
```
Worker getWorker(int i){
   int x = i;
   class MyWorker implements Worker  {
         public void perform_work() { System.out.println(x); }    };
   return new MyWorker();
}
```

D.
```
Worker getWorker(final int i){
   class MyWorker implements Worker {
       public void perform_work() { System.out.println(i); }    };
   return new MyWorker();
}
```

Check Answer

56. QID - 2.1201

Consider the following interface definition:

```
interface Bozo{
         int type = 0;
         public void jump();
}

Now consider the following class:

public class Type1Bozo implements Bozo{
         public Type1Bozo(){
            type = 1;
         }

         public void jump(){
            System.out.println("jumping..."+type);
         }

         public static void main(String[] args){
            Bozo b = new Type1Bozo();
            b.jump();
         }
}
```

What will the program print when compiled and run?

Select 1 option

A. jumping...0

B. jumping...1

C. This program will not compile.

D. It will throw an exception at runtime.

Check Answer

57. QID - 2.1344

Complete the following code by dragging one of the blue items on to the yellow target.

Object MyClass Object<MyClass>

Comparable<MyClass> Comparable

```
class MyClass
{
}

class MyComparable implements Comparable<MyClass>
{
    public int compareTo(              x)
    {
        return 0;
    }
}
```

Check Answer

58. QID - 2.1309

What can be inserted at //1 and //2 in the code below so that it can compile without errors:

```
class Doll{
    String name;
    Doll(String nm){
        this.name = nm;
    }
}

class Barbie extends Doll{
    Barbie(){
        //1
    }
    Barbie(String nm){
        //2
    }
}
  public class TestClass {
       public static void main(String[] args) {
          Barbie b = new Barbie("mydoll");
       }
   }
```

Select 2 options

A. `this("unknown");` at 1 and `super(nm);` at 2

B. `super("unknown");` at 1 and `super(nm);` at 2

C. `super();` at 1 and `super(nm);` at 2

D. `super();` at 1 and `Doll(nm);` at 2

E. `super("unknown");` at 1 and `this(nm);` at 2

F. `Doll();` at 1 and `Doll(nm);` at 2

Check Answer

59. QID - 2.996

Consider the following program:

```
class Game {
  public void play() throws Exception   {
    System.out.println("Playing...");
  }
}

class Soccer extends Game {
   public void play(String ball)    {
      System.out.println("Playing Soccer with "+ball);
   }
}

public class TestClass {
   public static void main(String[] args) throws Exception  {
       Game g = new Soccer();
       // 1
       Soccer s = (Soccer) g;
       // 2
   }
}
```

Which of the given options can be inserted at //1 and //2?

Select 2 options

A. It will not compile as it is.

B. It will throw an `Exception` at runtime if it is run as it is.

C. `g.play();` at //1 and `s.play("cosco");` at //2

D. `g.play();` at //1 and `s.play();` at //2

E. `g.play("cosco");` at //1 and `s.play("cosco");` at //2

Check Answer

60. QID - 2.1047

An overriding method must have a same parameter list and the same return type as that of the overridden method.

Select 1 option

A. True

B. False

Check Answer

61. QID - 2.837

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Bird) System.out.println("f is a Bird");
        if(e instanceof Flyer) System.out.println("e is a Flyer");
        if(b instanceof Flyer) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Select 1 option

A. It will not compile.

B. It will throw an exception when run.

C. f is a Bird
e is a Flyer

D. f is a Bird

E. e is a Flyer

Check Answer

62. QID - 2.1195

Which letters will be printed when the following program is run?

```
public class TestClass{
  public static void main(String args[]){
    B b = new C();
    A a = b;
    if (a instanceof A) System.out.println("A");
    if (a instanceof B) System.out.println("B");
    if (a instanceof C) System.out.println("C");
    if (a instanceof D) System.out.println("D");
  }
}
class A { }
class B extends A { }
class C extends B { }
class D extends C { }
```

Select 3 options

A. A will be printed.

B. B will be printed.

C. C will be printed.

D. D will be printed.

Check Answer

63. QID - 2.1147

Given the following code, which statements are true?

```
class A{
   int i;
}
class B extends A{
   int j;
}
```

Select 3 options

A. Class B extends class A.

B. Class B is the superclass of class A.

C. Class A inherits from class B.

D. Class B is a subclass of class A.

E. Objects of class B will always have a member variable named i .

Check Answer

64. QID - 2.910

Consider the following code:

```
interface Flyer{ String getName(); }

class Bird implements Flyer{
    public String name;
    public Bird(String name){
        this.name = name;
    }
    public String getName(){ return name; }
}

class Eagle extends Bird {
    public Eagle(String name){
        super(name);
    }
}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Flyer f = new Eagle("American Bald Eagle");
        //PRINT NAME HERE
   }
}
```

Which of the following lines of code will print the name of the Eagle object?

Select 3 options

A. `System.out.println(f.name);`

B. `System.out.println(f.getName());`

C. `System.out.println(((Eagle)f).name);`

D. `System.out.println(((Bird)f).getName());`

E. `System.out.println(Eagle.name);`

F. `System.out.println(Eagle.getName(f));`

Check Answer

65. QID - 2.998

What will the following program print when compiled and run?

```
class Game{
  public void play() throws Exception{
    System.out.println("Playing...");
  }
}

public class Soccer extends Game{
   public void play(){
      System.out.println("Playing Soccer...");
   }
   public static void main(String[] args){
       Game g = new Soccer();
       g.play();
   }
}
```

Select 1 option

A. It will not compile.

B. It will throw an Exception at runtime.

C. Playing Soccer...

D. Playing...

E. None of these.

Check Answer

66. QID - 2.961

Which one of these is a proper definition of a class Car that cannot be sub-classed?

Select 1 option

A. `class Car { }`

B. `abstract class Car { }`

C. `native class Car { }`

D. `static class Car { }`

E. `final class Car { }`

Check Answer

67. QID - 2.1203

A method with no access modifier can be overridden by a method marked protected.

Select 1 option

A. True

B. False

Check Answer

68. QID - 2.1101

Consider the following code:

```
class Base{
 private float f = 1.0f;
 void setF(float f1){ this.f = f1; }
}
class Base2 extends Base{
 private float f = 2.0f;
 //1
}
```

Which of the following options is a valid example of overriding?

Select 2 options

A. `protected void setF(float f1){ this.f = 2*f1; }`

B. `public void setF(double f1){ this.f = (float) 2*f1; }`

C. `public void setF(float f1){ this.f = 2*f1; }`

D. `private void setF(float f1){ this.f = 2*f1; }`

E. `float setF(float f1){ this.f = 2*f1; return f;}`

Check Answer

69. QID - 2.1222

Consider the following class hierarchy

```
class A{
   public void m1() {    }
}
class B extends A{
   public void m1() {    }
}
class C extends B{
   public void m1(){
      /*  //1
      ... lot of code.
      */
   }
}
```

Select 2 options

A. You cannot access class A's `m1()` from class C for the same object (i.e. `this`).

B. You can access class B's `m1()` using `super.m1()` from `class C`.

C. You can access `class A`'s `m1()` using `( (A) this ).m1()` from `class C`.

D. You can access `class A`'s `m1()` using `super.super.m1()` from `class C`.

Check Answer

70. QID - 2.1306

Which of these statements concerning interfaces are true?

Select 2 options

A. An interface may extend an interface.

B. An interface may extend a class and may implement an interface.

C. A class can implement an interface and extend a class.

D. A class can extend an interface and can implement a class.

E. An interface can only be implemented and cannot be extended.

Check Answer

71. QID - 2.1250

Which statements, when inserted at line 1, will cause an exception at run time?

```
class B {}
class B1 extends B {}
class B2 extends B {}
public class ExtendsTest{
  public static void main(String args[]){
     B b = new B();
     B1 b1 = new B1();
     B2 b2 = new B2();
     // insert statement here
  }
}
```

Select 1 option

A. `b = b1;`

B. `b2 = b;`

C. `b1 = (B1) b;`

D. `b2 = (B2) b1;`

E. `b1 = (B) b1;`

Check Answer

72. QID - 2.1168

Consider the following class hierarchy:

```
A
|
B1,  B2
|
C1, C2
```

(B1 and B2 are subclasses of A and C1, C2 are subclasses of B1) Which of the following statements are correct? Assume that objectOfA, objectOfC1, etc. are objects of classes A and C1 respectively.

Select 1 option

A. `objectOfC2 instanceof B2` will return `true`.

B. `objectOfC1 instanceof B1` will return `true`.

C. `objectOfA instanceof B1` will return `true`.

D. C1 c1 = objectOfA; is a valid statement.

E. `B1 b1 = objectOfB2;` is a valid statement.

Check Answer

73. QID - 2.1123

Consider the contents of following two files:

```
//File A.java
package a;
public class A{
   A(){ }
   public void  print(){ System.out.println("A"); }
}

//File B.java
package b;
import a.*;
public class B extends A{
   B(){ }
   public void  print(){ System.out.println("B"); }
   public static void main(String[] args){
      new B();
   }
}
```

What will be printed when you try to compile and run class B?

Select 1 option

A. It will print A.

B. It will print B.

C. It will not compile.

D. It will compile but will not run.

E. None of the above.

Check Answer

74. QID - 2.1149

Given the following class definitions :

```
interface MyIface{};
class A {};
class B extends A implements MyIface{};
class C implements MyIface{};
```

and the following object instantiations:

```
A a = new A();
B b = new B();
C c = new C();
```

Which of the following assignments are legal at compile time?

Select 1 option

A. `b = c;`

B. `c = b;`

C. `MyIface i = c;`

D. `c = (C) b;`

E. `b = a;`

Check Answer

75. QID - 2.1219

What will be the output of compiling and running the following program:

```
class TestClass implements I1, I2{
   public void m1() { System.out.println("Hello"); }
   public static void main(String[] args){
      TestClass tc = new TestClass();
      ( (I1) tc).m1();
   }
}
interface I1{
   int VALUE = 1;
   void m1();
}
interface I2{
   int VALUE = 2;
   void m1();
}
```

Select 1 option

A. It will print `Hello`.

B. There is no way to access any `VALUE` in TestClass.

C. The code will work fine only if `VALUE` is removed from one of the interfaces.

D. It will not compile.

E. None of the above.

Check Answer

76. QID - 2.1220

Given the following classes and declarations, which of these statements about //1 and //2 are true?

```
class A{
   private int i = 10;
   public void  f(){}
   public void g(){}
}

class B extends A{
   public int i = 20;
   public void g(){}
}

public class C{
   A a = new A();//1
   A b = new B();//2
}
```

Select 1 option

A. `System.out.println(b.i);` will print 10.

B. The statement `b.f( );` will give compile time error..

C. `System.out.println(b.i);` will print 20

D. All the above are correct.

E. None of the above statements is correct.

Check Answer

77. QID - 2.1002

Given the following definitions and reference declarations:

```
interface I1 { }
interface I2 { }
class C1 implements I1 { }
class C2 implements I2 { }
class C3 extends C1 implements I2 { }
C1 o1;
C2 o2;
C3 o3;
```

Which of these statements are legal?

Select 3 options

A. `class C4 extends C3 implements I1, I2 { }`

B. `o3 = o1;`

C. `o3 = o2;`

D. `I1 i1 = o3; I2 i2 = (I2) i1;`

E. `I1 b = o3;`

Check Answer

78. QID - 2.1037

What will the following code print when compiled and run?

```
class Base{
   void methodA(){
      System.out.println("base - MethodA");
   }
}

class Sub extends Base{
   public void methodA(){
      System.out.println("sub - MethodA");
   }
   public void methodB(){
      System.out.println("sub - MethodB");
   }
   public static void main(String args[]){
      Base b=new Sub(); //1
      b.methodA(); //2
      b.methodB(); //3
   }
}
```

Select 1 option

A. `sub - MethodA` and `sub - MethodB`

B. `base - MethodA` and `sub - MethodB`

C. Compile time error at `//1`

D. Compile time error at `//2`

E. Compile time error at `//3`

Check Answer

79. QID - 2.1154

Consider the following code:

```
class A{
 public XXX m1(int a){
   return a*10/4-30;
 }
}
class A2 extends A{
 public YYY m1(int a){
   return a*10/4.0;
 }
}
```

What can be substituted for XXX and YYY so that it can compile without any problems?

Select 1 option

A. int, int

B. int, double

C. double, double

D. double, int

E. Nothing, they are simply not compatible.

Check Answer

80. QID - 2.907

Consider the following code appearing in Eagle.java

```
class Bird {
    private Bird(){     }
}
class Eagle extends Bird {
    public String name;
    public Eagle(String name){
        this.name = name;
    }

    public static void main(String[] args) {
        System.out.println(new Eagle("Bald Eagle").name);
    }
}
```

What needs to be done to make this code compile?

Select 1 option

A. Nothing, it will compile as it is.

B. Make Eagle class declaration public:

```
public class Eagle { ... }
```

C. Make the Eagle constructor private:

```
private Eagle(String name){ ... }
```

D. Make Bird constructor public:

```
public Bird() { ... }
```

E. Insert super(); as the first line in Eagle constructor:

```
public Eagle(String name){
    super();
    this.name = name;
}
```

Check Answer

81. QID - 2.1209

Which statements concerning the following code are true?

```
class A{
  public A() {} // A1
  public A(String s) {  this();  System.out.println("A :"+s);  }  //
}

class B extends A{
  public int B(String s) {  System.out.println("B :"+s);  return 0;
}
class C extends B{
    private C(){ super(); } // C1
    public C(String s){  this();  System.out.println("C :"+s);  } //
    public C(int i){} // C3
}
```

Select 4 options

A. At least one of the constructors of each class is called as a result of constructing an object of class C.

B. Constructor at //A2 will never be called in creation of an object of class C

C. Class C can be instantiated only in two ways by users of this class.

D. //B1 will never be called in creation of objects if class A, B, or C

E. The code will not compile.

Check Answer

82. QID - 2.885

Given:

```
//Insert code here

   public abstract void draw();
}


//Insert code here
   public void draw(){  System.out.println("in draw..."); }
}
```

Which of the following lines of code can be used to complete the above code?

Select 2 options

A. class Shape {

and

class Circle extends Shape {

B. public class Shape {

and

class Circle extends Shape {

C. abstract Shape {

and

public class Circle extends Shape {

D. public abstract class Shape {

and

class Circle extends Shape {

E. public abstract class Shape {

and

class Circle implements Shape {

F. public interface Shape {

and

class Circle implements Shape {

Check Answer

Working with Inheritance (Answered)

01. QID - 2.1095 : Working with Inheritance

What will be the result of compiling and running the following code?

```
class Base{
   public short getValue(){ return 1; } //1
}
class Base2 extends Base{
   public byte getValue(){ return 2; } //2
}
public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Correct Option is : D

~~**A.** It will print 1~~

~~**B.** It will print 2.~~

~~**C.** Compile time error at //1~~

D. Compile time error at //2

~~**E.** Compile time error at //3~~

Explanation:

In case of overriding, the return type of the overriding method must match exactly to the return type of the overridden method if the return type is a primitive.
(In case of objects, the return type of the overriding method may be a subclass of the return type of the overridden method.)

Back to Question without Answer

02. QID - 2.1273 : Working with Inheritance

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      A o1 = new C( );
      B o2 = (B) o1;
      System.out.println(o1.m1( ) );
      System.out.println(o2.i );
   }
}
class A { int i = 10;  int m1( ) { return i; } }
class B extends A { int i = 20;  int m1() { return i; } }
class C extends B { int i = 30;  int m1() { return i; } }
```

Correct Option is : C

~~**A.**~~ The program will fail to compile.

~~**B.**~~ Class cast exception at runtime.

C. It will print 30, 20.

~~**D.**~~ It will print 30, 30.

~~**E.**~~ It will print 20, 20.

Explanation:

Remember : variables are SHADOWED and methods are OVERRIDDEN.

Which variable will be used depends on the class that the variable is declared of.
Which method will be used depends on the actual class of the object that is referenced by the variable.
So, in line `o1.m1()`, the actual class of the object is C, so C's `m1()` will be used. So it returns 30.
In line o2.i, o2 is declared to be of class B, so B's i is used. So it returns 20.

Back to Question without Answer

03. QID - 2.1145 : Working with Inheritance

Assume the following declarations:

```
class A{ }
class B extends A{ }
class C extends B{ }

class X{
   B getB(){ return new B(); }
}

class Y extends X{
  // method declaration here
}
```

Which of the following methods can be inserted in class Y?

Correct Options are : B C

A. ~~`public C getB(){ return new B(); }`~~

> Its return type is specified as C, but it is actually returning an object of type B. Since B is NOT a C, this will not compile.

B. `protected B getB(){ return new C(); }`

> Since C is-a B, this is valid. Also, an overriding method can be made less restrictive. protected is less restrictive than 'default' access.

C. `C getB(){ return new C(); }`

> Covariant returns are allowed in Java 1.5, which means that an overriding method can change the return type of the overridden method to a subclass of the

original return type. Here, C is a subclass of B. So this is valid.

D. ~~`A getB(){ return new A(); }`~~

An overriding method cannot return a superclass object of the return type defined in the overridden method. A subclass is allowed in Java 1.5.

Back to Question without Answer

04. QID - 2.1135 : Working with Inheritance

Which of the following statements is/are true?

Correct Option is : C

~~**A.**~~ Subclasses must define all the abstract methods that the superclass defines.

> Not if the subclass is also defined abstract!

~~**B.**~~ A class implementing an interface must define all the methods of that interface.

> Not if the class is defined abstract.

C. A class cannot override the super class's constructor.

> Because constructors are not inherited.

~~**D.**~~ It is possible for two classes to be the superclass of each other.

~~**E.**~~ An interface can implement multiple interfaces.

> Interface cannot "implement" anything. It can extend multiple interfaces. The following is a valid declaration :
> interface I1 extends I2, I3, I4 { }

Back to Question without Answer

05. QID - 2.1337 : Working with Inheritance

Consider the following code:

```
class A{
   A() {  print();   }
   void print() { System.out.println("A"); }
}
class B extends A{
   int i =   Math.round(3.5f);
   public static void main(String[] args){
      A a = new B();
      a.print();
   }
   void print() { System.out.println(i); }
}
```

What will be the output when class B is run ?

Correct Option is : C

~~**A.**~~ It will print A, 4.

~~**B.**~~ It will print A, A

C. It will print 0, 4

~~**D.**~~ It will print 4, 4

~~**E.**~~ None of the above.

Explanation:

Note that method `print()` is overridden in class B. Due to polymorphism, the method to be executed is selected depending on the class of the actual object.
Here, when an object of class B is created, first A's constructor is called, which in turn calls `print()`. Now, since the class of actual object is B, B's `print()` is selected. At this point of time, variable i has not been initialized (because we are still initializing A at this point), so its default value i.e. 0 is printed.
This happens because the method `print()` is non-private, hence polymorphic.

Finally, 4 is printed.

Back to Question without Answer

06. QID - 2.1300 : Working with Inheritance

An abstract method cannot be overridden.

Correct Option is : B

~~**A.**~~ True

B. False

Explanation:

Abstract methods are meant to be overridden in the subclass. Abstract methods describe a behavior but do not implement it. So the subclasses have to override it to actually implement the behavior. A subclass may chose not to override it, in which case, the subclass will have to be abstract too.

Back to Question without Answer

07. QID - 2.836 : Working with Inheritance

Which of the following statements are correct?

Correct Options are : A B C

A. An abstract class can be extended by an abstract or a concrete class.

B. A concrete class can be extended by an abstract or a concrete class.

C. An interface can be extended by another interface.

~~**D.**~~ An interface can be extended by an abstract class.

A class "implements" an interface. It does not "extend" an interface.

~~**E.**~~ An interface can be extended by a concrete class.

~~**F.**~~ An abstract class cannot implement an interface.

Any class, whether abstract or concrete, can implement any interface.

Back to Question without Answer

08. QID - 2.1181 : Working with Inheritance

What would be the result of attempting to compile and run the following code?

```
// Filename: TestClass.java
public class TestClass{
   public static void main(String args[]){
      B c = new C();
      System.out.println(c.max(10, 20));
   }
}
class A{
   int max(int x, int y)  { if (x>y) return x; else return y; }
}
class B extends A{
  int max(int x, int y)  {  return 2 * super.max(x, y) ; }
}
class C extends B{
  int max(int x, int y)  {  return super.max( 2*x, 2*y); }
}
```

Correct Option is : C

~~**A.**~~ The code will fail to compile.

~~**B.**~~ Runtime error.

C. The code will compile without errors and will print 80 when run.

~~**D.**~~ The code will compile without errors and will print 40 when run.

~~**E.**~~ The code will compile without errors and will print 20 when run.

Explanation:

When the program is run, the `main()` method will call the `max()` method in C with parameters 10 and 20 because the actual object referenced by 'c' is of class C. This method will call the `max()` method in B with the parameters 20 and 40. The `max()` method in B will in turn call the `max()` method in A with the parameters 20 and 40. The `max()` method in A will return 40 to the `max()` method in B. The `max()` method in B will return 80 to the `max()` method in C. And finally the `max()` of C will return 80 in `main()` which will be printed out.

Back to Question without Answer

09. QID - 2.1086 : Working with Inheritance

What will be the result of compiling and running the following code?

```
class Base{
   public Object getValue(){ return new Object(); } //1
}

class Base2 extends Base{
   public String getValue(){ return "hello"; } //2
}

public class TestClass{
   public static void main(String[] args){
      Base b = new Base2();
      System.out.println(b.getValue()); //3
   }
}
```

Correct Option is : B

~~**A.**~~ It will print the hash code of the object.

B. It will print `hello`.

> Covariant returns are allowed since Java 1.5, which means that an overriding method can change the return type to a subclass of the return type declared in the overridden method. But remember than covarient returns does not apply to primitives.

~~**C.**~~ Compile time error at `//1`.

~~**D.**~~ Compile time error at `//2`.

~~**E.**~~ Compile time error at `//3`.

Explanation:

Observe that at run time `b` points to an object of class `Base2`. Further, `Base2` overrides `getValue()`. Therefore, `Base2`'s `getValue()` will be invoked and it will return `hello`.

Back to Question without Answer

10. QID - 2.1143 : Working with Inheritance

Which of the given statements are correct for a method that overrides the following method:

```
public Set getSet(int a) {...}
```

Correct Options are : B D E

~~**A.**~~ Its return type must be declared as `Set`.

> Return type may also be a subclass/subinterface. So it can also return `SortedSet`, `TreeSet`, `HashSet`, or any other class that implements or subclasses a `Set`.

B. It may return HashSet.

~~**C.**~~ It can declare any Exception in throws clause

> Since the original (overridden) method does not have any throws clause, the overriding method cannot declare any checked exceptions.

D. It can declare any RuntimeException in throws clause.

> A method can throw any RuntimeException (such as a NullPointerException) even without declaring it in its throws clause.

E. It can be abstract.

> Yes, you can make it abstract!! You would have to make the class as abstract as well though.

Explanation:

To override a method in the subclass, the overriding method (i.e. the one in the subclass) MUST HAVE:
.same name
.same return type in case of primitives (a subclass is allowed for classes, this is also known as covariant return types).
.same type and order of parameters
.It may throw only those exceptions that are declared in the throws clause of the superclass's method or exceptions that are subclasses of the declared exceptions. It may also choose NOT to throw any exception.
The names of the parameter types do not matter. For example, void `methodX(int i)` is same as void `methodX(int k)`

Back to Question without Answer

11. QID - 2.1256 : Working with Inheritance

Which of these statements are true?

Correct Options are : B E

~~**A.**~~ A `super( )` or `this( )` call must always be provided explicitly as the first statement in the body of the constructor.

> `super()` is automatically added if the sub class constructor doesn't call any of the super class's constructor.

B. If a subclass does not have any declared constructors, the implicit default constructor of the subclass will have a call to `super( )`.

~~**C.**~~ If neither `super( )` or `this( )` is declared as the first statement of the body of a constructor, then `this( )` will implicitly be inserted as the first statement.

> `super()` is added and not `this()`

~~**D.**~~ `super(...)` can only be called in the first line of the constructor but `this(...)` can be called from anywhere.

E. You can either call `super(...)` or `this(...)` but not both.

Explanation:

Note that calling `super()` will not always work because if the super class has defined a constructor with arguments and has not defined a no args constructor then no args constructor will not be provided by the compiler. It is provided only to the class that does not define ANY constructor explicitly.

Back to Question without Answer

12. QID - 2.1231 : Working with Inheritance

Given the following interface definition, which definitions are valid?

```
interface I1{
   void setValue(String s);
   String getValue();
}
```

Correct Options are : B C

~~A.~~
```
class A extends I1{
   String s;
   void setValue(String val) { s = val; }
   String getValue() { return s; }
}
```

> Classes do not `extend` interfaces, they implement interfaces.

B.
```
interface I2 extends I1{
   void analyse();
}
```

C.
```
abstract class B implements I1{
   int getValue(int i) { return 0; }
}
```

~~D.~~
```
interface I3 implements I1{
   void perform_work();
}
```

> Interfaces do not `implement` anything, they can extend multiple interfaces.

Explanation:

The `getValue(int i)` method of class B in option c, is different than the method defined in the interface because their parameters are different. Therefore, this class does not actually implement the method of the interface and that is why it needs to be declared abstract.

Back to Question without Answer

13. QID - 2.1328 : Working with Inheritance

Consider the following classes :

```
interface I{
}
class A implements I{
}

class B extends A {
}

class C extends B{
}
```

And the following declarations:

```
A a = new A();
B b = new B();
```

Identify options that will compile and run without error.

Correct Option is : A

A. `a = (B)(I)b;`

> class B does implement I because it extends A, which implements I. A reference of type I can be cast to any class at compile time. Since B is-a A, it can be assigned to a.

B. `b = (B)(I) a;`

> This will fail at run time because a does not point to an object of class B.

C. `a = (I) b;`

An I is not an A. Therefore, it will not compile.

D. `I i = (C) a;`

It will compile because a C is-a A, which is-a I, and a reference of class A can point to an object of class C. But it will fail at runtime because a does not point to an object of class C.

Back to Question without Answer

14. QID - 2.905 : Working with Inheritance

Given the following line of code:

```
List students = new ArrayList();
```

Identify the correct statement:

Correct Option is : A

A. The reference type is List and the object type is ArrayList.

> Since you are doing new ArrayList, you are creating an object of class ArrayList. You are assigning this object to variable "students", which is declared of class List. Reference type means the declared type of the variable.

~~**B.**~~ The reference type is ArrayList and the object type is ArrayList.

~~**C.**~~ The reference type is ArrayList and the object type is List.

~~**D.**~~ The reference type is List and the object type is List.

Back to Question without Answer

15. QID - 2.945 : Working with Inheritance

Which of the following statements are true?

Correct Options are : A E

A. Private methods cannot be overridden in subclasses.

> Only methods that are inherited can be overridden and private methods are not inherited.

~~**B.**~~ A subclass can override any method in a non-final superclass.

> Only the methods that are not declared to be final can be overridden. Further, private methods are not inherited so they cannot be overridden either.

~~**C.**~~ An overriding method can declare that it throws a wider spectrum of checked exceptions than the method it is overriding.

~~**D.**~~ The parameter list of an overriding method must be a subset of the parameter list of the method that it is overriding.

> An overriding method (the method that is trying to override the base class's method) must have the same parameters.

E. The overriding method may opt not to declare any throws clause even if the original method has a throws clause.

> No exception (i.e. an empty set of exceptions) is a valid subset of the set of exceptions thrown by the original method so an overriding method can choose to not have any throws clause.

Explanation:

A method can be overridden by defining a method with the same signature(i.e. name and parameter list) and return type as the method in a superclass. The return type can also be a subclass of the orginal method's return type.

Only methods that are accessible can be overridden. A private method cannot, therefore, be overridden in subclasses, but the subclasses are allowed to define a new method with exactly the same signature.

A final method cannot be overridden.

An overriding method cannot exhibit behavior that contradicts the declaration of the original method. An overriding method therefore cannot return a different type (except a subtype) or throw a wider spectrum of exceptions than the original method in the superclass.

A subclass may have a static method with the same signature as a static method in the base class but it is not called overriding. It is called shadowing because the concept of polymorphism doesn't apply to static members.

Back to Question without Answer

16. QID - 2.1136 : Working with Inheritance

Which of the given statements are correct about the following code?

```
//Filename: TestClass.java
class TestClass{
   public static void main(String[] args){
      A a = new A();
      B b = new B();
   };
}
class A implements T1, T2{}
class B extends A implements T1{}
interface T1 { }
interface T2 { }
```

Correct Options are : A B C D

A. `(a instanceof T1)` will return true.

B. `(a instanceof T2)` will return true.

C. `(b instanceof T1)` will return true.

D. `(b instanceof T2)` will return true.

~~**E.**~~ `(b instanceof A)` will return false.

It will return true because B extends A and 'b' is referring to an object of class B.

Explanation:

Since A implements both T1 and T2, 1 and 2 are correct.
b instanceof A will return true as B is a subclass of A. Note that it is 'A' and not 'a'.
`( b instanceof a )` will not compile.

Back to Question without Answer

17. QID - 2.1107 : Working with Inheritance

Which of the following are correct about "encapsulation"?

Correct Options are : B C

~~**A.**~~ Encapsulation is same as polymorphism.

B. It helps make sure that clients have no accidental dependence on the choice of representation

C. It helps avoiding name clashes as internal variables are not visible outside.

~~**D.**~~ Encapsulation makes sure that messages are sent to the right object at run time.

This is dynamic binding, an outcome of polymorphism.

~~**E.**~~ Encapsulation helps you inherit the properties of another class.

Explanation:

Encapsulation: Encapsulation is the technique used to package the information in such a way as to hide what should be hidden, and make visible what is intended to be visible. In simple terms, encapsulation generally means making the data variables private and providing public accessors.

Back to Question without Answer

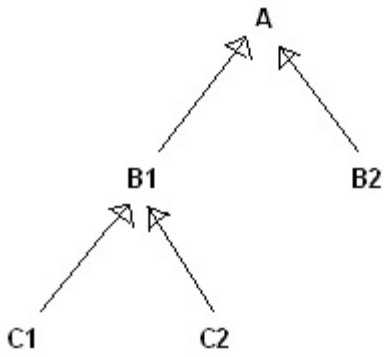
18. QID - 2.1287 : Working with Inheritance

Consider the following class hierarchy shown in the image. (B1 and B2 are subclasses of A and C1, C2 are subclasses of B1)

Assume that method `public void m1(){ ... }` is defined in all of these classes EXCEPT B1 and C1.

Assume that `"objectOfXX"` means a variable that points to an object of class XX. So, `objectOfC1` means a reference variable that is pointing to an object of class C1.

Which of the following statements are correct?



Correct Option is : C

~~**A.**~~ `objectOfC1.m1();` will cause a compilation error.

> C1 will inherit B1's `m1()` which in turn inherits `m1()` from A.

~~**B.**~~ `objectOfC2.m1();` will cause A's `m1()` to be called.

> C2 has `m1()`, so its `m1()` will override A's `m1()`.

C. `objectOfC1.m1();` will cause A's `m1()` to be called.

> C1 will inherit B1's `m1()` which in turn inherits `m1()` from A.

~~**D.**~~ `objectOfB1.m1();` will cause an exception at runtime.

> B1 will inherit `m1()` from A. So this is valid.

~~**E.**~~ `objectOfB2.m1();` will cause an exception at runtime.

> B2 overrides `m1()` of A. So there will be no exception.

Back to Question without Answer

19. QID - 2.931 : Working with Inheritance

Consider the following classes :

```
class A{
    public static void sM1() {  System.out.println("In base static"),
}
class B extends A{
Line 1 :   // public static void sM1() {  System.out.println("In sub
Line 2 :   // public  void sM1() {  System.out.println("In sub non-s
}
```

Which of the following statements are true?

Correct Options are : B C

~~**A.**~~ class B will not compile if line 1 is uncommented.

> static method sM1() can be shadowed by a static method sM1() in the subclass.

B. class B will not compile if line 2 is uncommented.

> static method cannot be overridden by a non-static method and vice versa

C. class B will not compile if line 1 and 2 are both uncommented.

~~**D.**~~ Only Option 2 is correct.

~~**E.**~~ Only Option 3 is correct.

Explanation:

Another concept (although not related to this question but about static methods) is that static methods are never overridden. They are HIDDEN or SHADOWED just like static or non-static fields. For example,

```
class A{
  int i = 10;
  public static void m1(){  }
  public void m2() { }
}
class B extends A{
  int i = 20;
  public static void m1() {  }
  public void m2() { }
}
```

Here, UNLIKE m2, m1() of B does not override m1() of A, it just shadows it, as proven by the following code:

```
A a  = new B();
System.out.println(a.i)  //will print 10 instead of 20
a.m1();  //will call A's m1
a.m2();  //will call B's m2 as m2() is not static and so overrides A
```

Back to Question without Answer

20. QID - 2.936 : Working with Inheritance

Consider the following code:

```
class Super{
  static{ System.out.print("super "); }
}
class One{
  static {  System.out.print("one "); }
}
class Two extends Super{
  static {  System.out.print("two "); }
}
class Test{
  public static void main(String[] args){
     One o = null;
     Two t = new Two();
  }
}
```

What will be the output when class Test is run ?

Correct Option is : C

~~**A.**~~ It will print `one, super` and `two`.

> "`one`" will not be printed as class One is not actively used.

~~**B.**~~ It will print `one, two` and `super`.

C. It will print `super` and `two`.

> `Super` will be instantiated before `Two`.

~~**D.**~~ It will print `two` and `super`

~~**E.**~~ None of the above.

Explanation:

As per JLS 12.4.1 - A class or interface type T will be initialized immediately before the first occurrence of any one of the following:

T is a class and an instance of T is created.
T is a class and a static method declared by T is invoked.
A static field declared by T is assigned.
A static field declared by T is used and the field is not a constant variable.
T is a top-level class, and an assert statement lexically nested within T is executed.

The statement `One o = null;` does not fall in either of the cases mentioned above. So class One is not initialized and its static block is not executed. Class Two is initialized only after its superclass Super has been initialized.

Back to Question without Answer

21. QID - 2.872 : Working with Inheritance

Consider the following classes:

```
class A {
      public int getCode(){ return 2;}
}

class AA extends A {
  public void doStuff() {
  }
}
```

Given the following two declarations, which of the options will compile?

```
A a = null;
AA aa = null;
```

Correct Options are : A B D F

A. `a = (AA)aa;`

B. `a = new AA();`

C. ~~`aa = new A();`~~

D. `aa = (AA) a;`

`a` is declared as a reference of class A and therefore, at run time, it is possible for `a` to point to an object of class AA (because A is a super class of AA). Hence, the compiler will not complain. Although if `a` does not point to an object of class AA at run time, a `ClassCastException` will be thrown.

E. ~~`aa = a;`~~

A cast is required because the compiler needs to be assured that at run time `a` will point to an object of class AA.

F. `((AA)a).doStuff();`

Once you cast `a` to AA, you can call methods defined in AA. Of course, if `a` does not point to an object of class AA at runtime, a `ClassCastException` will be thrown.

Back to Question without Answer

22. QID - 2.1292 : Working with Inheritance

Given the definitions of I and Klass, complete the definition of SubClass so that it extends from Klass and implements I.
Use minimum number of elements.

```
interface I
{
    void m1();
}

abstract class Klass
{
    void m1() { };
}

class SubClass extends Klass implements I
{
    public void m1(){ }
}
```

Explanation:

Even though class Klass implements `m1()`, it does not declare that it implements I. Therefore, for a subclass to 'implement' I, it must have 'implements I' in its declaration.
Further, `m1()` in Klass is not public. So even though Subclass inherits m1() from Klass, it will not be a valid implementation of I because interface methods must be public. Therefore, SubClass must override `m1()` and make it public.

Back to Question without Answer

23. QID - 2.1064 : Working with Inheritance

Consider this code:

```
interface X1{ }
interface X2{ }
class A { }
class B extends A implements X1{ }
class C extends B implements X2{
   D d = new D();
}
class D { }
```

Which of the following statements are true?

Correct Options are : C D E

~~**A.**~~ D is-a B.

~~**B.**~~ B has-a D.

C has-a D.

C. C is-a A

Because C 'is-a' B and B 'is-a' A.

D. C is-a X1

Because C is-a B and B is-a X1.

E. C is-a X2

Because C implements X2

Explanation:

Consider this code:

```
class A extends B implements I{
  D d = new D();
}
```

Now, Inheritance defines an is-a relation , so A is-a B because A extends B. This actually means that A can be used in all the places where B is used. A can substitute for B anywhere because A is-a B. As all objects have Object as their super class, every object 'is-a' Object.

Since A implements I, it is sometimes said that A 'is-like-a' I. That is, although A is not a I but for all purposes A is like an I. The distinction between is-a and is-like-a is not important for the exam. For the purpose of the exam, is-like-a is same as is-a.

Aggregation defines a has-a relation. Here, D is a member object in A so D is contained in A. So it is said that A 'has-a' D.

All Java objects have the class Object as the ultimate superclass, and so Object is always at the root of any inheritance hierarchy.

Back to Question without Answer

24. QID - 2.1015 : Working with Inheritance

Which statement regarding the following code is correct?

```
class A{
   public int i = 10;
   private int j = 20;

}

class B extends A{
   private int i = 30; //1
   public int k = 40;

}

class C extends B{
}

public class TestClass{
   public static void main(String args[]){
      C c = new C();
      System.out.println(c.i); //2
      System.out.println(c.j); //3
      System.out.println(c.k);
   }
}
```

Correct Option is : B

~~**A.**~~ It will print `10` and `40` if //3 is commented.

B. It will print `40` if //2 and //3 are commented.

~~**C.**~~ It will not compile because of `//1`.

~~**D.**~~ It will compile if `//2` is commented.

~~**E.**~~ None of these.

Explanation:

You cannot access c.i because i is private in B. But you can access `( (A)c).i` because `i` is public in A. Remember that member variables are shadowed and not overridden. So, B's i shadows A's i and since B's i is private, you can't access A's i unless you cast the reference to A.
You cannot access `c.j` because `j` is private in A.

Back to Question without Answer

25. QID - 2.1208 : Working with Inheritance

Given the following class definition:

```
class A{
  protected int i;
  A(int i) {    this.i = i;    }

}
// 1 : Insert code here
```

Which of the following would be a valid class that can be inserted at //1 ?

Correct Options are : A D

A. `class B {}`

~~**B.**~~ `class B extends A {}`

Since class B does not have any constructor, the compiler will try to insert the default constructor, which will look like this:

```
B(){
    super();  //Notice that it is trying to call the no args
constructor of the super class, A.
}
```

Since A doesn't have any no-args constructor, the above code will fail to compile.

~~**C.**~~ `class B extends A {  B()  {  System.out.println("i = " + i); }  }`

It has the same problem as the one above.

D. `class B { B() {} }`

Explanation:

Notice that class A does not define a no-argument constructor. Also note that the class B does not define a constructor. Thus, class B relies on the default constructor B(). Class B's default constructor looks like this: public B() {} However, Constructors implicitly (if an explicit call to the superclass's constructor is not present) call their superclass's constructor super(). So, class B's default constructor actually looks like this:

```
public B(){
  super();
}
```

Now, since class A does not define a no-argument constructor the above code will not compile. However, class B would be correct if changed to:

```
class B extends A{
  B(){
    super(1); // pass it any integer
  }
  // or
  B(int number){
    super(number);
  }
}
```

You could also add a no-argument constructor to class A and leave class B as is.

Back to Question without Answer

26. QID - 2.1091 : Working with Inheritance

What will the following code print when TestClass is run?

```
class Employee {

  static int i = 10;  {
      i = 15;
      System.out.print(" Employee "+i);
  }
  static { System.out.print(" Employee static "+i); }
}

class Manager extends Employee {
  static {
    i = 45;
    System.out.print(" Manager static ");
  }{
    i = 30;
    System.out.print(" Manager "+i);
  }
}

class Owner extends Manager{
  static { System.out.println("Owner"); }
}

public class TestClass {
  public static void main(String[] args) {
      Manager m = new Manager();
  }
}
```

Correct Option is : C

~~**A.**~~ Employee static 10 Manager static Employee 10 Manager 30

B. ~~Employee static 10 Manager static Owner Manager 30~~

> Since Owner is not at all referred any where, its static block will not be executed.

C. Employee static 10 Manager static Employee 15 Manager 30

D. ~~Employee static 10 Manager static 15 Manager 20~~

E. ~~It will not compile.~~

Explanation:

Although there is more to it that the following sequence, for the purpose of exam, this is all you need to know:

1. Static blocks of the base class (only once, in the order they appear in the class).

2. Static blocks of the class.

3. Non-static blocks of the base class.

4. Constructor of the base class.

5. Non-static blocks of the class.

6. Constructor of the class.

7. Derived class's static or non-static blocks are not executed if that class is not being used.
(For example, in this question class Owner is not being used.)

Back to Question without Answer

27. QID - 2.1320 : Working with Inheritance

What will the following code print when compiled and run?

```
class ABCD{
   int x = 10;
   static int y = 20;
}
class MNOP extends ABCD{
   int x = 30;
   static int y = 40;
}

public class TestClass {
   public static void main(String[] args) {
     System.out.println(new MNOP().x+", "+new MNOP().y);
   }
}
```

Correct Option is : D

~~**A.**~~ 10, 40

~~**B.**~~ 30, 20

~~**C.**~~ 10, 20

D. 30, 40

~~**E.**~~ 20, 30

~~**F.**~~ Compilation error.

Explanation:

Access to static and instance fields and static methods depends on the class of reference variable and not the actual object to which the variable points to. Observe that this is opposite of what happens in the case of instance methods. In case of instance methods the method of the actual class of the object is called.

Therefore, in case of `System.out.println(new MNOP().x);` the reference is of type MNOP and so MNOP's x will be accessed.

Had it been like this:

```
ABCD a = new MNOP();
System.out.println(a.x);
System.out.println(a.y);
```

ABCD's x and y would have been accessed because a is of type ABCD even though the actual object is of type MNOP.

Back to Question without Answer

28. QID - 2.970 : Working with Inheritance

Consider the following classes...

```
class Car{
   public int gearRatio = 8;
   public String accelerate() {  return "Accelerate : Car";  }
}
class SportsCar extends Car{
   public int gearRatio = 9;
   public String accelerate() {  return  "Accelerate : SportsCar";
   public static void main(String[] args){
      Car c = new SportsCar();
      System.out.println( c.gearRatio+"  "+c.accelerate() );
   }
}
```

What will be printed when SportsCar is run?

Correct Option is : C

~~**A.**~~ 8 Accelerate : Car

~~**B.**~~ 9 Accelerate : Car

C. 8 Accelerate : SportsCar

~~**D.**~~ 9 Accelerate : SportsCar

~~**E.**~~ None of the above.

Explanation:

The concept is : variables are shadowed and methods are overridden.
Method to be executed depends on the class of the actual object the variable is referencing to. Here, c refers to object of class SportsCar so SportsCar's accelerate() is selected.

Back to Question without Answer

29. QID - 2.1171 : Working with Inheritance

Consider the following variable declaration within the definition of an interface:

```
int i = 10;
```

Which of the following declarations defined in a non-abstract class, is equivalent to the above?

Correct Option is : C

~~**A.**~~ `public static int i = 10;`

~~**B.**~~ `public final int i = 10;`

C. `public static final int i = 10;`

~~**D.**~~ `public int i = 10;`

~~**E.**~~ `final int i = 10;`

Explanation:

Fields in an interface are implicitly `public`, `static` and `final`. Although you can put these words in the interface definition but it is not a good practice to do so.

Back to Question without Answer

30. QID - 2.1049 : Working with Inheritance

Consider that you are writing a set of classes related to a new Data Transmission Protocol and have created your own exception hierarchy derived from java.lang.Exception as follows:

```
enthu.trans.ChannelException
              +-- enthu.trans.DataFloodingException,
                    enthu.trans.FrameCollisionException
```

You have a TransSocket class that has the following method:

```
long connect(String ipAddr) throws ChannelException
```

Now, you also want to write another "AdvancedTransSocket" class, derived from "TransSocket" which overrides the above mentioned method. Which of the following are valid declaration of the overriding method?

Correct Options are : C E

~~**A.**~~ `int connect(String ipAddr) throws DataFloodingException`

> The return type must match. Otherwise the method is OK.

~~**B.**~~ `int connect(String ipAddr) throws ChannelException`

> The return type must match. Otherwise the method is OK.

C. `long connect(String ipAddr) throws FrameCollisionException`

~~**D.**~~ `long connect(String ipAddr) throws Exception`

This option is invalid because `Exception` is a super class of `ChannelException` so it cannot be thrown by the overriding method.

E. `long connect(String str)`

Explanation:

There are 2 important concepts involved here:
1. The overriding method must have same return type in case of primitives (a subclass is allowed in case of classes) (Therefore, the choices returning int are not valid.) and the parameter list must be the same (The name of the parameter does not matter, just the Type is important).

2. The overriding method can throw a subset of the exception or subclass of the exceptions thrown by the overridden class. Having no throws clause is also valid since an empty set is a valid subset.

Back to Question without Answer

31. QID - 2.1003 : Working with Inheritance

Given the following code, which statements are true?

```
interface Automobile { String describe(); }

class FourWheeler implements Automobile{
   String name;
   public String describe(){ return " 4 Wheeler " + name; }
}

class TwoWheeler extends FourWheeler{
    String name;
    public String describe(){ return " 2 Wheeler " + name; }
}
```

Correct Options are : A B C

A. An instance of `TwoWheeler` is also an instance of `FourWheeler`.

B. An instance of `TwoWheeler` is a valid instance of `Automobile`.

C. The use of inheritance is not justified here because a `TwoWheeler` is not really a `FourWheeler`.

~~**D.** The code will compile only if `name` is removed from `TwoWheeler`.~~

~~**E.** The code will fail to compile.~~

Explanation:

The use of inheritance in this code is not justifiable, since conceptually, a `TwoWheeler` is-not-a `FourWheeler`.

Back to Question without Answer

32. QID - 2.993 : Working with Inheritance

Which is the first line that will cause compilation to fail in the following program?

```
// Filename: A.java
class A{
   public static void main(String args[]){
      A a = new A();
      B b = new B();
      a = b;  // 1
      b = a;  // 2
      a = (B) b; // 3
      b = (B) a; // 4
   }
}
class B extends A { }
```

Correct Option is : B

~~**A.**~~ At Line 1.

B. At Line 2.

> Because 'a' is declared of class A and 'b' is of B which is a subclass of A. So an explicit cast is needed.

~~**C.**~~ At Line 3.

~~**D.**~~ At Line 4.

~~**E.**~~ None of the above.

Explanation:

Casting a base class to a subclass as in : `b = (B) a;` is also called as narrowing (as you are trying to narrow the base class object to a more specific class object) and needs explicit cast.
Casting a sub class to a base class as in: `A a = b;` is also called as widening and does not need any casting.

For example, consider two classes: Automobile and Car, where Car extends Automobile
Now, `Automobile a = new Car();` is valid because a car is definitely an Automobile. So it does not need an explicit cast.

But, Car c = a; is not valid because 'a' is an Automobile and it may be a Car, a Truck, or a MotorCycle, so the programmer has to explicitly let the compiler know that at runtime 'a' will point to an object of class Car. Therefore, the programmer must use an explicit cast:

```
Car c = (Car) a;
```

Back to Question without Answer

33. QID - 2.1035 : Working with Inheritance

Which of the following statements are true?

Correct Options are : A C

A. The `extends` keyword is used to specify inheritance.

~~**B.**~~ subclass of a non-abstract class cannot be declared `abstract`.

C. subclass of an abstract class can be declared `abstract`.

~~**D.**~~ subclass of a final class cannot be `abstract`.

> final class cannot be subclassed.

~~**E.**~~ A class, in which all the members are declared `private`, cannot be declared `public`.

> There is no such rule.

Explanation:

The `extends` clause is used to specify that a class extends another class and thereby inherits all non-private instance members of that class.

A subclass can be declared `abstract` regardless of whether the superclass was declared abstract. A class cannot be declared `abstract` and `final` at the same time. This restriction makes sense because abstract classes need to be subclassed to be useful and final forbids subclasses.

The visibility of the class is not limited by the visibility of its members. A class with all the members declared private can still be declared public or a class having all public members may be declared private.

Back to Question without Answer

34. QID - 2.1009 : Working with Inheritance

Consider the following code:

```
class Super { static String ID = "QBANK"; }

class Sub extends Super{
   static { System.out.print("In Sub"); }
}
public class Test{
   public static void main(String[] args){
      System.out.println(Sub.ID);
   }
}
```

What will be the output when class Test is run?

Correct Option is : B

~~**A.**~~ It will print `In Sub` and `QBANK`.

B. It will print `QBANK`.

~~**C.**~~ Depends on the implementation of JVM.

~~**D.**~~ It will not even compile.

~~**E.**~~ None of the above.

Explanation:

A class or interface type T will be initialized at its first active use, which occurs if:

T is a class and a method actually declared in T (rather than inherited from a superclass) is invoked.

T is a class and a constructor for class T is invoked, or U is an array with element type T, and an array of type U is created.

A non-constant field declared in T (rather than inherited from a superclass or superinterface) is used or assigned. A constant field is one that is (explicitly or implicitly) both final and static, and that is initialized with the value of a compile-time constant expression . Java specifies that a reference to a constant field must be resolved at compile time to a copy of the compile-time constant value, so uses of such a field are never active uses.

All other uses of a type are passive

A reference to a field is an active use of only the class or interface that actually declares it, even though it might be referred to through the name of a subclass, a subinterface, or a class that implements an interface.

Back to Question without Answer

35. QID - 2.1215 : Working with Inheritance

Consider the following classes :

```
class A{
   public void mA(){ };
}

class B extends A {
   public void mA(){ }
   public void mB() { }
}

class C extends B {
   public void mC(){ }
}
```

and the following declarations:

```
A x = new B(); B y = new B(); B z = new C();
```

Which of the following calls are polymorphic calls?

Correct Options are : A C E

A. `x.mA();`

~~B.~~ `x.mB();`

C. `y.mA();`

~~D.~~ `z.mC();`

E. `z.mB();`

Explanation:

A Polymorphic call means that irrespective of the type of the variable, the method of the actual class of the object referred by the variable is invoked. In java, all non-private method calls are polymorphic because which method is invoked is decided at runtime based on the class of the object instead of compile time.

In this case, `x.mB()` is invalid call. It will not even compile because the class of x is A, which does not contain method `mB()`. Even though the object referred to by x is of class B which does contain `mB()`. `z.mC()` is invalid for the same reason.

Back to Question without Answer

36. QID - 2.1329 : Working with Inheritance

Which of the following method definitions will prevent overriding of that method?

Correct Options are : A B C E

A. `public final void m1()`

> final methods cannot be overridden. That is the purpose of final keyword.

B. `public static void m1()`

C. `public static final void m1()`

> Keep in mind that static methods are not overridden, they are shadowed.

D. ~~`public abstract void m1()`~~

E. `private void m1()`

> private methods are not inherited at all so there is no question of overriding a private method.

Back to Question without Answer

37. QID - 2.838 : Working with Inheritance

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Flyer) System.out.println("f is a Flyer");
        if(e instanceof Bird) System.out.println("e is a Bird");
        if(b instanceof Bird) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Correct Option is : A

A. It will not compile.

b points to an object of class Bat, which does not extend from Bird. Now, it is possible for b to point to an object of any subclass of Bat. However, it is not possible for that sub class to extend Bird (because a class can at most extend from only one class). Therefore, it is not possible for b to point to an object of a class that extends Bird. The compiler figures out this fact at compile time itself and so the code fails to compile.

~~**B.**~~ It will throw an exception when run.

~~**C.**~~ f is a Flyer
e is a Bird

> At run time, f points to an object of class Eagle. Now, Eagle extends Bird, which implements Flyer so f instanceof Flyer returns true.
>
> e points to an object of class Eagle. Eagle extends Bird. Therefore, e instanceof Bird will also return true.

~~**D.**~~ f is a Flyer

~~**E.**~~ e is a Bird

Explanation:

Note that there is no compilation issue with b instanceof Flyer because Flyer is an interface and it is possible for b to point to an object of a class that is a sub class of Bat and also implements Flyer. So the compiler doesn't complain. If you make Bat class as final, b instanceof Flyer will not compile because the compiler knows that it is not possible for b to point to an object of a class that implements Flyer.

Back to Question without Answer

38. QID - 2.1150 : Working with Inheritance

Which of the following are valid declarations inside an interface?

Correct Options are : A B

A. `void compute();`

> All interface methods have to be public. No access control keyword in the method declaration also means public in an interface. (Note that the absence of access control keyword in the method declaration in a class means package protected.)

B. `public void compute();`

~~C.~~ `public final void compute();`

> final is not allowed.

~~D.~~ `static void compute();`

> static is not allowed.

~~E.~~ `protected void compute();`

> All interface methods have to be public.

Back to Question without Answer

39. QID - 2.952 : Working with Inheritance

Which of the following lines of code that, when inserted at line 1, will make the overriding method in SubClass invoke the overridden method in BaseClass on the current object with the same parameter.

```
class BaseClass{
   public void print(String s) {  System.out.println("BaseClass :"+s)
}
class SubClass extends BaseClass{
   public void print(String s){
      System.out.println("SubClass :"+s);
      // Line 1
   }
   public static void main(String args[]){
      SubClass sc = new SubClass();
      sc.print("location");
   }
}
```

Correct Option is : B

A. ~~`this.print(s);`~~

B. `super.print(s);`

This is the right syntax to call the base class's overridden method. However, note that there is no way call a method if it has been overriden more than once. For example, if you make BaseClass extend from another base class SubBase, and if SubBase also has the same method, then there is no way to invoke SubBase's print method from SubClass's print method. You cannot have something like super.super.print(s);

~~C.~~ `print(s);`

This will call the same method and will cause a recursion.

~~D.~~ `BaseClass.print(s);`

print is not a static method.

Back to Question without Answer

40. QID - 2.1117 : Working with Inheritance

Given the following classes, what will be the output of compiling and running the class Truck?

```
class Automobile{
   public void drive() {  System.out.println("Automobile: drive");
}

public class Truck extends Automobile{
   public void drive() {  System.out.println("Truck: drive");   }
   public static void main (String args [ ]){
      Automobile  a = new Automobile();
      Truck t  = new Truck();
      a.drive(); //1
      t.drive(); //2
      a = t;     //3
      a.drive(); //4
   }
}
```

Correct Option is : D

~~**A.**~~ Compiler error at line 3.

~~**B.**~~ Runtime error at line 3.

~~**C.**~~ It will print:

```
Automobile: drive
Truck: drive
Automobile: drive
```

in that order.

D. It will print:

```
Automobile: drive
Truck: drive
Truck: drive
```

in that order.

~~**E.**~~ It will print:

```
Automobile: drive
Automobile: drive
Automobile: drive
```

in that order.

Explanation:

Since `Truck` is a subclass of `Automobile`, `a = t` will be valid at compile time as well runtime. But a cast is needed to make for `t = (Truck) a;` This will be ok at compile time but if at run time 'a' does not refer to an object of class Truck then a `ClassCastException` will be thrown. Now, method to be executed is decided at run time and it depends on the actual class of object referred to by the variable. Here, at line 4, variable a refers to an object of class Truck. So Truck's `drive()` will be called which prints `Truck: drive`. This is polymorphism in action!

Back to Question without Answer

41. QID - 2.995 : Working with Inheritance

Which of these statements about interfaces are true?

Correct Options are : A C E

A. Interfaces are abstract by default.

Because they don't have any implementation and so can't be instantiated.

~~**B.**~~ An interface can have static methods.

C. All methods in an interface are abstract although you need not declare them to be so.

~~**D.**~~ Fields of an interface may be declared as transient or volatile but not synchronized.

E. interfaces cannot be final.

Explanation:

Here are the rules:
1. Every interface is implicitly `abstract`. This modifier is obsolete for interfaces and should not be used in new Java programs.
2. An interface can extend any number of other interfaces and can be extended by any number of other interfaces
3. Every field declaration in the body of an interface is implicitly `public`, `static` and `final`. It is permitted, but strongly discouraged as a matter of style, to redundantly specify any or all of these modifiers for such fields. A constant declaration in an interface must not include any of the modifiers `synchronized`, `transient` or

`volatile`, or a compile-time error occurs.
4.It is possible for an interface to inherit more than one field with the same name. Such a situation does not in itself cause a compile-time error. However, any attempt within the body of the interface to refer to either field by its simple name will result in a compile-time error, because such a reference is ambiguous.
5. Every method declaration in the body of an interface is implicitly `public` and `abstract`, so its body is always represented by a semicolon, not a block.
6. A method in an interface cannot be declared static, because in Java static methods cannot be abstract.
7. A method in an interface cannot be declared `native` or `synchronized`, or a compile-time error occurs, because those keywords describe implementation properties rather than interface properties. However, a method declared in an interface may be implemented by a method that is declared `native` or `synchronized` in a class that implements the interface.

Back to Question without Answer

42. QID - 2.1113 : Working with Inheritance

Which of the following is a legal return type of a method overriding the given method:

```
public Object  myMethod() {...}
```
(Select the best option.)

Correct Option is : C

~~**A.**~~ Object

~~**B.**~~ String

C. Return type can be any object since all objects can be cast to Object.

> Note that the return type cannot be a primitive such as int or char. It must be a class. So it can be Integer or Character as well.

~~**D.**~~ void

~~**E.**~~ None of the above.

Explanation:

Since the original method is returning Object, the Overriding method can return any object type because all classes in Java ultimately extend from Object. Since 1.5, Java allows covariant return types, which means an overriding method can have its return type as any subclass of the original return type of the overridden method.

Back to Question without Answer

43. QID - 2.983 : Working with Inheritance

Consider the following classes:

```
class A implements Runnable{ ...}
class B extends A implements Observer { ...}
```

(Assume that Observer has no relation to Runnable.)

and the declarations :

```
A a = new A() ;
B b = new B();
```

Which of the following Java code fragments will compile and execute without throwing exceptions?

Correct Options are : B E

~~**A.**~~ `Object o = a; Runnable r = o;`

B. `Object o = a; Runnable r = (Runnable) o;`

> Here you are explicitly telling the compiler that o refers to an object that is Runnable.

~~**C.**~~ `Object o = a; Observer ob = (Observer) o ;`

> It will compile but will fail at run time as at runtime 'a' does not refer to an object which is Observable

~~**D.**~~ `Object o = b; Observer o2 = o;`

> 'o' is declared as an Object. so same case as in choice 1.

E. `Object o = b; Runnable r = (Runnable) b;`

> Since b is declared of a type that indirectly implements `Runnable`, the compiler can figure out that b will always point to an object that is assignable to a `Runnable`. Therefore, explicit cast is not required here. It is will still work fine with the explicit cast though.

Explanation:

Although o refers to an object which is Runnable but the compiler doesn't know about it. You have to do: `Runnable r = (Runnable) o;`
You can assign a subclass object reference to superclass reference without a cast but to assign a super class object reference to a subclass (or interface) reference you need an explicit cast as in option 2.

Back to Question without Answer

44. QID - 2.1072 : Working with Inheritance

What, if anything, is wrong with the following code?

```
// Filename: TestClass.java
class TestClass implements T1, T2{
   public void m1(){}
}
interface T1{
   int VALUE = 1;
   void m1();
}
interface T2{
   int VALUE = 2;
   void m1();
}
```

Correct Option is : B

~~**A.**~~ `TestClass` cannot implement them both because it leads to ambiguity.

B. There is nothing wrong with the code.

~~**C.**~~ The code will work fine only if `VALUE` is removed from one of the interfaces.

~~**D.**~~ The code will work fine only if `m1()` is removed from one of the interfaces.

~~**E.**~~ None of the above.

Explanation:

Having ambiguous fields or methods does not cause any problems by itself but

referring to such fields/methods in an ambiguous way will cause a compile time error. So you cannot call : `System.out.println(VALUE);` because it will be ambiguous (there are two `VALUE` definitions). But the following lines are valid :

```
TestClass tc = new TestClass();
System.out.println(( ( T1) tc).VALUE);
```

However, explicit cast is not required for calling the method m1() : `( ( T2) tc).m1();`
`tc.m1()` is also fine because even though `m1()` is declared in both the interfaces, the definition to both resolves unambiguously to only one `m1()`, which is defined in TestClass.

Back to Question without Answer

45. QID - 2.1205 : Working with Inheritance

Which of these statements about interfaces are true?

Correct Options are : B D E

~~**A.**~~ Interfaces permit multiple implementation inheritance.

> Interfaces do not contain any implementations and only permit multiple interface inheritance.

B. Unlike a class, an interface can extend from multiple interfaces.

> For example - interface I1 extends I2, I2, I3{ }

~~**C.**~~ Members of an interface are never static.

> Fields are static and methods are non static.

D. Members of an interface may be static.

> methods of an interface are public and non-static. Fields are public, static and final.

E. Interfaces cannot be final.

Explanation:

An interface can extend any number of other interfaces and can be extended by any number of other interfaces. Variables in interfaces are always static and method prototypes in interfaces can never be static.

Back to Question without Answer

46. QID - 2.1127 : Working with Inheritance

Consider the following class and interface definitions (in separate files):

```
public class Sample implements IInt{
 public static void main(String[] args){
  Sample s = new Sample();  //1
  int j = s.thevalue;       //2
  int k = IInt.thevalue;    //3
  int l = thevalue;         //4
 }
}
public interface IInt{
 int thevalue = 0;
}
```

What will happen when the above code is compiled and run?

Correct Option is : E

~~**A.**~~ It will give an error at compile time at line //1.

~~**B.**~~ It will give an error at compile time at line //2.

~~**C.**~~ It will give an error at compile time at line //3

~~**D.**~~ It will give an error at compile time at line //4.

E. It will compile and run without any problem.

Explanation:

As a rule, fields defined in an interface are public, static, and final. (The methods are public and abstract.)
Here, the interface IInt defines 'thevalue' and thus any class that implements this interface inherits this field. Therefore, it can be accessed using s.thevalue or just 'thevalue' inside the class. Also, since it is static, it can also be accessed using IInt.thevalue or Sample.thevalue.

Back to Question without Answer

47. QID - 2.1288 : Working with Inheritance

What will the following code print?

```
class Baap {
    public int h = 4;
    public int getH(){ System.out.println("Baap   "+h); return h; }
}

class Beta extends Baap {
    public int h = 44;
    public int getH(){ System.out.println("Beta   "+h); return h; }
    public static void main(String[] args) {
        Baap b = new Beta();
        System.out.println(b.h+"   "+b.getH());
        Beta bb = (Beta) b;
        System.out.println(bb.h+"   "+bb.getH());
    }
}
```

-------- Output ---------

Explanation:

Always remember: Methods are overridden and variables are shadowed.
Here, b refers to an object of class Beta so `b.getH()` will always call the overridden (subclass's method). However, the type of reference of b is Baap. so b.h will always refer to Baap's h.

Further, inside Beta's `getH()`, Beta's h will be accessed instead of Baap's h because you are accessing this.h ('this' is implicit) and the type of this is Beta.

Back to Question without Answer

48. QID - 2.1359 : Working with Inheritance

Consider :

```
class A {  public void perform_work(){}  }
class B extends A {  public void perform_work(){}  }
class C extends B {  public void perform_work(){}  }
```

How can you let `perform_work()` method of A to be called from an instance method in C?

Correct Option is : E

~~**A.**~~ `( (A) this ).perform_work( );`

~~**B.**~~ `super.perform_work( );`

~~**C.**~~ `super.super.perform_work( );`

~~**D.**~~ `this.super.perform_work( );`

E. It is not possible.

Explanation:

The method in C needs to call a method in a superclass two levels up. But `super` is a keyword and not an attribute so `super.super.perform_work( )` strategy will not work. There is no way to go more than one level up for methods.
Remember that this problem doesn't occur for instance variables because variable are never overridden. They are shadowed. So to access any of the super class's variable,

you can unshadow it using a cast. For example, `((A) c).data;` This will give you the data variable defined in A even if it is shadowed in B as well as in C.

Back to Question without Answer

49. QID - 2.1019 : Working with Inheritance

You are modeling a class hierarchy for living things. You have a class `LivingThing` which has an abstract method `reproduce()`.
Now, you want to have 2 subclasses of `LivingThing` - `Plant` and `Animal`. Both do reproduce but the mechanisms are different. What would you do?

Correct Option is : C

~~**A.**~~ Overload the `reproduce` method in Plant and Animal classes

~~**B.**~~ Overload the `reproduce` method in LivingThing class.

C. Override the `reproduce` method in Plant and Animal classes

~~**D.**~~ Either overload or override `reproduce` in `Plant` and `Animal` classes, it depends on the preference of the designer.

Explanation:

This kind of scenario where the subclass HAS the behavior of the base class but implements it in a different way is called as overriding. Here, both `Plant` and `Animal` reproduce, so they have the behavior of the base class but they do it differently, so you have to override the base class method in their code. Inheritance is always involved in overriding.
Overloading is quite different, when you want to do similar (not same) things but the inputs are different then you overload a method. For example, you may have two add methods:
`add(int i1, int i2)` and `add(ComplexNo c1, ComplexNo 2)`. Here both are doing similar things (that is why both are named as add) but inputs are different. Both

are two entirely different methods and there is no inheritance involved.

Back to Question without Answer

50. QID - 2.1130 : Working with Inheritance

You want to invoke the overridden method (the method in the base class) from the overriding method (the method in the derived class) named `m()`.
Which of the following constructs which will let you do that?

Correct Option is : A

A. `super.m();`

~~**B.**~~ `super.this();`

~~**C.**~~ `base.m();`

~~**D.**~~ `parent.m();`

~~**E.**~~ `super();`

Explanation:

Note that calling `super();` means you are trying to call the super class's constructor. But you can't call the super class's constructor (or its own constructor) from a method (because by the time a method gets to run, the object has already been constructed), therefore calling `super()` from a method is not valid.
`super()` can only be the first statement of a constructor.

Back to Question without Answer

51. QID - 2.1367 : Working with Inheritance

What should be inserted in the code given below at line marked //10:

```
class MyClass{
}

class MyComparable implements Comparable<MyClass>{
   public int compareTo(  *INSERT CODE HERE*  x ){ //10
         return 0;
   }
}
```

Correct Option is : B

~~**A.**~~ `Object`

B. `MyClass`

~~**C.**~~ `Object<MyClass>`

~~**D.**~~ `Comparable<MyClass>`

~~**E.**~~ Comparable

Explanation:

Since MyComparable class specifies that it implements the Comparable interface that has been typed to MyClass, it must implement compareTo method that takes a MyClass.
Had it not declared a typed Comparable in its implements clause, compareTo(Object

x) would have been correct.

Back to Question without Answer

52. QID - 2.1165 : Working with Inheritance

Drag blue items xxx and yyy on yellow targets
to complete the code.

xxx yyy

```
class XXX {
    public void m() throws Exception {
        throw new Exception();
    }
}
class YYY extends XXX{
    public void m(){    }
}

public class TestClass {

    public static void main(String[] args)
    {
        YYY     s   = new  YYY     ();

        s.m();
    }
}
```

Explanation:

1. The overriding method may choose to have no `throws` clause even if the overridden method has a `throws` clause.
2. Whether a call needs to be wrapped in a try/catch or whether the enclosing method requires a throws clause depends on the class of the reference and not the class of the actual object.

Here, if you define s of type XXX, the call `s.m()` will have to be wrapped into a try/catch because `main()` doesn't have a throws clause. But if you define s of class YYY, there is no need of try catch because YYY's `m()` does not throw an exception. Now, if the class of s is YYY, you cannot assign it an object of class XXX because XXX is a superclass of YYY. So the only option is to do:

```
YYY s = new YYY();
```

Back to Question without Answer

53. QID - 2.1261 : Working with Inheritance

Consider the following code:

```
public class SubClass extends SuperClass{
     int i, j, k;
     public SubClass( int m, int n )      {  i = m ;  j = m ;  } //1
     public SubClass( int m )  {   super(m );   } //2
 }
```

Which of the following constructors MUST exist in SuperClass for SubClass to compile correctly?

Correct Options are : C D

~~**A.**~~ It is ok even if no explicit constructor is defined in SuperClass

> The //2 will fail as it needs a constructor taking an int!

~~**B.**~~ `public SuperClass(int a, int b)`

> It is not used anywhere so it is not necessary.

C. `public SuperClass(int a)`

> Because it is called in the second constructor of SubClass.

D. `public SuperClass()`

> The default no args constructor will not be provided because SuperClass has to define one arg constructor.

~~**E.**~~ only `public SuperClass(int a)` is required.

You'll have to explicitly define a no args constructor because it is needed in the first constructor of SubClass.

Back to Question without Answer

54. QID - 2.1129 : Working with Inheritance

Consider the following class:

```
public class PortConnector{
   public PortConnector(int port) throws IOException{
   ...lot of valid code.
   }
   ...other valid code.
}
```

You want to write another class `CleanConnector` that extends from `PortConnector`. Which of the following statements should hold true for `CleanConnector` class?

Correct Option is : F

~~**A.**~~ It is not possible to define `CleanConnector` that does not throw `IOException` at instantiation.

> It is possible. You can also throw a superclass of `IOException` from the `CleanConnector`'s constructor. For example, the following is valid:
>
> ```
> class CleanConnector extends PortConnector {
>    public CleanConnector(int port) throws Exception {
>          super(port);
>    }
> }
> ```

~~**B.**~~ `PortConnector` class itself is not valid because you cannot throw any exception from a constructor.

> A constructor is free to throw any exception.

~~**C.**~~ `CleanConnector`'s constructor cannot throw any exception other than `IOException`.

It can throw any exception but it must also throw `IOException` (or its super class). So the following is valid:

```
class CleanConnector extends PortConnector {
   public CleanConnector(int port) throws IOException,
FileNotFoundException, SomeOtherCheckedException {
         super(port);
   }
}
```

D. `CleanConnector`'s constructor cannot throw any exception other than subclass of `IOException`.

As described above, it can throw any exception but it must throw `IOException` (or its superclass) as well.

E. `CleanConnector`'s constructor cannot throw any exception other than superclass of `IOException`.

As described above, it can throw any exception but it must throw `IOException` (or its superclass) as well.

F. None of these.

Observe that the rule for overriding a method is opposite to the rule for constructors. An overriding method cannot throw a superclass exception, while a constructor of a subclass cannot throw subclass exception (Assuming that the same exception or its super class is not present in the subclass constructor's throws clause). For example:

```
class A{
  public A() throws IOException{ }
  void m() throws IOException{ }
}
```

```
  class B extends A{
    //IOException is valid here, but FileNotFoundException is
invalid
    public B() throws IOException{ }

    //FileNotFoundException is valid here, but Exception is
invalid
    void m() throws FileNotFoundException{ }
}
```

(Note: `FileNotFoundException` is a subclass of `IOException`, which is a subclass of `Exception`)

If the subclass constructor's throws clause includes the same exception or its superclass, then it can throw any other exception as well.

Explanation:

As PortConnector has only one constructor, there is only one way to instantiate it. Now, to instantiate any subclass of PortConnector, the subclass's constructor should call super(int). But that throws throws IOException. And so it (or its super class) must be defined in the throws clause of subclass's constructor. Note that you cannot do something like:

```
public CleanConnector(){
   try{ super(); }catch(Exception e){} //WRONG : call to super must b
}
```

Remember: Constructor must declare all the checked exceptions declared in the base constructor (or the super classes of the checked exceptions). They may add other exception. This behavior is exactly opposite from that of methods. The overriding method cannot throw any exception other than overridden method. It may throw subclasses of those exceptions.

Back to Question without Answer

55. QID - 2.1315 : Working with Inheritance

Note: This question may be considered too advanced for this exam.

Given the declaration

```
interface Worker { void perform_work(); }
```

which of the following methods/classes are valid?

Correct Options are : B D

~~A.~~
```
Worker getWorker(int i){
   return new Worker(){   public void perform_work() {
        System.out.println(i); }    };
}
```

Since method parameter i is not final, it cannot be accessed from perform_work().

B.
```
Worker getWorker(final int i){
   return new Worker() {   public void perform_work() {
   System.out.println(i); }    };
}
```

Since method paramter 'i' is final, it can be accessed from perform_work();

~~C.~~
```
Worker getWorker(int i){
   int x = i;
   class MyWorker implements Worker  {
         public void perform_work() { System.out.println(x); }    };
   return new MyWorker();
}
```

x is not accessible from within perform_work() either. In fact, i and x are

similar for all practical purposes in terms of accessibility.

D.
```
Worker getWorker(final int i){
   class MyWorker implements Worker {
       public void perform_work() { System.out.println(i); }    };
   return new MyWorker();
}
```

Explanation:

Do not get confused with option 2. You are not instantiating an interface, you are instantiating an anonymous class that implements the interface Worker.

FYI, if you have a nested static class `MyWorker` in `TestClass` as follows,

```
public class TestClass{
   public static class MyWorker implements Worker{
       public MyWorker(int i){ }
       public void perform_work(){ }
   }
}
```

`MyWorker` can be instantiated in any other class by doing:
`new TestClass.MyWorker( someInt );`

There is no instance of `TestClass` associated with the `MyWorker` class in this case. Inside `TestClass` you can instantiate MyWorker directly `(new MyWorker(someInt))` in static or non static context.

Remember: A nested class is any class whose declaration occurs within the body of another class or interface. A top level class is a class that is not a nested class. An inner class is a nested class that is not explicitly or implicitly declared static. A class defined inside an interface is implicitly static.

Back to Question without Answer

56. QID - 2.1201 : Working with Inheritance

Consider the following interface definition:

```
interface Bozo{
         int type = 0;
         public void jump();
}

Now consider the following class:

public class Type1Bozo implements Bozo{
         public Type1Bozo(){
            type = 1;
         }

         public void jump(){
            System.out.println("jumping..."+type);
         }

         public static void main(String[] args){
            Bozo b = new Type1Bozo();
            b.jump();
         }
}
```

What will the program print when compiled and run?

Correct Option is : C

~~**A.**~~ jumping...0

~~**B.**~~ jumping...1

C. This program will not compile.

~~**D.**~~ It will throw an exception at runtime.

Explanation:

Fields defined in an interface are ALWAYS considered as public, static, and final. (Methods are public and abstract.) Even if you don't explicitly define them as such. In fact, you cannot even declare a field to be private or protected in an interface. Therefore, you cannot assign any value to 'type' outside the interface definition.

Back to Question without Answer

57. QID - 2.1344 : Working with Inheritance

Complete the following code by dragging one of the blue items on to the yellow target.

Object MyClass Object<MyClass>

Comparable<MyClass> Comparable

```
class MyClass
{
}

class MyComparable implements Comparable<MyClass>
{
    public int compareTo( MyClass x)
    {
        return 0;
    }
}
```

Explanation:

Since MyComparable class specifies that it implements the Comparable interface that has been typed to MyClass, it must implement compareTo method that takes a MyClass.
Had it not declared a typed Comparable in its implements clause, compareTo(Object x) would have been correct.

Back to Question without Answer

58. QID - 2.1309 : Working with Inheritance

What can be inserted at //1 and //2 in the code below so that it can compile without errors:

```
class Doll{
    String name;
    Doll(String nm){
        this.name = nm;
    }
}

class Barbie extends Doll{
    Barbie(){
        //1
    }
    Barbie(String nm){
        //2
    }
}
  public class TestClass {
       public static void main(String[] args) {
          Barbie b = new Barbie("mydoll");
       }
   }
```

Correct Options are : A B

A. `this("unknown");` at 1 and `super(nm);` at 2

B. `super("unknown");` at 1 and `super(nm);` at 2

C. `super();` at 1 and `super(nm);` at 2

> `super();` at 1 will not compile because super class Doll does not have a no args constructor.

~~**D.**~~ `super();` at 1 and `Doll(nm);` at 2

> `super();` at 1 will not compile because super class Doll does not have a no args constructor. `Doll(nm);` at 2 is an invalid syntax for calling the super class's construtor.

~~**E.**~~ `super("unknown");` at 1 and `this(nm);` at 2

> `this(nm);` at 2 will not compile because it is a recursive call to the same constructor.

~~**F.**~~ `Doll();` at 1 and `Doll(nm);` at 2

> Both are using invalid syntax for calling the super class's construtor.

Explanation:

Since the super class `Doll` explicitly defines a constructor, compiler will not provide the default no-args constructor. Therefore, each of `Barbie`'s constructor must directly or indirectly call `Doll`'s string argument constructor, otherwise it will not compile. Although not relevant for this question, it is interesting to know that `super(name);` at //1 or //2, would not be valid because `name` is defined in the superclass and so it cannot be used by a subclass until super class's constructor has executed. For the same reason, `this(name);` cannot be used either.

Back to Question without Answer

59. QID - 2.996 : Working with Inheritance

Consider the following program:

```
class Game {
  public void play() throws Exception   {
    System.out.println("Playing...");
  }
}

class Soccer extends Game {
   public void play(String ball)    {
      System.out.println("Playing Soccer with "+ball);
   }
}

public class TestClass {
   public static void main(String[] args) throws Exception  {
       Game g = new Soccer();
       // 1
       Soccer s = (Soccer) g;
       // 2
   }
}
```

Which of the given options can be inserted at //1 and //2?

Correct Options are : C D

~~**A.**~~ It will not compile as it is.

> There is no problem with the existing code.

~~**B.**~~ It will throw an `Exception` at runtime if it is run as it is.

> `Soccer s = (Soccer) g;` is a valid because `g` does refer to an object of class

`Soccer` at run time. So there will be no exception at run time.

C. `g.play();` at //1 and `s.play("cosco");` at //2

This is valid because `g` is of type `Game`, which has the no-args play method and `s` is of type `Soccer`, which has defined `play(String )` method.

D. `g.play();` at //1 and `s.play();` at //2

This is valid because `g` is of type `Game`, which has the no-args play method and `s` is of type `Soccer`, which inherits that method.

E. ~~`g.play("cosco");`~~ at //1 and `s.play("cosco");` at //2

`g.play("cosco")` is not valid because even though the object referred to by `g` is of class `Soccer`, the reference type of `g` is `Game`, which does not have `play(String )` method.

Back to Question without Answer

60. QID - 2.1047 : Working with Inheritance

An overriding method must have a same parameter list and the same return type as that of the overridden method.

Correct Option is : B

~~**A.** True~~

B. False

Explanation:

This would have been true prior to Java 1.5. But from Java 1.5, an overriding method is allowed to change the return type to any subclass of the original return type, also known as covariant return type. This does not apply to primitives, in which case, the return type of the overriding method must match exactly to the return type of the overridden method.

Back to Question without Answer

61. QID - 2.837 : Working with Inheritance

Consider the following code:

```
interface Flyer{ }
class Bird implements Flyer { }
class Eagle extends Bird { }
class Bat { }

public class TestClass {

    public static void main(String[] args) {
        Flyer f = new Eagle();
        Eagle e = new Eagle();
        Bat b = new Bat();

        if(f instanceof Bird) System.out.println("f is a Bird");
        if(e instanceof Flyer) System.out.println("e is a Flyer");
        if(b instanceof Flyer) System.out.println("f is a Bird");
    }
}
```

What will be printed when the above code is compiled and run?

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will throw an exception when run.

C. f is a Bird
e is a Flyer

At run time, f points to an object of class Eagle. Now, Eagle extends Bird so f

instanceof Bird returns true.
e points to an object of class Eagle. Eagle extends Bird, which in turn implements Flyer. So an Eagle is a Flyer. Therefore, e instanceof Flyer will also return true.

b points to an object of class Bat, which does not extend from Bird. Therefore, b instanceof Flyer returns false.

~~**D.**~~ f is a Bird

~~**E.**~~ e is a Flyer

Explanation:

Note that there is no compilation issue with `b instanceof Flyer` because `Flyer` is an interface and it is possible for b to point to an object of a class that is a sub class of `Bat` and also implements `Flyer`. So the compiler doesn't complain. If you make Bat class as final, `b instanceof Flyer` will not compile because the compiler knows that it is not possible for b to point to an object of a class that implements `Flyer`.

Back to Question without Answer

62. QID - 2.1195 : Working with Inheritance

Which letters will be printed when the following program is run?

```
public class TestClass{
  public static void main(String args[]){
    B b = new C();
    A a = b;
    if (a instanceof A) System.out.println("A");
    if (a instanceof B) System.out.println("B");
    if (a instanceof C) System.out.println("C");
    if (a instanceof D) System.out.println("D");
  }
}
class A { }
class B extends A { }
class C extends B { }
class D extends C { }
```

Correct Options are : A B C

A. A will be printed.

B. B will be printed.

C. C will be printed.

~~**D.**~~ D will be printed.

Explanation:

The program will print A, B and C when run. The object denoted by reference a is of

type C. The object is also an instance of A and B, since C is a subclass of B and B is a subclass of A. The object is not an instance of D.

Back to Question without Answer

63. QID - 2.1147 : Working with Inheritance

Given the following code, which statements are true?

```
class A{
   int i;
}
class B extends A{
   int j;
}
```

Correct Options are : A D E

A. Class B extends class A.

~~B.~~ Class B is the superclass of class A.

A is the super class of B.

~~C.~~ Class A inherits from class B.

B inherits from A

D. Class B is a subclass of class A.

Class B is a subclass of class A. Given the declaration "class B extends A" we can conclude that class B extends class A, class A is the superclass of class B, class B is a subclass of class A, and class B inherits from class A, which means that objects of class B also have all the members that objects of class A have.

E. Objects of class B will always have a member variable named i .

Note that 'i' is not public or protected. So it will be inherited only if both the

classes are in same package.

Explanation:

Here are a few good words from the Java Language Specification:
Members of a class that are declared private are not inherited by subclasses of that class. Only members of a class that are declared protected or public are inherited by subclasses declared in a package other than the one in which the class is declared. Constructors and static initializers are not members and therefore are not inherited.

Back to Question without Answer

64. QID - 2.910 : Working with Inheritance

Consider the following code:

```
interface Flyer{ String getName(); }

class Bird implements Flyer{
    public String name;
    public Bird(String name){
        this.name = name;
    }
    public String getName(){ return name; }
}

class Eagle extends Bird {
    public Eagle(String name){
        super(name);
    }
}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Flyer f = new Eagle("American Bald Eagle");
        //PRINT NAME HERE
   }
}
```

Which of the following lines of code will print the name of the Eagle object?

Correct Options are : B C D

~~**A.**~~ `System.out.println(f.name);`

B. `System.out.println(f.getName());`

C. `System.out.println(((Eagle)f).name);`

D. `System.out.println(((Bird)f).getName());`

~~**E.**~~ `System.out.println(Eagle.name);`

> name is not a static field in class Eagle.

~~**F.**~~ `System.out.println(Eagle.getName(f));`

> This option doesn't make any sense.

Explanation:

While accessing a method or variable, the compiler will only allow you to access a method or variable that is visible through the class of the reference.

When you try to use `f.name`, the class of the reference f is `Flyer` and `Flyer` has no field named "`name`", thus, it will not compile. But when you cast f to `Bird` (or `Eagle`), the compiler sees that the class `Bird` (or `Eagle`, because `Eagle` inherits from `Bird`) does have a field named "`name`" so `((Eagle)f).name or ((Bird)f).name` will work fine.

`f.getName()` will work because `Flyer` does have a `getName()` method.

Back to Question without Answer

65. QID - 2.998 : Working with Inheritance

What will the following program print when compiled and run?

```
class Game{
  public void play() throws Exception{
    System.out.println("Playing...");
  }
}

public class Soccer extends Game{
   public void play(){
      System.out.println("Playing Soccer...");
   }
   public static void main(String[] args){
       Game g = new Soccer();
       g.play();
   }
}
```

Correct Option is : A

A. It will not compile.

~~**B.**~~ It will throw an Exception at runtime.

~~**C.**~~ Playing Soccer...

~~**D.**~~ Playing...

~~**E.**~~ None of these.

Explanation:

Observe that play() in Game declares Exception in its throws clause. Further, class Soccer overrides the play() method without any throws clause. This is valid because a list of no exception is a valid subset of a list of exceptions thrown by the superclass method.
Now, even though the actual object referred to by 'g' is of class Soccer, the class of the variable g is of class Game. Therefore, at compile time, compiler assumes that g.play() might throw an exception, because Game's play method declares it, and thus expects this call to be either wrapped in a try-catch or the main method to have a throws clause for the main() method.

Back to Question without Answer

66. QID - 2.961 : Working with Inheritance

Which one of these is a proper definition of a class Car that cannot be sub-classed?

Correct Option is : E

~~**A.**~~ `class Car { }`

> This can be subclassed.

~~**B.**~~ `abstract class Car { }`

> it cannot be instantiated but it can be subclassed.

~~**C.**~~ `native class Car { }`

> Classes and variables can't be declared native. Only methods can be native.

~~**D.**~~ `static class Car { }`

> package level classes can't be declared static.

E. `final class Car { }`

> final keyword prevents a class from being subclassed and a method from being overridden.

Explanation:

A class can be extended unless it is declared final. While declaring a method, static usually implies that it is also final, this is not true for classes.
An inner class can be declared static and still be extended. Notice the distinction. For

classes, final means it cannot be extended, while for methods, final means it cannot be overridden in a subclass.
The native keyword can only be used on methods, not on classes and instance variables.

Back to Question without Answer

67. QID - 2.1203 : Working with Inheritance

A method with no access modifier can be overridden by a method marked protected.

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

An Overriding method is allowed to make the overridden method more accessible, and since protected is more accessible than default (package), this is allowed. Note that protected access will allow access to the subclass even if the subclass is in a different package but package access will not.

Back to Question without Answer

68. QID - 2.1101 : Working with Inheritance

Consider the following code:

```
class Base{
 private float f = 1.0f;
 void setF(float f1){ this.f = f1; }
}
class Base2 extends Base{
 private float f = 2.0f;
 //1
}
```

Which of the following options is a valid example of overriding?

Correct Options are : A C

A. `protected void setF(float f1){ this.f = 2*f1; }`

protected is less restrictive than default, so it is valid.

B. `public void setF(double f1){ this.f = (float) 2*f1; }`

Since the parameter type is different, it is overloading not overriding.

C. `public void setF(float f1){ this.f = 2*f1; }`

public is less restrictive than default, so it is valid.

D. `private void setF(float f1){ this.f = 2*f1; }`

private is more restrictive than default, so it is NOT valid.

E. ~~`float setF(float f1){ this.f = 2*f1; return f;}`~~

return types must match.

Explanation:

An overriding method can be made less restrictive than the overridden method. The restrictiveness of access modifiers is as follows:
private>default>protected>public (where private is most restrictive and public is least restrictive).

Note that there is no modifier named default. The absence of any access modifiers implies default access.

Back to Question without Answer

69. QID - 2.1222 : Working with Inheritance

Consider the following class hierarchy

```
class A{
   public void m1() {    }
}
class B extends A{
   public void m1() {    }
}
class C extends B{
   public void m1(){
      /*  //1
      ... lot of code.
      */
   }
}
```

Correct Options are : A B

A. You cannot access class A's `m1()` from class C for the same object (i.e. `this`).

B. You can access class B's `m1()` using `super.m1()` from `class C`.

~~**C.**~~ You can access `class A`'s `m1()` using `( (A) this ).m1()` from `class C`.

> Note that selection of method to be executed depends upon the actual object class. So no matter what you do, in `class C` you can only access C's `m1()` even by casting this to B or A. So, this option will not work.

~~**D.**~~ You can access `class A`'s `m1()` using `super.super.m1()` from `class C`.

Explanation:

There is no construct like super.super. So, there is no way you can access `m1()` of A from C.

Back to Question without Answer

70. QID - 2.1306 : Working with Inheritance

Which of these statements concerning interfaces are true?

Correct Options are : A C

A. An interface may extend an interface.

> Unlike a class, an interface can extend from multiple interfaces.

~~**B.**~~ An interface may extend a class and may implement an interface.

> interface does not implement anything. It can extend another interface but not class.

C. A class can implement an interface and extend a class.

~~**D.**~~ A class can extend an interface and can implement a class.

~~**E.**~~ An interface can only be implemented and cannot be extended.

> It can be extended by another interface.

Explanation:

The keyword implements is used when a class inherits method prototypes from an interface. The keyword extends is used when an interface inherits from another interface, or a class inherits from another class.

Back to Question without Answer

71. QID - 2.1250 : Working with Inheritance

Which statements, when inserted at line 1, will cause an exception at run time?

```
class B {}
class B1 extends B {}
class B2 extends B {}
public class ExtendsTest{
  public static void main(String args[]){
     B b = new B();
     B1 b1 = new B1();
     B2 b2 = new B2();
     // insert statement here
  }
}
```

Correct Option is : C

~~**A.**~~ `b = b1;`

> There won't be a problem anytime because B1 is a B

~~**B.**~~ `b2 = b;`

> It fails at Compile time as an object referenced by b may not be a B2, so an explicit cast will be needed.

C. `b1 = (B1) b;`

> It will pass at compile time but fail at run time as the actual object referenced by b is not a B1.

~~**D.**~~ `b2 = (B2) b1;`

It will not compile because b1 can never point to an object of class B2.

E. ~~`b1 = (B) b1;`~~

This won't compile. Another cast is needed. i.e. `b1 = (B1) (B) b1;`

Back to Question without Answer

72. QID - 2.1168 : Working with Inheritance

Consider the following class hierarchy:

```
A
|
B1,  B2
|
C1, C2
```

(B1 and B2 are subclasses of A and C1, C2 are subclasses of B1) Which of the following statements are correct? Assume that objectOfA, objectOfC1, etc. are objects of classes A and C1 respectively.

Correct Option is : B

~~**A.**~~ `objectOfC2 instanceof B2` will return `true`.

> `objectOfC2` is an instance of C2 and as C2 extends B1, it cannot be a subclass of B2 and so `objectOfC2` instanceof B2 cannot be `true`.

B. `objectOfC1 instanceof B1` will return `true`.

> This is because C1 extends B1. Therefore, anything that is a C1 is a B1. It is like saying a Dog is a Pet or a Cat is a Pet, if Dog and Cat extend from Pet.

~~**C.**~~ `objectOfA instanceof B1` will return `true`.

~~**D.**~~ C1 c1 = objectOfA; is a valid statement.

> Since c1 is declared of type C1, an object of class A, cannot be assigned to c1 because A is not a C1. A C1 is an A. So `A a = objectOfC1;` would have been valid.

E. ~~`B1 b1 = objectOfB2;`~~ is a valid statement.

B2 does not extend from B1 and so there is no is-a relation between B1 and B2. Therefore, an object of class B2 cannot be assigned to a variable of class B1.

Back to Question without Answer

73. QID - 2.1123 : Working with Inheritance

Consider the contents of following two files:

```
//File A.java
package a;
public class A{
   A(){ }
   public void  print(){ System.out.println("A"); }
}

//File B.java
package b;
import a.*;
public class B extends A{
   B(){ }
   public void  print(){ System.out.println("B"); }
   public static void main(String[] args){
      new B();
   }
}
```

What will be printed when you try to compile and run class B?

Correct Option is : C

~~**A.**~~ It will print A.

~~**B.**~~ It will print B.

C. It will not compile.

Because A() is not accessible in B.

~~**D.**~~ It will compile but will not run.

~~**E.**~~ None of the above.

Explanation:

Note that there is no modifier for A's constructor. So it has default access. This means only classes in package a can use it. Also note that class B is in a different package and is extending from A. In B's constructor the compiler will automatically add super() as the first line. But since A() is not accessible in B, this code will not compile.

Back to Question without Answer

74. QID - 2.1149 : Working with Inheritance

Given the following class definitions :

```
interface MyIface{};
class A {};
class B extends A implements MyIface{};
class C implements MyIface{};
```

and the following object instantiations:

```
A a = new A();
B b = new B();
C c = new C();
```

Which of the following assignments are legal at compile time?

Correct Option is : C

~~**A.**~~ `b = c;`

There is no relation between b and c.

~~**B.**~~ `c = b;`

There is no relation between b and c.

C. `MyIface i = c;`

Because C implements I.

~~D.~~ `c = (C) b;`

> Compiler can see that in no case can an object referred to by b can be of class c. So it is a compile time error.

~~E.~~ `b = a;`

> It will fail at compile time because a is of class A and can potentially refer to an object of class A, which cannot be assigned to b, which is a variable of class B. To make it compile, you have to put an explicit cast, which assures the compiler that a will point to an object of class B (or a subclass of B) at run time. Note that, in this case, an explicit cast can take it through the compiler but it will then fail at run time because a does not actually refer to an object of class B (or a subclass of B), so the JVM will throw a ClassCastException.

Explanation:

The statements `c = b` and `b = c` are illegal, since neither of the classes C and B is a subclass of the other. Even though a cast is provided, the statement `c = (C) b` is illegal because the object referred to by b cannot ever be of type C.

Back to Question without Answer

75. QID - 2.1219 : Working with Inheritance

What will be the output of compiling and running the following program:

```
class TestClass implements I1, I2{
   public void m1() { System.out.println("Hello"); }
   public static void main(String[] args){
      TestClass tc = new TestClass();
      ( (I1) tc).m1();
   }
}
interface I1{
   int VALUE = 1;
   void m1();
}
interface I2{
   int VALUE = 2;
   void m1();
}
```

Correct Option is : A

A. It will print `Hello`.

~~B.~~ There is no way to access any `VALUE` in TestClass.

~~C.~~ The code will work fine only if `VALUE` is removed from one of the interfaces.

> It works even now.

~~D.~~ It will not compile.

~~**E.**~~ None of the above.

Explanation:

Having ambiguous fields does not cause any problems but referring to such fields in an ambiguous way will cause a compile time error. So you cannot call :

`System.out.println(VALUE)` as it will be ambiguous.

as there is no ambiguity in referring the field:

```
TestClass tc = new TestClass();
System.out.println(( ( I1) tc).VALUE);
```

So, any of the `VALUE` fields can be accessed by casting.

Back to Question without Answer

76. QID - 2.1220 : Working with Inheritance

Given the following classes and declarations, which of these statements about //1 and //2 are true?

```
class A{
   private int i = 10;
   public void  f(){}
   public void g(){}
}

class B extends A{
   public int i = 20;
   public void g(){}
}

public class C{
   A a = new A();//1
   A b = new B();//2
}
```

Correct Option is : E

~~**A.**~~ `System.out.println(b.i);` will print 10.

> Since variable b is declared as of class A, you cannot do b.i even if the actual object is of class B because i in A is private.

~~**B.**~~ The statement `b.f( );` will give compile time error..

> `class A` has `f()` so `b.f()` is legal.

~~**C.**~~ `System.out.println(b.i);` will print 20

Since variable b is declared as of class A, you cannot do b.i even if the actual object is of class B because i in A is private.

D. All the above are correct.

E. None of the above statements is correct.

Back to Question without Answer

77. QID - 2.1002 : Working with Inheritance

Given the following definitions and reference declarations:

```
interface I1 { }
interface I2 { }
class C1 implements I1 { }
class C2 implements I2 { }
class C3 extends C1 implements I2 { }
C1 o1;
C2 o2;
C3 o3;
```

Which of these statements are legal?

Correct Options are : A D E

A. `class C4 extends C3 implements I1, I2 { }`

> Although, the `implements I1, I2` is redundant here because C3 already implements I1 and I2, it is not invalid.

~~B.~~ `o3 = o1;`

> superclass reference cannot be assigned to subclass reference without explicit cast.

~~C.~~ `o3 = o2;`

> There is no way a reference of class C2 (which is o2) can point to an object of class C3 because C2 and C3 have no inheritance relationship. So this assignment is rejected at compile time itself.

D. `I1 i1 = o3; I2 i2 = (I2) i1;`

This is valid because at run time `i1` actually refers to an object that implements I2.

E. `I1 b = o3;`

Because C3 extends C1 which implements I1.

Back to Question without Answer

78. QID - 2.1037 : Working with Inheritance

What will the following code print when compiled and run?

```
class Base{
   void methodA(){
      System.out.println("base - MethodA");
   }
}

class Sub extends Base{
   public void methodA(){
      System.out.println("sub - MethodA");
   }
   public void methodB(){
      System.out.println("sub - MethodB");
   }
   public static void main(String args[]){
      Base b=new Sub(); //1
      b.methodA(); //2
      b.methodB(); //3
   }
}
```

Correct Option is : E

~~**A.**~~ `sub - MethodA` and `sub - MethodB`

~~**B.**~~ `base - MethodA` and `sub - MethodB`

~~**C.**~~ Compile time error at `//1`

~~**D.**~~ Compile time error at `//2`

E. Compile time error at `//3`

Explanation:

The point to understand here is, `b` is declared to be a reference of class Base and `methodB()` is not defined in Base. So the compiler cannot accept the statement `b.methodB()` because it only verifies the validity of a call by looking at the declared class of the reference.
For example, the compiler is able to verify that `b.methodA()` is a valid call because class Base has method `methodA`. But it does not "bind" the call. Call binding is done at runtime by the jvm and the jvm looks for the actual class of object referenced by the variable before invoking the method.

Back to Question without Answer

79. QID - 2.1154 : Working with Inheritance

Consider the following code:

```
class A{
 public XXX m1(int a){
   return a*10/4-30;
 }
}
class A2 extends A{
 public YYY m1(int a){
   return a*10/4.0;
 }
}
```

What can be substituted for XXX and YYY so that it can compile without any problems?

Correct Option is : C

~~**A.**~~ int, int

> `a*10/4.0;` generates a double so, A2's m1() cannot return an int. (It will need a cast otherwise: `return (int) (a*10/4.0);`)

~~**B.**~~ int, double

> The return type should be same for overridden and overriding method.

C. double, double

> `a*10/4-30;` generates an int which can be returned as a double without any cast.

~~**D.**~~ double, int

The return type should be same for overridden and overriding method.

~~**E.**~~ Nothing, they are simply not compatible.

Explanation:

Note that when a method returns objects (as opposed to primitives, like in this question), the principle of covariant returns applies. Meaning, the overriding method is allowed to return a subclass of the return type defined in the overridden method. Thus, if a base class's method is: `public A m();` then a subclass is free to override it with: `public A1 m();` if A1 extends A.

Back to Question without Answer

80. QID - 2.907 : Working with Inheritance

Consider the following code appearing in Eagle.java

```
class Bird {
    private Bird(){     }
}
class Eagle extends Bird {
    public String name;
    public Eagle(String name){
        this.name = name;
    }

    public static void main(String[] args) {
        System.out.println(new Eagle("Bald Eagle").name);
    }
}
```

What needs to be done to make this code compile?

Correct Option is : D

~~**A.**~~ Nothing, it will compile as it is.

~~**B.**~~ Make Eagle class declaration public:

```
public class Eagle { ... }
```

~~**C.**~~ Make the Eagle constructor private:

```
private Eagle(String name){ ... }
```

D. Make Bird constructor public:

```
public Bird() { ... }
```

~~**E.**~~ Insert super(); as the first line in Eagle constructor:

```
public Eagle(String name){
    super();
    this.name = name;
}
```

Explanation:

Note that if a subclass class constructor doesn't explicitly call the super class constructor, the compiler automatically inserts super(); as the first statement of the base class constructor. So option 5 is not needed.

Since the constructor of Bird is private, the subclass cannot access it and therefore, it needs to be made public.

Back to Question without Answer

81. QID - 2.1209 : Working with Inheritance

Which statements concerning the following code are true?

```
class A{
  public A() {} // A1
  public A(String s) {  this();  System.out.println("A :"+s);  }  //
}

class B extends A{
  public int B(String s) {  System.out.println("B :"+s);  return 0;
}
class C extends B{
    private C(){ super(); } // C1
    public C(String s){  this();  System.out.println("C :"+s);  } //
    public C(int i){} // C3
}
```

Correct Options are : A B C D

A. At least one of the constructors of each class is called as a result of constructing an object of class C.

> To create any object one and only one constructor of that class and each of the super classes is called. (A constructor may delegate the construction to another constructor of the same class by calling this(...) as the first statement.)

B. Constructor at //A2 will never be called in creation of an object of class C

> Because B has no defined constructor and so a default no-argument constructor will be called, which will call the no-argument constructor of A

C. Class C can be instantiated only in two ways by users of this class.

As one constr. is private, users of this class (i.e. classes other than class C) can use only other 2 public constr.

D. //B1 will never be called in creation of objects if class A, B, or C

Because //B1 is not a constructor. Note that it is returning an int. A constructor does not have any return type, not even void.

~~**E.**~~ The code will not compile.

Back to Question without Answer

82. QID - 2.885 : Working with Inheritance

Given:

```
//Insert code here

   public abstract void draw();
}


//Insert code here
   public void draw(){  System.out.println("in draw..."); }
}
```

Which of the following lines of code can be used to complete the above code?

Correct Options are : D F

~~**A.**~~ class Shape {

and

class Circle extends Shape {

> Since there is an abstract method in the first class, the class must be declared abstract.

~~**B.**~~ public class Shape {

and

class Circle extends Shape {

~~**C.**~~ abstract Shape {

and

public class Circle extends Shape {

> class keyword is missing from the first declaration.

D. public abstract class Shape {

and

class Circle extends Shape {

~~**E.**~~ public abstract class Shape {

and

class Circle implements Shape {

> You can only implement an interface not a class. So Circle implements shape is wrong.

F. public interface Shape {

and

class Circle implements Shape {

> By default all the methods of an interface are public and abstract so there is no need to explicitly specify the "abstract" keyword for the draw() method if you make Shape an interface. But it is not wrong to do so.

Back to Question without Answer

Working with Java Data Types - String, StringBuilder

Exam Objectives -

Manipulate data using the StringBuilder class and its methods Create and manipulate strings

01. QID - 2.1022

What will be the output of the following program (excluding the quotes)?

```
public class SubstringTest{
   public static void main(String args[]){
      String String = "string isa string";
      System.out.println(String.substring(3, 6));
   }
}
```

Select 1 option

A. It will not compile.

B. "ing is"

C. "ing isa"

D. "ing " (There is a space after g)

E. None of the above.

Check Answer

02. QID - 2.1021

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      String s = "hello";
      StringBuilder sb = new StringBuilder( "hello" );
      sb.reverse();
      s.reverse();
      if( s == sb.toString() )  System.out.println( "Equal" );
      else System.out.println( "Not Equal" );
   }
}
```

Select 1 option

A. Compilation error.

B. It will print 'Equal'.

C. It will print 'Not Equal'.

D. Runtime error.

E. None of the above.

Check Answer

03. QID - 2.1155

Which line will print the string "MUM"?

```
public class TestClass{
    public static void main(String args []){
        String s = "MINIMUM";
        System.out.println(s.substring(4, 7)); //1
        System.out.println(s.substring(5)); //2
        System.out.println(s.substring(s.indexOf('I', 3))); //3
        System.out.println(s.substring(s.indexOf('I', 4))); //4
    }
}
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. None of these.

Check Answer

04. QID - 2.956

Consider the following class...

```
class MyString extends String{
   MyString(){ super(); }
}
```

The above code will not compile.

Select 1 option

A. True

B. False

Check Answer

05. QID - 2.1246

What will the following statement return?

```
"    hello java guru   ".trim();
```

Select 1 option

A. The line of code will not compile.

B. `"hellojavaguru"`

C. `"hello java guru"`

D. `"hello java guru   "`

E. None of the above

Check Answer

06. QID - 2.868

How can you initialize a StringBuilder to have a capacity of at least 100 characters?

Select 2 options

A.
```
StringBuilder sb = new StringBuilder(100);
```

B.
```
StringBuilder sb = StringBuilder.getInstance(100);
```

C.
```
StringBuilder sb = new StringBuilder();
sb.setCapacity(100);
```

D.
```
StringBuilder sb = new StringBuilder();
sb.ensureCapacity(100);
```

Check Answer

07. QID - 2.1192

What will be the output of the following lines ?

```
System.out.println("" +5 + 6);    //1
System.out.println(5 + "" +6);    // 2
System.out.println(5 + 6 +"");    // 3
System.out.println(5 + 6);        // 4
```

Select 1 option

A. 56, 56, 11, 11

B. 11, 56, 11, 11

C. 56, 56, 56, 11

D. 56, 56, 56, 56

E. 56, 56, 11, 56

Check Answer

08. QID - 2.1303

Consider following classes:

```
//In File Other.java
package other;
public class Other { public static String hello = "Hello"; }

//In File Test.java
package testPackage;
import other.*;
class Test{
   public static void main(String[] args){
      String hello = "Hello", lo = "lo";
      System.out.print((testPackage.Other.hello == hello) + " ");
      System.out.print((other.Other.hello == hello) + " ");   //line
      System.out.print((hello == ("Hel"+"lo")) + " ");           //l
      System.out.print((hello == ("Hel"+lo)) + " ");              //
      System.out.println(hello == ("Hel"+lo).intern());          //l
   }
}
class Other { static String hello = "Hello"; }
```

What will be the output of running class Test?

Select 1 option

A. false false true false true

B. false true true false true

C. true true true true true

D. true true true false true

E. None of the above.

Check Answer

09. QID - 2.1186

What will the following code print when compiled and run?

```
public class TestClass {
  public static void main(String[] args) {

    String s = "blooper";
    StringBuilder sb = new StringBuilder(s);
    s.append("whopper");
    sb.append("shopper");

    System.out.println(s);
    System.out.println(sb);
  }
}
```

Select 1 option

A. `blooper` and `bloopershopper`

B. `blooperwhopper` and `bloopershopper`

C. `blooper` and `blooperwhoppershopper`

D. It will not compile.

Check Answer

10. QID - 2.1284

What will the following class print when run?

```
public class Sample{
   public static void main(String[] args)  {
     String s1 = new String("java");
     StringBuilder s2 = new StringBuilder("java");
     replaceString(s1);
     replaceStringBuilder(s2);
     System.out.println(s1 + s2);
  }
  static void replaceString(String s) {
     s = s.replace('j', 'l');
  }
  static void replaceStringBuilder(StringBuilder s) {
     s.append("c");
  }
}
```

Select 1 option

A. `javajava`

B. `lavajava`

C. `javajavac`

D. `lavajavac`

E. None of these.

Check Answer

11. QID - 2.1302

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      StringBuilder sb = new StringBuilder("12345678");
      sb.setLength(5);
      sb.setLength(10);
      System.out.println(sb.length());
   }
}
```

Select 1 option

A. It will print 5.

B. It will print 10.

C. It will print 8.

D. Compilation error.

E. None of the above.

Check Answer

12. QID - 2.1351

Which of the following methods modify the object on which they are called?

Select 1 option

A. `setValue(int)` of java.lang.Integer class.

B. The `substring(int)` method of the String class

C. The `replace()` method of the String class.

D. The `reverse()` method of the StringBuffer class.

E. None of these.

Check Answer

13. QID - 2.1054

What will be written to the standard output when the following program is run?

```
public class TrimTest{
   public static void main(String args[]){
      String blank  = " ";  // one space
      String line = blank + "hello" + blank + blank;
      line.concat("world");
      String newLine  =  line.trim();
      System.out.println((int)(line.length() + newLine.length()));
   }
}
```

Select 1 option

A. 25

B. 24

C. 23

D. 22

E. None of the above.

Check Answer

14. QID - 2.999

Which of the following method calls can be applied to a String object?

Select 3 options

A. `equals(Object)`

B. `equalsIgnoreCase(String)`

C. `prune()`

D. `append()`

E. `intern()`

Check Answer

15. QID - 2.988

What will the following code print?

```
public class Test{
    public static void stringTest(String s){
        s.replace('h', 's');
    }
    public static void stringBuilderTest(StringBuilder s){
        s.append("o");
    }
    public static void main(String[] args){
        String s = "hell";
        StringBuilder sb = new StringBuilder("well");
        stringTest(s);
        stringBuilderTest(sb);
        System.out.println(s + sb);
    }
}
```

Select 1 option

A. sellwello

B. hellwello

C. hellwell

D. sellwell

E. None of these.

Check Answer

16. QID - 2.1285

Which of the following operators can be used in conjunction with a String object?

Select 3 options

A. +

B. ++

C. +=

D. .

E. *

Check Answer

17. QID - 2.1336

Which of these methods are not a part of the String class?

Select 1 option

A. `trim( )`

B. `length( )`

C. `concat(String)`

D. `hashCode( )`

E. `reverse( )`

Check Answer

18. QID - 2.935

Consider the following class...

```
class TestClass{
   int i;
   public TestClass(int i) { this.i = i;  }
   public String toString(){
       if(i == 0) return null;
       else return ""+i;
   }
   public static void main(String[ ] args){
      TestClass t1 = new TestClass(0);
      TestClass t2 = new TestClass(2);
      System.out.println(t2);
      System.out.println(""+t1);
   }
}
```

What will be the output of the following program?

Select 1 option

A. It will throw NullPointerException when run.

B. It will not compile.

C. It will print 2 and then will throw NullPointerException.

D. It will print 2 and null.

E. None of the above.

Check Answer

19. QID - 2.1174

What will the following program print?

```
public class TestClass{
  static String str = "Hello World";
  public static void changeIt(String s){
    s = "Good bye world";
  }
  public static void main(String[] args){
    changeIt(str);
    System.out.println(str);
  }
}
```

Select 1 option

A. `"Hello World"`

B. `"Good bye world"`

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

20. QID - 2.1363

In java, Strings are immutable. A direct implication of this is...

Select 2 options

A. you cannot call methods like "1234".replace('1', '9'); and expect to change the original String.

B. you cannot change a String object, once it is created.

C. you can change a String object only by the means of its methods.

D. you cannot extend String class.

E. you cannot compare String objects.

Check Answer

21. QID - 2.1109

Which of the following statements are true?

Select 2 options

A. method `length()` of String class is a final method.

B. You can make mutable subclasses of the String class.

C. `StringBuilder` extends `String`.

D. StringBuilder is a final class.

E. String class is not final.

Check Answer

22. QID - 2.852

What will the following code print?

```
System.out.println("12345".charAt(6));
```

Select 1 option

A. `5`

B. `null`

C. `-1`

D. It will throw an `ArrayIndexOutOfBoundsException`.

E. It will throw a `StringOutOfBoundsException`.

F. It will throw an `IndexOutOfBoundsException`

Check Answer

23. QID - 2.989

Consider the following code:

```
public class Logger{
    private StringBuilder sb = new StringBuilder();

    public void logMsg(String location, String message){
    sb.append(location);
    sb.append("-");
    sb.append(message);
    }

    public void dumpLog(){
    System.out.println(sb.toString());
    //Empty the contents of sb here
    }

}
```

Which of the following options will empty the contents of the StringBuilder referred to by variable sb in method dumpLog()?

Select 1 option

A. `sb.delete(0, sb.length());`

B. `sb.clear();`

C. `sb.empty();`

D. `sb.removeAll();`

E. `sb.deleteAll();`

Check Answer

24. QID - 2.941

What will the following code print?

```
String abc = "";
abc.concat("abc");
abc.concat("def");
System.out.print(abc);
```

Select 1 option

A. abc

B. abcdef

C. def

D. It will print empty string (or in other words, nothing).

E. It will not compile because there is no concat() method in String class.

Check Answer

25. QID - 2.1187

Which of the following methods can be called on a String object?

Select 3 options

A. `substring(int i)`

B. `substring(int i, int j)`

C. `substring(int i, int j, int k)`

D. `equals(Object o)`

Check Answer

26. QID - 2.1304

Which of these are not part of the StringBuilder class?

Select 1 option

A. `trim( )`

B. `ensureCapacity(int )`

C. `append(boolean)`

D. `reverse( )`

E. `setLength(int)`

Check Answer

27. QID - 2.968

Which of these expressions will obtain the substring "456" from a string defined by `String str = "01234567"`?

Select 1 option

A. `str.substring(4, 7)`

B. `str.substring(4)`

C. `str.substring(3, 6)`

D. `str.substring(4, 6)`

E. `str.substring(4, 3)`

Check Answer

28. QID - 2.1248

Which of these are valid expressions to create a string of value "hello world" ?

Select 4 options

A. `" hello world".trim()`

B. `("hello" + " world")`

C. `(new String("hello") + " world")`

D. `("hello" + new String("world"))`

E. `"hello".concat(" world")`

Check Answer

29. QID - 2.1274

What will be the result of attempting to compile and run the following code?

```
class TestClass{
   public static void main(String args[] ){
      String str1 = "str1";
      String str2 = "str2";
      System.out.println( str1.concat(str2) );
      System.out.println(str1);
   }
}
```

Select 1 option

A. The code will fail to compile.

B. The program will print `str1` and `str1`.

C. The program will print `str1` and `str1str2`

D. The program will print `str1str2` and `str1`

E. The program will print `str1str2` and `str1str2`.

Check Answer

30. QID - 2.1184

Given:

```
StringBuilder b1 = new StringBuilder("snorkler");
StringBuilder b2 = new StringBuilder("yoodler");
```

Write the contents of b1 and b2 after the statements shown on the left are executed independent of each other.

Statements	Contents of b1	Contents of b2
`b1.append(b2.substring(2, 5) .toUpperCase());`		
`b2.insert(3, b1.append("a"));`		
`b1.replace(3, 4, b2.substring(4)) .append(b2.append(false));`		

Check Answer

31. QID - 2.1152

Which of these expressions will return true?

Select 4 options

A. `"hello world".equals("hello world")`

B. `"HELLO world".equalsIgnoreCase("hello world")`

C. `"hello".concat(" world").trim().equals("hello world")`

D. `"hello world".compareTo("Hello world") < 0`

E. `"Hello world".toLowerCase( ).equals("hello world")`

Check Answer

32. QID - 2.1225

Which of these statements concerning the `charAt()` method of the `String` class are true?

Select 2 options

A. The `charAt( )` method can take a char value as an argument.

B. The `charAt( )` method returns a Character object.

C. The expression `char ch = "12345".charAt(3)` will assign 3 to ch.

D. The expression `char ch = str.charAt(str.length())` where str is "12345", will assign 3 to ch.

E. The index of the first character is 0.

F. It throws `StringIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

G. It throws `ArrayIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

Check Answer

Working with Java Data Types - String, StringBuilder (Answered)

01. QID - 2.1022 : Working with Java Data Types - String, StringBuilder

What will be the output of the following program (excluding the quotes)?

```
public class SubstringTest{
   public static void main(String args[]){
      String String = "string isa string";
      System.out.println(String.substring(3, 6));
   }
}
```

Correct Option is : E

~~**A.**~~ It will not compile.

> String String = "String"; is a perfectly valid syntax!

~~**B.**~~ "ing is"

~~**C.**~~ "ing isa"

~~**D.**~~ "ing " (There is a space after g)

E. None of the above.

> It will print 'ing'. (No space after 'g')

Explanation:

Remember, indexing always starts from 0.
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"

Parameters:
beginIndex - the beginning index, inclusive.
endIndex - the ending index, exclusive.
Returns:
the specified substring.
Throws:
IndexOutOfBoundsException - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

Back to Question without Answer

02. QID - 2.1021 : Working with Java Data Types - String, StringBuilder

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      String s = "hello";
      StringBuilder sb = new StringBuilder( "hello" );
      sb.reverse();
      s.reverse();
      if( s == sb.toString() )  System.out.println( "Equal" );
      else System.out.println( "Not Equal" );
   }
}
```

Correct Option is : A

A. Compilation error.

There is no reverse() method in String class.

~~**B.**~~ It will print 'Equal'.

~~**C.**~~ It will print 'Not Equal'.

~~**D.**~~ Runtime error.

~~**E.**~~ None of the above.

Back to Question without Answer

03. QID - 2.1155 : Working with Java Data Types - String, StringBuilder

Which line will print the string "MUM"?

```
public class TestClass{
    public static void main(String args []){
        String s = "MINIMUM";
        System.out.println(s.substring(4, 7)); //1
        System.out.println(s.substring(5)); //2
        System.out.println(s.substring(s.indexOf('I', 3))); //3
        System.out.println(s.substring(s.indexOf('I', 4))); //4
    }
}
```

Correct Option is : A

A. 1

~~B.~~ 2

It will print UM.

~~C.~~ 3

It will print IMUM. as s.indexOf('I', 3) will return 3.

~~D.~~ 4

It will throw an exception as s.indexOf('I', 4) will return -1.

~~E.~~ None of these.

Explanation:

You should know how substring and indexOf methods of String class work.

String substring(int beginIndex)

Returns a new string that is a substring of this string.

String substring(int beginIndex, int endIndex)

Returns a new string that is a substring of this string.

int indexOf(int ch)

Returns the index within this string of the first occurrence of the specified character.

int indexOf(int ch, int fromIndex)

Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

int indexOf(String str)

Returns the index within this string of the first occurrence of the specified substring.

int indexOf(String str, int fromIndex)

Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

Back to Question without Answer

04. QID - 2.956 : Working with Java Data Types - String, StringBuilder

Consider the following class...

```
class MyString extends String{
   MyString(){ super(); }
}
```

The above code will not compile.

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

This will not compile because `String` is a final class and final classes cannot be extended.
There are questions on this aspect in the exam and so you should remember that `StringBuffer` and `StringBuilder` are also final. All Primitive wrappers are also final (i.e. `Boolean, Integer, Byte` etc).
`java.lang.System` is also final.

Back to Question without Answer

05. QID - 2.1246 : Working with Java Data Types - String, StringBuilder

What will the following statement return?

```
"    hello java guru   ".trim();
```

Correct Option is : C

~~**A.**~~ The line of code will not compile.

> `"    hello java guru   "` is a valid String and `trim()` is a valid method in String class.

~~**B.**~~ `"hellojavaguru"`

> `trim()` does not remove spaces in within the string but the spaces at the beginning and at the end.

C. `"hello java guru"`

~~**D.**~~ `"hello java guru   "`

> It returns a string in which both the leading and trailing white space of the original string are removed.

~~**E.**~~ None of the above

Back to Question without Answer

06. QID - 2.868 : Working with Java Data Types - String, StringBuilder

How can you initialize a StringBuilder to have a capacity of at least 100 characters?

Correct Options are : A D

A. `StringBuilder sb = new StringBuilder(100);`

> public StringBuilder(int capacity)
> Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.

~~**B.**~~ `StringBuilder sb = StringBuilder.getInstance(100);`

~~**C.**~~
```
StringBuilder sb = new StringBuilder();
sb.setCapacity(100);
```

D.
```
StringBuilder sb = new StringBuilder();
sb.ensureCapacity(100);
```

> public void ensureCapacity(int minimumCapacity)
> Ensures that the capacity is at least equal to the specified minimum. If the current capacity is less than the argument, then a new internal array is allocated with greater capacity. The new capacity is the larger of:
> The minimumCapacity argument.
> Twice the old capacity, plus 2.
> If the minimumCapacity argument is nonpositive, this method takes no action and simply returns.

Explanation:

Observe that the question says "at least 100 characters". In the exam, you may get a question that says "100 characters", in that case, ensureCapacity() may not be a valid option.

Back to Question without Answer

07. QID - 2.1192 : Working with Java Data Types - String, StringBuilder

What will be the output of the following lines ?

```
System.out.println("" +5 + 6);    //1
System.out.println(5 + "" +6);    // 2
System.out.println(5 + 6 +"");    // 3
System.out.println(5 + 6);        // 4
```

Correct Option is : A

A. 56, 56, 11, 11

~~**B.**~~ 11, 56, 11, 11

~~**C.**~~ 56, 56, 56, 11

~~**D.**~~ 56, 56, 56, 56

~~**E.**~~ 56, 56, 11, 56

Explanation:

In line 1, `"" + 5 + 6 => "5"+6 => "56"`
In line 2, `5 + "" +6  => "5"+6 => "56"`
In line 3, `5 + 6 +"" => 11+"" => "11"`
In line 4, `5 + 6 => 11 => "11"`

Back to Question without Answer

08. QID - 2.1303 : Working with Java Data Types - String, StringBuilder

Consider following classes:

```
//In File Other.java
package other;
public class Other { public static String hello = "Hello"; }

//In File Test.java
package testPackage;
import other.*;
class Test{
   public static void main(String[] args){
      String hello = "Hello", lo = "lo";
      System.out.print((testPackage.Other.hello == hello) + " ");
      System.out.print((other.Other.hello == hello) + " ");   //line
      System.out.print((hello == ("Hel"+"lo")) + " ");           //l
      System.out.print((hello == ("Hel"+lo)) + " ");              //
      System.out.println(hello == ("Hel"+lo).intern());          //l
   }
}
class Other { static String hello = "Hello"; }
```

What will be the output of running class Test?

Correct Option is : D

~~**A.**~~ false false true false true

~~**B.**~~ false true true false true

~~**C.**~~ true true true true true

D. true true true false true

~~**E.**~~ None of the above.

Explanation:

These are the six facts on Strings:
1. Literal strings within the same class in the same package represent references to the same String object.
2. Literal strings within different classes in the same package represent references to the same String object.
3. Literal strings within different classes in different packages likewise represent references to the same String object.
4. Strings computed by constant expressions are computed at compile time and then treated as if they were literals.
5. Strings computed at run time are newly created and therefore are distinct. (So line 4 prints false.)
6. The result of explicitly interning a computed string is the same string as any pre-existing literal string with the same contents. (So line 5 prints true.)

We advise you to read section 3.10.5 String Literals in Java Language Specification.

Back to Question without Answer

09. QID - 2.1186 : Working with Java Data Types - String, StringBuilder

What will the following code print when compiled and run?

```
public class TestClass {
  public static void main(String[] args) {

    String s = "blooper";
    StringBuilder sb = new StringBuilder(s);
    s.append("whopper");
    sb.append("shopper");

    System.out.println(s);
    System.out.println(sb);
  }
}
```

Correct Option is : D

~~**A.**~~ `blooper` and `bloopershopper`

~~**B.**~~ `blooperwhopper` and `bloopershopper`

~~**C.**~~ `blooper` and `blooperwhoppershopper`

D. It will not compile.

`append()` method does not exist in `String` class. It exits only in `StringBuffer` and `StringBuilder`. The value of sb will be `bloopershopper` though.

Back to Question without Answer

10. QID - 2.1284 : Working with Java Data Types - String, StringBuilder

What will the following class print when run?

```
public class Sample{
   public static void main(String[] args)  {
     String s1 = new String("java");
     StringBuilder s2 = new StringBuilder("java");
     replaceString(s1);
     replaceStringBuilder(s2);
     System.out.println(s1 + s2);
  }
  static void replaceString(String s) {
     s = s.replace('j', 'l');
  }
  static void replaceStringBuilder(StringBuilder s) {
     s.append("c");
  }
}
```

Correct Option is : C

A. ~~javajava~~

B. ~~lavajava~~

C. javajavac

D. ~~lavajavac~~

E. ~~None of these.~~

Explanation:

String is immutable while StringBuilder is not. So no matter what you do in `replaceString()` method, the original String that was passed to it will not change. On the other hand, StringBuilder methods, such as replace or append, change the StringBuilder itself. So, 'c' is appended to java in `replaceStringBuilder()` method.

Back to Question without Answer

11. QID - 2.1302 : Working with Java Data Types - String, StringBuilder

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[ ] ){
      StringBuilder sb = new StringBuilder("12345678");
      sb.setLength(5);
      sb.setLength(10);
      System.out.println(sb.length());
   }
}
```

Correct Option is : B

~~**A.**~~ It will print 5.

> Although it truncates the string to length 5 but setLength(10) will append 5 spaces (actually null chars i.e. \u0000).

B. It will print 10.

~~**C.**~~ It will print 8.

~~**D.**~~ Compilation error.

~~**E.**~~ None of the above.

> The program will compile without error and will print 10 when run.

Explanation:

If you do `System.out.println(sb);` it will indeed print 12345 but the length will be 10.

From javadocs:

public void setLength(int newLength)
Sets the length of the character sequence. The sequence is changed to a new character sequence whose length is specified by the argument. For every nonnegative index k less than newLength, the character at index k in the new character sequence is the same as the character at index k in the old sequence if k is less than the length of the old character sequence; otherwise, it is the null character '\u0000'. In other words, if the newLength argument is less than the current length, the length is changed to the specified length.
If the newLength argument is greater than or equal to the current length, sufficient null characters ('\u0000') are appended so that length becomes the newLength argument.

The newLength argument must be greater than or equal to 0.

Parameters:
newLength - the new length
Throws:
IndexOutOfBoundsException - if the newLength argument is negative.

Back to Question without Answer

12. QID - 2.1351 : Working with Java Data Types - String, StringBuilder

Which of the following methods modify the object on which they are called?

Correct Option is : D

~~**A.**~~ `setValue(int)` of java.lang.Integer class.

> There is no such method in Integer class. Note that Integer, Float and other such wrapper objects are immutable.

~~**B.**~~ The `substring(int)` method of the String class

> String is an immutable object. calling substring(...) returns a new different String object. It cannot change the original object.

~~**C.**~~ The `replace()` method of the String class.

> String objects can never be modified once created.

D. The `reverse()` method of the StringBuffer class.

~~**E.**~~ None of these.

Back to Question without Answer

13. QID - 2.1054 : Working with Java Data Types - String, StringBuilder

What will be written to the standard output when the following program is run?

```
public class TrimTest{
   public static void main(String args[]){
      String blank  = " ";  // one space
      String line = blank + "hello" + blank + blank;
      line.concat("world");
      String newLine  =  line.trim();
      System.out.println((int)(line.length() + newLine.length()));
   }
}
```

Correct Option is : E

~~**A.**~~ 25

~~**B.**~~ 24

~~**C.**~~ 23

~~**D.**~~ 22

E. None of the above.

It will print 13 !!!

Explanation:

Note that line.concat("world") does not change line itself. It creates a new String object containing " hello world " but it is lost because there is no reference to it.

Similarly, calling trim() does not change the object itself.
So the answer is 8 + 5 = 13 !

Back to Question without Answer

14. QID - 2.999 : Working with Java Data Types - String, StringBuilder

Which of the following method calls can be applied to a String object?

Correct Options are : A B E

A. `equals(Object)`

B. `equalsIgnoreCase(String)`

~~C.~~ `prune()`

There is no such method.

~~D.~~ `append()`

This method is in StringBuffer and StringBuilder but not in String.

E. `intern()`

Explanation:

public String intern()
Returns a canonical representation for the string object.
A pool of strings, initially empty, is maintained privately by the class String.
When the intern method is invoked, if the pool already contains a string equal to this String object as determined by the equals(Object) method, then the string from the pool is returned. Otherwise, this String object is added to the pool and a reference to this String object is returned.
It follows that for any two strings s and t, s.intern() == t.intern() is true if and only if s.equals(t) is true.

All literal strings and string-valued constant expressions are interned. String literals are defined in 3.10.5 of the Java Language Specification

Returns:

a string that has the same contents as this string, but is guaranteed to be from a pool of unique strings.

Back to Question without Answer

15. QID - 2.988 : Working with Java Data Types - String, StringBuilder

What will the following code print?

```
public class Test{
    public static void stringTest(String s){
        s.replace('h', 's');
    }
    public static void stringBuilderTest(StringBuilder s){
        s.append("o");
    }
    public static void main(String[] args){
        String s = "hell";
        StringBuilder sb = new StringBuilder("well");
        stringTest(s);
        stringBuilderTest(sb);
        System.out.println(s + sb);
    }
}
```

Correct Option is : B

~~**A.**~~ sellwello

B. hellwello

~~**C.**~~ hellwell

~~**D.**~~ sellwell

~~**E.**~~ None of these.

Explanation:

A String is immutable while a StringBuilder is not. So in `stringTest()`, `"hell".replace('h', 's')` will produce a new String `"sell"` but will not affect the original String that was passed to the method.
However, the `append()` method of `StringBuilder` appends to the original String object. So, `"well"` becomes `"wello"`.

Back to Question without Answer

16. QID - 2.1285 : Working with Java Data Types - String, StringBuilder

Which of the following operators can be used in conjunction with a String object?

Correct Options are : A C D

A. +

~~**B.** ++~~

C. +=

D. .

~~**E.** *~~

Explanation:

Only + is overloaded for String. a+=x is actually converted to a = a + x. so it is valid for Strings. dot (.) operator accesses members of the String object. There is only one member variable though: CASE_INSENSITIVE_ORDER. It is of type Comparator (which is an interface).

Back to Question without Answer

17. QID - 2.1336 : Working with Java Data Types - String, StringBuilder

Which of these methods are not a part of the String class?

Correct Option is : E

~~**A.**~~ `trim( )`

~~**B.**~~ `length( )`

~~**C.**~~ `concat(String)`

~~**D.**~~ `hashCode( )`

E. `reverse( )`

The String class has no `reverse( )` method but `StringBuffer` (and `StringBuilder`) do have this method.

Back to Question without Answer

18. QID - 2.935 : Working with Java Data Types - String, StringBuilder

Consider the following class...

```
class TestClass{
   int i;
   public TestClass(int i) { this.i = i;  }
   public String toString(){
       if(i == 0) return null;
       else return ""+i;
   }
   public static void main(String[ ] args){
      TestClass t1 = new TestClass(0);
      TestClass t2 = new TestClass(2);
      System.out.println(t2);
      System.out.println(""+t1);
   }
}
```

What will be the output of the following program?

Correct Option is : D

~~**A.**~~ It will throw NullPointerException when run.

~~**B.**~~ It will not compile.

~~**C.**~~ It will print 2 and then will throw NullPointerException.

D. It will print 2 and null.

~~**E.**~~ None of the above.

Explanation:

The method `print()/println()` of `OutputStream` takes an `Object` and prints out a String that is returned by calling `toString()` on that object. Note that as `toString()` is defined in Object class, all objects in java have this method. So it prints 2 first.

The second object's `toString()` returns `null`, so it prints "`null`". There is no `NullPointerException` because no method is called on `null`.

Now, the other feature of `print/println` methods is that if they get `null` as input parameter, they print "`null`". They do not try to call `toString()` on `null`.

So, if you have, `Object o = null; System.out.println(o);` will print `null` and will not throw a `NullPointerException`.

Back to Question without Answer

19. QID - 2.1174 : Working with Java Data Types - String, StringBuilder

What will the following program print?

```
public class TestClass{
  static String str = "Hello World";
  public static void changeIt(String s){
    s = "Good bye world";
  }
  public static void main(String[] args){
    changeIt(str);
    System.out.println(str);
  }
}
```

Correct Option is : A

A. `"Hello World"`

~~**B.**~~ `"Good bye world"`

~~**C.**~~ It will not compile.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

Theoretically, java supports Pass by Value for everything (i.e. primitives as well as Objects).

. Primitives are always passed by value.
. Object "references" are passed by value. So it looks like the object is passed by reference but actually it is the value of the reference that is passed.

An example:

```
    Object o1 = new Object(); //Let us say, the object is stored at
memory location 15000.
        //Since o1 actually stores the address of the memory
location where the object is stored, it contains 15000.
```

Now, when you call someMethod(o1); the value 15000 is passed to the method.

Inside the method someMethod():

```
    someMethod( Object localVar) {
            /*localVar now contains 15000, which means it also
points to the same memory location where the object is stored.
            Therefore, when you call a method on localVar, it will
be executed on the same object.
            However, when you change the value of localVar itself,
for example if you do localVar=null,
            it then it starts pointing to a different memory
location. But the original variable o1 still
            contains 15000 so it still points to the same object. */
        }
```

This is what happens in the this question.
In the method changeIt(...) you are giving a new value to the local variable but the original reference remains the same.
If you need even more detailed explanation, please check
http://www.javaranch.com/campfire/StoryPassBy.jsp

Back to Question without Answer

20. QID - 2.1363 : Working with Java Data Types - String, StringBuilder

In java, Strings are immutable. A direct implication of this is...

Correct Options are : A B

A. you cannot call methods like "1234".replace('1', '9'); and expect to change the original String.

> calling such methods do not change this object. They create a new String object.

B. you cannot change a String object, once it is created.

~~**C.**~~ you can change a String object only by the means of its methods.

~~**D.**~~ you cannot extend String class.

> That's because it is final, not because it is immutable. You can have a final class whose objects are mutable.

~~**E.**~~ you cannot compare String objects.

> String class implements Comparable interface.

Back to Question without Answer

21. QID - 2.1109 : Working with Java Data Types - String, StringBuilder

Which of the following statements are true?

Correct Options are : A D

A. method `length()` of String class is a final method.

> Actually, String class itself is final and so all of its methods are implicitly final.

~~**B.**~~ You can make mutable subclasses of the String class.

> Both - `String` and `StringBuilder` are final classes. So is `StringBuffer`.

~~**C.**~~ `StringBuilder` extends `String`.

> `StringBuilder` extends `Object`

D. StringBuilder is a final class.

> String, StringBuilder, and StringBuffer - all are final classes.
>
> 1. Remember that wrapper classes (java.lang.Boolean, java.lang.Integer, java.lang.Long, java.lang.Short etc.) are also final and so they cannot be extended.
>
> 2. java.lang.Number, however, is not final. Integer, Long, Double etc. extend Number.
>
> 3. java.lang.System is final as well.

~~**E.**~~ String class is not final.

Back to Question without Answer

22. QID - 2.852 : Working with Java Data Types - String, StringBuilder

What will the following code print?

```
System.out.println("12345".charAt(6));
```

Correct Option is : F

~~**A.**~~ `5`

~~**B.**~~ `null`

~~**C.**~~ `-1`

~~**D.**~~ It will throw an `ArrayIndexOutOfBoundsException`.

~~**E.**~~ It will throw a `StringOutOfBoundsException`.

> There is no such exception. The correct name is `StringIndexOutOfBoundsException`. But that is also not the correct answer.

F. It will throw an `IndexOutOfBoundsException`

> As per the API documentation of this method (http://docs.oracle.com/javase/6/docs/api/java/lang/String.html#charAt(int)), this method throws `IndexOutOfBoundsException`. Although, in practice the method throws `StringIndexOutOfBoundsException`, which is a subclass of `IndexOutOfBoundsException`, the implementation is free to throw some other subclass of `IndexOutOfBoundsException`. Thus, you should rely only on the

published API documentation instead of what it actually throws.

Explanation:

Since indexing starts with 0, the maximum value that you can pass to charAt is length-1.

As per the API documentation for `charAt`, it throws `IndexOutOfBoundsException` if you pass an invalid value (that is, if the index argument is negative or not less than the length of this string).

Both - `ArrayIndexOutOfBoundsException` and `StringIndexOutOfBoundsException`, extend `IndexOutOfBoundsException` and although in practice, the `charAt` method throws `StringIndexOutOfBoundsException`, it is not a valid option because the implementation is free to throw some other exception as long as it is an `IndexOutOfBoundsException`.

(There are questions on the exam on this aspect.)

Back to Question without Answer

23. QID - 2.989 : Working with Java Data Types - String, StringBuilder

Consider the following code:

```
public class Logger{
    private StringBuilder sb = new StringBuilder();

    public void logMsg(String location, String message){
    sb.append(location);
    sb.append("-");
    sb.append(message);
    }

    public void dumpLog(){
    System.out.println(sb.toString());
    //Empty the contents of sb here
    }

}
```

Which of the following options will empty the contents of the StringBuilder referred to by variable sb in method dumpLog()?

Correct Option is : A

A. `sb.delete(0, sb.length());`

~~**B.**~~ `sb.clear();`

~~**C.**~~ `sb.empty();`

~~**D.**~~ `sb.removeAll();`

~~**E.**~~ `sb.deleteAll();`

Explanation:

`public StringBuilder delete(int start, int end)`
Removes the characters in a substring of this sequence. The substring begins at the specified start and extends to the character at index end - 1 or to the end of the sequence if no such character exists. If start is equal to end, no changes are made.
Parameters:
start - The beginning index, inclusive.
end - The ending index, exclusive.
Returns:
This object.
Throws:
StringIndexOutOfBoundsException - if start is negative, greater than length(), or greater than end.

Back to Question without Answer

24. QID - 2.941 : Working with Java Data Types - String, StringBuilder

What will the following code print?

```
String abc = "";
abc.concat("abc");
abc.concat("def");
System.out.print(abc);
```

Correct Option is : D

~~**A.**~~ abc

~~**B.**~~ abcdef

~~**C.**~~ def

D. It will print empty string (or in other words, nothing).

~~**E.**~~ It will not compile because there is no concat() method in String class.

Explanation:

Strings are immutable so doing abc.concat("abc") will create a new string "abc" but will not affect the original string "".

Back to Question without Answer

25. QID - 2.1187 : Working with Java Data Types - String, StringBuilder

Which of the following methods can be called on a String object?

Correct Options are : A B D

A. `substring(int i)`

returns substring starting from i to end.

B. `substring(int i, int j)`

returns substring starting from i to j-1.

C. `substring(int i, int j, int k)`

D. `equals(Object o)`

Since Object class has this method, every java class inherits it.

Back to Question without Answer

26. QID - 2.1304 : Working with Java Data Types - String, StringBuilder

Which of these are not part of the StringBuilder class?

Correct Option is : A

A. `trim( )`

> This method is in String class.

B. `ensureCapacity(int )`

> Ensures that the capacity of the buffer is at least equal to the specified minimum.

C. `append(boolean)`

> It has all sorts of overloaded append methods !!!

D. `reverse( )`

E. `setLength(int)`

> Sets the length of this String buffer. This string buffer is altered to represent a new character sequence whose length is specified by the argument. For every nonnegative index k less than newLength, the character at index k in the new character sequence is the same as the character at index k in the old sequence if k is less than the length of the old character sequence; otherwise, it is the null character '' (\u0000). In other words, if the newLength argument is less than the current length of the string buffer, the string buffer is truncated to contain exactly the number of characters given by the newLength argument.
> If the newLength argument is greater than or equal to the current length, sufficient

null characters ('\u0000') are appended to the string buffer so that length becomes the newLength argument.
The newLength argument must be greater than or equal to 0.
Parameters:
newLength - the new length of the buffer.
Throws:
IndexOutOfBoundsException - if the newLength argument is negative.

Back to Question without Answer

27. QID - 2.968 : Working with Java Data Types - String, StringBuilder

Which of these expressions will obtain the substring "456" from a string defined by `String str = "01234567"`?

Correct Option is : A

A. `str.substring(4, 7)`

~~B.~~ `str.substring(4)`

It will return "4567".

~~C.~~ `str.substring(3, 6)`

It will return "345".

~~D.~~ `str.substring(4, 6)`

It will return "45".

~~E.~~ `str.substring(4, 3)`

Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: -1

Explanation:

Read this carefully:
public String substring(int beginIndex, int endIndex)
Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the

length of the substring is endIndex-beginIndex.

"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
"unhappy".substring(2) returns "happy"
"Harbison".substring(3) returns "bison"
"emptiness".substring(9) returns "" (an empty string)

Back to Question without Answer

28. QID - 2.1248 : Working with Java Data Types - String, StringBuilder

Which of these are valid expressions to create a string of value "hello world" ?

Correct Options are : A B C E

A. `" hello world".trim()`

> trim() removes starting and ending spaces.

B. `("hello" + " world")`

> operator + is overloaded for Strings.

C. `(new String("hello") + " world")`

> This will create "hello world"

~~D.~~ `("hello" + new String("world"))`

> It will create helloworld. No space between hello and world.

E. `"hello".concat(" world")`

Explanation:

All the expressions are legal. String literals are String objects and can be used just like any other object.

Back to Question without Answer

29. QID - 2.1274 : Working with Java Data Types - String, StringBuilder

What will be the result of attempting to compile and run the following code?

```
class TestClass{
   public static void main(String args[] ){
      String str1 = "str1";
      String str2 = "str2";
      System.out.println( str1.concat(str2) );
      System.out.println(str1);
   }
}
```

Correct Option is : D

~~**A.**~~ The code will fail to compile.

~~**B.**~~ The program will print `str1` and `str1`.

~~**C.**~~ The program will print `str1` and `str1str2`

D. The program will print `str1str2` and `str1`

> `str1.concat(str2)` actually creates a new object that contains `"str1str2"`. So it does not affect the object referenced by `str1`.

~~**E.**~~ The program will print `str1str2` and `str1str2`.

Explanation:

Note that String objects are immutable. No matter what operation you do, the original object will remain the same and a new object will be returned. Here, the statement

str1.concat(str2) creates a new String object which is printed but its reference is lost after the printing.

Back to Question without Answer

30. QID - 2.1184 : Working with Java Data Types - String, StringBuilder

Given:

```
StringBuilder b1 = new StringBuilder("snorkler");
StringBuilder b2 = new StringBuilder("yoodler");
```

Write the contents of b1 and b2 after the statements shown on the left are executed independent of each other.

Statements	Contents of b1	Contents of b2
`b1.append(b2.substring(2, 5) .toUpperCase());`	snorklerODL	yoodler
`b2.insert(3, b1.append("a"));`	snorklera	yoosnorkleradler
`b1.replace(3, 4, b2.substring(4)) .append(b2.append(false));`	snolerkleryoodlerfalse	yoodlerfalse

Explanation:

You need to understand how `append, insert, delete,` and `substring` methods of `StringBuilder/StringBuffer` work. Please go through JavaDoc API for these methods. This is very important for the exam. Observe that `substring()` does not modify the object it is invoked on but `append, insert` and `delete` do.

In the exam, you will find questions that use such quirky syntax, where multiple calls are chained together. For example: `sb.append("a").append("asdf").insert(2, "asdf")`. Make yourself familiar with this technique. If in doubt, just break it down into multiple calls. For example, the aforementioned statement can be thought of as:

```
sb.append("a");
sb.append("asdf");
sb.insert(2, "asdf")
```

Note that the method `substring()` in `StringBuilder/StringBuffer` returns a `String` (and not a reference to itself, unlike `append, insert,` and `delete`). So

another StringBuilder method cannot be chained to it. For example, the following is not valid: `sb.append("a").substring(0, 4).insert(2, "asdf");`

The following is valid though: `String str = sb.append("a").insert(2, "asdf").substring(0, 4);`

Back to Question without Answer

31. QID - 2.1152 : Working with Java Data Types - String, StringBuilder

Which of these expressions will return true?

Correct Options are : A B C E

A. `"hello world".equals("hello world")`

B. `"HELLO world".equalsIgnoreCase("hello world")`

equalsIgnoreCase() method treats both cases (upper and lower) as same.

C. `"hello".concat(" world").trim().equals("hello world")`

"hello".concat(" world") will return "hello world" and trim() won't do any change because there is no space at the beginning or end of the string.

D. `"hello world".compareTo("Hello world") < 0`

Notice that the Strings differ at the first position. The value returned by compareTo is (Unicode value of the left hand side - Unicode value of the right hand side).

Although not required for the exam, it is good to know that for English alphabets, the unicode value of any lower case letter is always 32 more than the unicode value of the same letter in upper case. So, 'a' - 'A' or 'h' - 'H' is 32.

E. `"Hello world".toLowerCase( ).equals("hello world")`

toLowerCase() converts all uppercase letters to lower case.

Explanation:

compareTo() does a lexicographical (like a dictionary) comparison. It stops at the first place where the strings have different letters.
If left hand side is bigger, it returns a positive number otherwise it returns a negative number. The value is equal to the difference of their unicode values.
If there is no difference then it returns zero. In this case, it will return ('h' - 'H') which is 32.

Back to Question without Answer

32. QID - 2.1225 : Working with Java Data Types - String, StringBuilder

Which of these statements concerning the `charAt()` method of the `String` class are true?

Correct Options are : A E

A. The `charAt( )` method can take a char value as an argument.

> Yes, it can because it takes an int and char will be implicitly promoted to int.

~~**B.**~~ The `charAt( )` method returns a Character object.

> It returns char.

~~**C.**~~ The expression `char ch = "12345".charAt(3)` will assign 3 to ch.

> It will assign 4 as indexing starts from 0.

~~**D.**~~ The expression `char ch = str.charAt(str.length())` where str is "12345", will assign 3 to ch.

> It will throw IndexOutOfBoundsException as `str.length()` is 5 and there is no `str.charAt(5);`

E. The index of the first character is 0.

~~**F.**~~ It throws `StringIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

~~G.~~ It throws `ArrayIndexOutOfBoundsException` if passed an value higher than or equal to the length of the string (or less than 0).

Explanation:

Since indexing starts with 0, the maximum value that you can pass to `charAt` is length-1.

As per the API documentation for `charAt`, it throws `IndexOutOfBoundsException` if you pass an invalid value.

Both - `ArrayIndexOutOfBoundsException` and `StringIndexOutOfBoundsException`, extend `IndexOutOfBoundsException` and although in practice, the `charAt` method throws `StringIndexOutOfBoundsException`, it is not a valid option because the implementation is free to throw some other exception as long as it is an `IndexOutOfBoundsException`. There are questions in the exam on this aspect.

Back to Question without Answer

Working with Java Data Types - Variables and Objects

Exam Objectives -

Declare and initialize variables Differentiate between object reference variables and primitive variables Read or write to object fields Explain an object's lifecycle Call methods on objects

01. QID - 2.835

Which of the following can be valid declarations of an integer variable?

Select 2 options

A. `global int x = 10;`

B. `final int x = 10;`

C. `public Int x = 10;`

D. `Int x = 10;`

E. `static int x = 10;`

Check Answer

02. QID - 2.892

Which of the following are valid classes?

Select 1 option

A.
```
public class ImaginaryNumber extends Number {
}
```

B.
```
public class ThreeWayBoolean extends Boolean {
}
```

C.
```
public class NewSystem extends System {
}
```

D.
```
public class ReverseString extends String {
}
```

Check Answer

03. QID - 2.1251

What will be the result of attempting to compile and run the following code?

```
public class InitClass{
   public static void main(String args[ ] ){
      InitClass obj = new InitClass(5);
   }
   int m;
   static int i1 = 5;
   static int i2 ;
   int  j = 100;
   int x;
   public InitClass(int m){
      System.out.println(i1 + "  " + i2 + "   " + x + "  " + j + "  '
   }
  { j = 30; i2 = 40; }  // Instance Initializer
   static { i1++; }      // Static Initializer
}
```

Select 1 option

A. The code will fail to compile, since the instance initializer tries to assign a value to a static member.

B. The code will fail to compile, since the member variable x will be uninitialized when it is used.

C. The code will compile without error and will print 6, 40, 0, 30, 5 when run.

D. The code will compile without error and will print 5, 0, 0, 100, 5 when run.

E. The code will compile without error and will print 5, 40, 0, 30, 0 when run.

Check Answer

04. QID - 2.1153

What will the following program print?

```
public class TestClass{
  static String str;
  public static void main(String[] args){
     System.out.println(str);
  }
}
```

Select 1 option

A. It will not compile.

B. It will compile but throw an exception at runtime.

C. It will print 'null' (without quotes).

D. It will print nothing.

E. None of the above.

Check Answer

05. QID - 2.975

In which of these variable declarations, will the variable remain uninitialized unless explicitly initialized?

Select 1 option

A. Declaration of an instance variable of type int.

B. Declaration of a static class variable of type float.

C. Declaration of a local variable of type float.

D. Declaration of a static class variable of class Object

E. Declaration of an instance variable of class Object.

Check Answer

06. QID - 2.898

Given:

```
public class Employee{
    String name;
    public Employee(){
    }
}
```

Which of the following lines creates an Employee instance?

Select 1 option

A. `Employee e;`

B. `Employee e = new Employee();`

C. `Employee e = Employee.new();`

D. `Employee e = Employee();`

Check Answer

07. QID - 2.1226

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){
      k = j++;
   }
}
```

Select 2 options

A. The class TestClass cannot be declared abstract.

B. The variable j cannot be declared transient.

C. The variable k cannot be declared synchronized.

D. The constructor TestClass( ) cannot be declared final.

E. The method f( ) cannot be declared static.

Check Answer

08. QID - 2.982

Given the following class, which of the given blocks can be inserted at line 1 without errors?

```
public class InitClass{
       private static int loop = 15 ;
       static final int INTERVAL = 10 ;
       boolean flag ;
       //line 1
}
```

Select 4 options

A. `static {System.out.println("Static"); }`

B. `static { loop = 1; }`

C. `static { loop += INTERVAL; }`

D. `static { INTERVAL = 10; }`

E. `{ flag = true; loop = 0; }`

Check Answer

09. QID - 2.1332

What happens when you try to compile and run the following program?

```
public class CastTest{
   public static void main(String args[ ] ){
      byte b = -128 ;
      int i = b ;
      b = (byte) i;
      System.out.println(i+" "+b);
   }
}
```

Select 1 option

A. The compiler will refuse to compile it because i and b are of different types cannot be assigned to each other.

B. The program will compile and will print -128 and -128 when run .

C. The compiler will refuse to compile it because -128 is outside the legal range of values for a byte.

D. The program will compile and will print 128 and -128 when run .

E. The program will compile and will print 255 and -128 when run .

Check Answer

10. QID - 2.1159

What happens when you try to compile and run the following class...

```
public class TestClass{
      public static void main(String[] args) throws Exception{
            int a = Integer.MIN_VALUE;
            int b = -a;
            System.out.println( a+ "   "+b);
      }
}
```

Select 1 option

A. It throws an `OverFlowException`.

B. It will print two same negative numbers.

C. It will print two different negative numbers.

D. It will print one negative and one positive number of same magnitude.

E. It will print one negative and one positive number of different magnitude.

Check Answer

11. QID - 2.1331

Note: This question may be considered too advanced for this exam.

Which statements can be inserted at line 1 in the following code to make the program write x on the standard output when run?

```
public class AccessTest{
   String a = "x";
   static char b = 'x';
   String  c = "x";
   class Inner{
      String  a = "y";
      String  get(){
         String c = "temp";
         // Line 1
         return c;
      }
   }

   AccessTest() {
     System.out.println(  new Inner().get()  );
   }

   public static void main(String args[]) {  new AccessTest();  }
}
```

Select 3 options

A. `c = c;`

B. `c = this.a;`

C. `c = ""+AccessTest.b;`

D. `c = AccessTest.this.a;`

E. `c = ""+b;`

Check Answer

12. QID - 2.890

Given:

```
class Square {
    private double side = 0;
    String color;
    public Square(double length){
        this.side = length;
    }
    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Square mysq = new Square(10);
        mysq.color = "red";

        //set mysq's side to 20
    }
}
```

Which of the following statements will set mysq's side to 20?

Select 1 option

A. `mysq.side = 20;`

B. `mysq = new Square(20);`

C. `mysq.setSide(20);`

D. `side = 20;`

E. `Square.mysql.side = 20;`

Check Answer

13. QID - 2.1073

Which line(s) of code in the following program will cause a compilation error?

```
public class TestClass{
   static int value = 0; //1
   public static void main(String args[]) //2
   {
      int 2ndArgument = Integer.parseInt(args[2]); //3
      if( true == 2 > 10 ) //4
      {
         value = -10;
      }
      else{
         value =  2ndArgument;
      }
      for( ; value>0; value--) System.out.println("A"); //5
   }
}
```

Select 1 option

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

14. QID - 2.1027

Note: This question may be considered too advanced for this exam. What will the following code print when run?

```
public class TestClass{
    public static Integer wiggler(Integer x){
       Integer y = x + 10;
       x++;
       System.out.println(x);
       return y;
    }

    public static void main(String[] args){
       Integer dataWrapper = new Integer(5);
       Integer value = wiggler(dataWrapper);
       System.out.println(dataWrapper+value);
    }
}
```

Select 1 option

A. 5 and 20

B. 6 and 515

C. 6 and 20

D. 6 and 615

E. It will not compile.

Check Answer

15. QID - 2.1189

Which of the following statements are acceptable?

Select 4 options

A. `Object o = new java.io.File("a.txt");`
(Assume that java.io.File is a valid class.)

B. `Boolean bool = false;`

C. `char ch = 10;`

D. `Thread t = new Runnable();`
(Assume that Runnable is a valid interface.)

E. `Runnable r = new Thread();`
(Assume that Thread is a class that implements Runnable interface)

Check Answer

16. QID - 2.1243

Which of the following are correct ways to initialize the static variables MAX and CLASS_GUID ?

```
class Widget{
   static int MAX;      //1
   static final String CLASS_GUID;    // 2
   Widget(){
       //3
   }
   Widget(int k){
       //4
   }
}
```

Select 2 options

A. Modify lines //1 and //2 as : `static int MAX = 111; static final String CLASS_GUID = "XYZ123";`

B. Add the following line just after //2 : `static {  MAX = 111; CLASS_GUID = "XYZ123"; }`

C. Add the following line just before //1 : `{ MAX = 111; CLASS_GUID = "XYZ123"; }`

D. Add the following line at //3 as well as //4 : `MAX = 111; CLASS_GUID = "XYZ123";`

E. Only option 3 is valid.

Check Answer

17. QID - 2.1126

Note: This question may be considered too advanced for this exam.
Given:

```
String mStr = "123";
long m = // 1
```

Which of the following options when put at //1 will assign `123` to `m`?

Select 3 options

A. `new Long(mStr);`

B. `Long.parseLong(mStr);`

C. `Long.longValue(mStr);`

D. `(new Long()).parseLong(mStr);`

E. `Long.valueOf(mStr).longValue();`

Check Answer

18. QID - 2.1346

Which of the following statements can be inserted at // 1 to make the code compile without errors?

```
public class InitTest{
   static int si = 10;
   int  i;
   final boolean bool;
   // 1
}
```

Select 1 option

A. `instance { bool = true; }`

B. `InitTest() { si += 10; }`

C. `{ si = 5; i = bool ? 1000 : 2000;}`

D. `{ i = 1000; }`

E. `{ bool = (si > 5); i = 1000; }`

Check Answer

19. QID - 2.917

Given:

```
public class Square {
    private double side = 0;  // LINE 2

    public static void main(String[] args) {   // LINE 4
        Square sq = new Square();  // LINE 5
        side = 10;  // LINE 6
   }
}
```

What can be done to make this code compile and run?

Select 1 option

A. replace // LINE 2 with:

```
private int side = 0;
```

B. replace // LINE 2 with:

```
public int side = 0;
```

C. replace // LINE 5 with:

```
double sq = new Square();
```

D. replace // LINE 6 with:

```
sq.side = 10;
```

Check Answer

20. QID - 2.972

Consider the following program :

```
class Test{
  public static void main(String[] args){
    short s = 10;    // 1
    char c = s;      // 2
    s = c;           // 3
  }
}
```

Identify the correct statements.

Select 2 options

A. Line 3 is not valid.

B. Line 2 is not valid.

C. It will compile because both short and char can hold 10.

D. None of the lines 1, 2 and 3 is valid.

Check Answer

21. QID - 2.865

What will the following program print when compiled and run?

```
class Data {
    private int x = 0;
    private String y = "Y";
    public Data(int k){
        this.x = k;
    }
    public Data(String k){
        this.y = k;
    }
    public void showMe(){
        System.out.println(x+y);
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        new Data(10).showMe();
        new Data("Z").showMe();
    }
}
```

Select 1 option

A. 0Z
10Y

B. 10Y
0Z

C. It will not compile.

D. It will throws an exception at run time.

Check Answer

22. QID - 2.1077

What will the following program print?

```
public class TestClass{
  static boolean b;
  static int[] ia = new int[1];
  static char ch;
  static boolean[] ba = new boolean[1];
  public static void main(String args[]) throws Exception{
    boolean x = false;
    if( b ){
      x = ( ch == ia[ch]);
    }
    else x = ( ba[ch] = b );
    System.out.println(x+" "+ba[ch]);
  }
}
```

Select 1 option

A. true true

B. true false

C. false true

D. false false

E. It will not compile.

Check Answer

23. QID - 2.1199

Given:

```
public class TestClass{
  public static int getSwitch(String str){
      return (int) Math.round( Double.parseDouble(str.substring(1, s
  }
  public static void main(String args []){
    switch(getSwitch(args[0])){
      case 0 : System.out.print("Hello ");
      case 1 : System.out.print("World"); break;
      default : System.out.print(" Good Bye");
    }
  }
}
```

What will be printed by the above code if it is run with command line:
`java TestClass --0.50`
(There are two minuses before 0.)

Select 1 option

A. Hello

B. World

C. Hello World

D. Hello World Good Bye

E. Good Bye

Check Answer

24. QID - 2.1289

Given the following statements, what will be the value of k and m after these statements are executed.

will not compile exception at runtime 12

```
int a = 5, b = 7, k = 0;
Integer m = null;
k = new Integer(a) + new Integer(b);
k = new Integer(a) + b;
k = b + new Integer(a);
m = new Integer(a) + new Integer(b);
```

Check Answer

25. QID - 2.1151

Given that TestClass is a class, how many objects and reference variables are created by the following code?

```
TestClass t1, t2, t3, t4;
t1 = t2 = new TestClass();
t3 = new TestClass();
```

Select 1 option

A. 2 objects, 3 references.

B. 2 objects, 4 references.

C. 3 objects, 2 references.

D. 2 objects, 2 references.

E. None of the above.

Check Answer

26. QID - 2.1221

Which of the following is illegal ?

Select 1 option

A. char c = 320;

B. float f = 320;

C. double d = 320;

D. byte b = 320;

E. None of the above is illegal.

Check Answer

27. QID - 2.1041

Which of the following is not a primitive data value in Java?

Select 2 options

A. `"x"`

B. `'x'`

C. `10.2F`

D. `Object`

E. `false`

Check Answer

28. QID - 2.1319

Given the following code snippet:

```
int rate = 10;
int t = 5;
XXX amount = 1000.0;
for(int i=0; i<t; t++){
   amount = amount*(1 - rate/100);
}
```

What can XXX be?

Select 1 option

A. int

B. long

C. only double

D. double or float

E. float

Check Answer

29. QID - 2.1146

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    unsigned byte b = 0;
    b--;
    System.out.println(b);
  }
}
```

Select 1 option

A. 0

B. -1

C. 255

D. -128

E. It will not compile.

Check Answer

30. QID - 2.1062

The following code snippet will print 4.

```
int i1 = 1, i2 = 2, i3 = 3;
int i4 = i1 + (i2=i3 );
System.out.println(i4);
```

Select 1 option

A. True

B. False

Check Answer

Working with Java Data Types - Variables and Objects (Answered)

01. QID - 2.835 : Working with Java Data Types - Variables and Objects

Which of the following can be valid declarations of an integer variable?

Correct Options are : B E

~~**A.**~~ `global int x = 10;`

> global is an invalid modifier. There is nothing like global in java. The closest you can get is static.

B. `final int x = 10;`

~~**C.**~~ `public Int x = 10;`

> Int with a capital I is invalid.

~~**D.**~~ `Int x = 10;`

> Int with a capital I is invalid.

E. `static int x = 10;`

Back to Question without Answer

02. QID - 2.892 : Working with Java Data Types - Variables and Objects

Which of the following are valid classes?

Correct Option is : A

A.
```
public class ImaginaryNumber extends Number {
}
```

> `Number` is not a final class so you can extend it.

~~**B.**~~
```
public class ThreeWayBoolean extends Boolean {
}
```

~~**C.**~~
```
public class NewSystem extends System {
}
```

~~**D.**~~
```
public class ReverseString extends String {
}
```

Explanation:

`String, StringBuilder, and StringBuffer` - all are final classes.

1. Remember that wrapper classes for primitives (`java.lang.Boolean, java.lang.Integer, java.lang.Long, java.lang.Short` etc.) are also final and so they cannot be extended.

2. `java.lang.Number`, however, is not final. `Integer, Long, Double` etc. extend Number.

3. `java.lang.System` is final as well.

Back to Question without Answer

03. QID - 2.1251 : Working with Java Data Types - Variables and Objects

What will be the result of attempting to compile and run the following code?

```
public class InitClass{
   public static void main(String args[ ] ){
      InitClass obj = new InitClass(5);
   }
   int m;
   static int i1 = 5;
   static int i2 ;
   int  j = 100;
   int x;
   public InitClass(int m){
      System.out.println(i1 + "  " + i2 + "   " + x + "  " + j + "  '
   }
  { j = 30; i2 = 40; }  // Instance Initializer
   static { i1++; }      // Static Initializer
}
```

Correct Option is : C

~~**A.**~~ The code will fail to compile, since the instance initializer tries to assign a value to a static member.

~~**B.**~~ The code will fail to compile, since the member variable x will be uninitialized when it is used.

C. The code will compile without error and will print 6, 40, 0, 30, 5 when run.

~~**D.**~~ The code will compile without error and will print 5, 0, 0, 100, 5 when run.

~~E.~~ The code will compile without error and will print 5, 40, 0, 30, 0 when run.

Explanation:

The value 5 is passed to the constructor to the local (automatic) variable m. So the instance variable m is shadowed. Before the body of the constructor is executed, the instance initializer is executed and assigns values 30 and 40 to variables j and i2, respectively. A class is loaded when it is first used. For example,

```
class A1{
  static int i = 10;
  static { System.out.println("A1 Loaded "); }
}
public class A{
  static { System.out.println("A Loaded "); }
  public static void main(String[] args){
    System.out.println(" A should have been loaded");
    A1 a1 = null;
    System.out.println(" A1 should not have been loaded");
    System.out.println(a1.i);
  }
}
```

When you run it you get the output:

```
A Loaded
 A should have been loaded
 A1 should not have been loaded
A1 Loaded
10
```

Now, A should be loaded first as you are using its main method. Even though you are doing `A1 a1 = null;` A1 will not be loaded as it is not yet used (so the JVM figures out that it does not need to load it yet.) When you do a1.i, you are using A1, so before you use it, it must be loaded. That's when A1 is loaded. Finally 10 is printed.

Back to Question without Answer

04. QID - 2.1153 : Working with Java Data Types - Variables and Objects

What will the following program print?

```
public class TestClass{
  static String str;
  public static void main(String[] args){
     System.out.println(str);
  }
}
```

Correct Option is : C

~~**A.**~~ It will not compile.

~~**B.**~~ It will compile but throw an exception at runtime.

C. It will print 'null' (without quotes).

~~**D.**~~ It will print nothing.

~~**E.**~~ None of the above.

Explanation:

All member fields (static and non-static) are initialized to their default values. Objects are initialized to null (String is also an object), numeric types to 0 (or 0.0) and boolean to false.

Back to Question without Answer

05. QID - 2.975 : Working with Java Data Types - Variables and Objects

In which of these variable declarations, will the variable remain uninitialized unless explicitly initialized?

Correct Option is : C

~~**A.**~~ Declaration of an instance variable of type int.

~~**B.**~~ Declaration of a static class variable of type float.

C. Declaration of a local variable of type float.

~~**D.**~~ Declaration of a static class variable of class Object

~~**E.**~~ Declaration of an instance variable of class Object.

Explanation:

We have to explicitly initialize local variables other wise they remain uninitialized and it will be a compile time error if such variables are accessed without getting initialized.
Instance variables and static variables receive a default value if not explicitly initialized. All primitive types get a defaults value equivalent to 0. (eg. int to 0 and float to 0.0f etc) and boolean to false.
The type/class of a variable does not affect whether a variable is initialized or not.

Back to Question without Answer

06. QID - 2.898 : Working with Java Data Types - Variables and Objects

Given:

```
public class Employee{
    String name;
    public Employee(){
    }
}
```

Which of the following lines creates an Employee instance?

Correct Option is : B

~~A.~~ `Employee e;`

> This declares a variable of class Employee but does not create any object.

B. `Employee e = new Employee();`

> Using the new operator is the right way to create an object.

~~C.~~ `Employee e = Employee.new();`

~~D.~~ `Employee e = Employee();`

Back to Question without Answer

07. QID - 2.1226 : Working with Java Data Types - Variables and Objects

What, if anything, is wrong with the following code?

```
abstract class TestClass{
   transient int j;
   synchronized int k;
   final void TestClass(){}
   static void f(){
      k = j++;
   }
}
```

Correct Options are : C E

~~**A.**~~ The class TestClass cannot be declared abstract.

> Any class can be declared abstract even if it does not have any abstract method.

~~**B.**~~ The variable j cannot be declared transient.

C. The variable k cannot be declared synchronized.

> Variables cannot be declared synchronized. Only methods can be declared synchronized.

~~**D.**~~ The constructor TestClass( ) cannot be declared final.

> It is not a constructor, it is a simple method. Notice void return type.

E. The method f( ) cannot be declared static.

Because it refers to instance variables j and k

Explanation:

The moment you put a return type, it ceases to be a constructor. So the compiler thinks that option 4 is a method.

FYI, constructors are not inherited, and so it doesn't make sense to mark them as final. (It is illegal to mark a constructor as final for the same reason). So there is no question of overriding them.

Static methods cannot refer to non-static/instance members (this includes fields and methods).

Back to Question without Answer

08. QID - 2.982 : Working with Java Data Types - Variables and Objects

Given the following class, which of the given blocks can be inserted at line 1 without errors?

```
public class InitClass{
        private static int loop = 15 ;
        static final int INTERVAL = 10 ;
        boolean flag ;
        //line 1
}
```

Correct Options are : A B C E

A. `static {System.out.println("Static"); }`

B. `static { loop = 1; }`

C. `static { loop += INTERVAL; }`

D. `static { INTERVAL = 10; }`

> `INTERVAL` is final and so it can never be changed after it is given a value.

E. `{ flag = true; loop = 0; }`

> `flag` is not static and so it can be accessed only from a non-static block. `loop` is static so can be accessed from any block.

Back to Question without Answer

09. QID - 2.1332 : Working with Java Data Types - Variables and Objects

What happens when you try to compile and run the following program?

```
public class CastTest{
   public static void main(String args[ ] ){
      byte b = -128 ;
      int i = b ;
      b = (byte) i;
      System.out.println(i+" "+b);
   }
}
```

Correct Option is : B

~~**A.**~~ The compiler will refuse to compile it because i and b are of different types cannot be assigned to each other.

B. The program will compile and will print -128 and -128 when run .

> A byte can ALWAYS be assigned to an int.

~~**C.**~~ The compiler will refuse to compile it because -128 is outside the legal range of values for a byte.

> Range of byte is -128 to 127

~~**D.**~~ The program will compile and will print 128 and -128 when run .

~~**E.**~~ The program will compile and will print 255 and -128 when run .

Explanation:

`byte` and `int` both hold signed values. So when `b` is assigned to `i`, the sign is preserved.

Back to Question without Answer

10. QID - 2.1159 : Working with Java Data Types - Variables and Objects

What happens when you try to compile and run the following class...

```
public class TestClass{
      public static void main(String[] args) throws Exception{
            int a = Integer.MIN_VALUE;
            int b = -a;
            System.out.println( a+ "   "+b);
      }
}
```

Correct Option is : B

~~**A.** It throws an~~ `OverFlowException`.

B. It will print two same negative numbers.

~~**C.** It will print two different negative numbers.~~

~~**D.** It will print one negative and one positive number of same magnitude.~~

~~**E.** It will print one negative and one positive number of different magnitude.~~

Explanation:

It prints: `-2147483648   -2147483648`

This happens because negative integers are stored in 2's complement form (complement the bits and add 1). For example:

Integer 1 in binary is `00000000 00000000 00000000 00000001` (32 bits)

So -1 in binary would be (complement the bits for 1 and add 1) :

Step 1 (complement the bits of 1): `11111111 11111111 11111111 11111110`
Step 2 (add 1 to step 1): `11111111 11111111 11111111 11111111`.

Now, let's see what happens in this question:

`a = Integer.MIN_VALUE = 10000000 00000000 00000000 00000000`

To get `-a`, apply the above two steps:

Step 1 (complement the bits): `011111111 11111111 11111111 11111111`
Step 2 (add 1) : `10000000 00000000 00000000 00000000`

So you got the exact same value that you started with!

(Note that you can see the binary form of an integer using `Integer.toBinaryString(i)` method.)

Back to Question without Answer

11. QID - 2.1331 : Working with Java Data Types - Variables and Objects

Note: This question may be considered too advanced for this exam.

Which statements can be inserted at line 1 in the following code to make the program write x on the standard output when run?

```
public class AccessTest{
   String a = "x";
   static char b = 'x';
   String  c = "x";
   class Inner{
      String  a = "y";
      String  get(){
         String c = "temp";
         // Line 1
         return c;
      }
   }

   AccessTest() {
     System.out.println(  new Inner().get()  );
   }

   public static void main(String args[]) {  new AccessTest();  }
}
```

Correct Options are : C D E

A. ~~`c = c;`~~

It will reassign 'temp' to c!

B. ~~`c = this.a;`~~

It will assign "y" to c.

C. `c = ""+AccessTest.b;`

Because b is static.

D. `c = AccessTest.this.a;`

E. `c = ""+b;`

Back to Question without Answer

12. QID - 2.890 : Working with Java Data Types - Variables and Objects

Given:

```
class Square {
    private double side = 0;
    String color;
    public Square(double length){
        this.side = length;
    }
    public double getSide() {  return side;    }

    public void setSide(double side) {  this.side = side;   }

}

public class TestClass {
    public static void main(String[] args) throws Exception {
        Square mysq = new Square(10);
        mysq.color = "red";

        //set mysq's side to 20
    }
}
```

Which of the following statements will set mysq's side to 20?

Correct Option is : C

~~**A.**~~ `mysq.side = 20;`

> Since side is a private variable, you cannot access it from outside Square class.

~~**B.**~~ `mysq = new Square(20);`

> This will create a new Square object.

C. `mysq.setSide(20);`

D. `side = 20;`

E. `Square.mysql.side = 20;`

Back to Question without Answer

13. QID - 2.1073 : Working with Java Data Types - Variables and Objects

Which line(s) of code in the following program will cause a compilation error?

```
public class TestClass{
   static int value = 0; //1
   public static void main(String args[]) //2
   {
      int 2ndArgument = Integer.parseInt(args[2]); //3
      if( true == 2 > 10 ) //4
      {
         value = -10;
      }
      else{
         value =  2ndArgument;
      }
      for( ; value>0; value--) System.out.println("A"); //5
   }
}
```

Correct Option is : C

~~**A.**~~ 1

~~**B.**~~ 2

C. 3

'2ndArgument' is not a valid identifier name because an identified must not start with a digit (although it can contain a digit.)

~~**D.**~~ 4

== has less precedence than > so true == 2 > 10 is same as true == (2 > 10)

~~E.~~ 5

Back to Question without Answer

14. QID - 2.1027 : Working with Java Data Types - Variables and Objects

Note: This question may be considered too advanced for this exam. What will the following code print when run?

```
public class TestClass{
    public static Integer wiggler(Integer x){
       Integer y = x + 10;
       x++;
       System.out.println(x);
       return y;
    }

    public static void main(String[] args){
       Integer dataWrapper = new Integer(5);
       Integer value = wiggler(dataWrapper);
       System.out.println(dataWrapper+value);
    }
}
```

Correct Option is : C

~~**A.**~~ 5 and 20

~~**B.**~~ 6 and 515

C. 6 and 20

~~**D.**~~ 6 and 615

~~**E.**~~ It will not compile.

Explanation:

1. Wrapper objects are always immutable. Therefore, when `dataWrapper` is passed into `wiggler()` method, it is never changed even when `x++;` is executed. However, `x`, which was pointing to the same object as `dataWrapper`, is assigned a new Integer object (different from `dataWrapper`) containing 6.

2. If both the operands of the + operator are numeric, it adds the two operands. Here, the two operands are Integer 5 and Integer 15, so it unboxes them, adds them, and prints 20.

Back to Question without Answer

15. QID - 2.1189 : Working with Java Data Types - Variables and Objects

Which of the following statements are acceptable?

Correct Options are : A B C E

A. `Object o = new java.io.File("a.txt");`
(Assume that java.io.File is a valid class.)

> This is valid because every object in Java is an Object.

B. `Boolean bool = false;`

> bool is a variable of type Boolean and not of a primitive type boolean however this is still valid because Java performs auto-boxing (and unboxing) for primitives and their wrapper types which allows false to be automatically be boxed into a Boolean false object.

C. `char ch = 10;`

> Because 10 can fit into a char.

D. ~~`Thread t = new Runnable();`~~
(Assume that Runnable is a valid interface.)

> Since Runnable is an interface, it cannot be instantiated like this. But you can do :
>
> ```
> Runnable r = new Runnable(){
>                         public void run(){ }
>                     };
> ```

E. `Runnable r = new Thread();`
(Assume that Thread is a class that implements Runnable interface)

Since Thread implements Runnable, this is a valid assignment.

Back to Question without Answer

16. QID - 2.1243 : Working with Java Data Types - Variables and Objects

Which of the following are correct ways to initialize the static variables MAX and CLASS_GUID ?

```
class Widget{
   static int MAX;      //1
   static final String CLASS_GUID;    // 2
   Widget(){
       //3
   }
   Widget(int k){
       //4
   }
}
```

Correct Options are : A B

A. Modify lines //1 and //2 as : `static int MAX = 111; static final String CLASS_GUID = "XYZ123";`

> You can initialize both the variables at declaration itself.

B. Add the following line just after //2 : `static {  MAX = 111; CLASS_GUID = "XYZ123"; }`

> Initializing the static variables in a static block ensures that they are initialized even when no instance of the class is created.

~~**C.**~~ Add the following line just before //1 : `{ MAX = 111; CLASS_GUID = "XYZ123"; }`

> This is not a static initializer and so will not be executed until an instance is created.

~~**D.**~~ Add the following line at //3 as well as //4 : `MAX = 111; CLASS_GUID = "XYZ123";`

> This works for non-static final fields but not for static final fields.

~~**E.**~~ Only option 3 is valid.

Explanation:

The rules are:
1. static variables can be left without initializing. (They will get default values).
2. final variables must be explicitly initialized.
Now, here `CLASS_GUID` is a 'static final' variable and not just final or static. As static fields can be accessed even without creating an instance of the class, it is entirely possible that this field can be accessed before even a single instance is created. In this case, no constructor or non-static initializer had ever been called. And so, the field (as it is final and so must be initialized explicitly) remains uninitialized. This causes the compiler to complain.

Had `CLASS_GUID` been just a final variable, option 4 would work but as it is also static, it cannot wait till the constructor executes to be initialized.

Back to Question without Answer

17. QID - 2.1126 : Working with Java Data Types - Variables and Objects

Note: This question may be considered too advanced for this exam.
Given:

```
String mStr = "123";
long m = // 1
```

Which of the following options when put at //1 will assign `123` to `m`?

Correct Options are : A B E

A. `new Long(mStr);`

> Auto unboxing will occur.

B. `Long.parseLong(mStr);`

~~C.~~ `Long.longValue(mStr);`

> longValue is a non-static method in Long class.

~~D.~~ `(new Long()).parseLong(mStr);`

> Long (or any wrapper class) does not have a no-args constructor, so new Long() is invalid.

E. `Long.valueOf(mStr).longValue();`

> `Long.valueOf(mStr)` returns a Long object containing 123. `longValue()` on the Long object returns 123.

Back to Question without Answer

18. QID - 2.1346 : Working with Java Data Types - Variables and Objects

Which of the following statements can be inserted at // 1 to make the code compile without errors?

```
public class InitTest{
   static int si = 10;
   int  i;
   final boolean bool;
   // 1
}
```

Correct Option is : E

~~**A.**~~ `instance { bool = true; }`

> you cannot put the word instance here. It is not a keyword.

~~**B.**~~ `InitTest() { si += 10; }`

> It is a valid constructor but does not initialize bool, which is a final variable and must be initialized either in an instance block or in a constructor.

~~**C.**~~ `{ si = 5; i = bool ? 1000 : 2000;}`

> cannot use bool before initializing it !

~~**D.**~~ `{ i = 1000; }`

> bool remains uninitialized.

E. `{ bool = (si > 5); i = 1000; }`

Explanation:

A final variable must be initialized when an instance is constructed, or else the code will not compile. This can be done either in an instance initializer or in EVERY constructor.
The keyword static is used to signify that a block is static initializer. If nothing is there before starting curly brace then it is an instance initializer.

Back to Question without Answer

19. QID - 2.917 : Working with Java Data Types - Variables and Objects

Given:

```
public class Square {
    private double side = 0;  // LINE 2

    public static void main(String[] args) {   // LINE 4
        Square sq = new Square();  // LINE 5
        side = 10;  // LINE 6
   }
}
```

What can be done to make this code compile and run?

Correct Option is : D

~~**A.**~~ replace // LINE 2 with:

```
private int side = 0;
```

~~**B.**~~ replace // LINE 2 with:

```
public int side = 0;
```

~~**C.**~~ replace // LINE 5 with:

```
double sq = new Square();
```

D. replace // LINE 6 with:

```
sq.side = 10;
```

`side` is not a global variable that you can access directly (Note that Java doesn't have the concept of a global variable). `side` is a field in `Square` class. So you need to specify which Square object's `side` you are trying to access.

An integer can be assigned to a double but not vice versa.

Back to Question without Answer

20. QID - 2.972 : Working with Java Data Types - Variables and Objects

Consider the following program :

```
class Test{
  public static void main(String[] args){
    short s = 10;    // 1
    char c = s;      // 2
    s = c;           // 3
  }
}
```

Identify the correct statements.

Correct Options are : A B

A. Line 3 is not valid.

B. Line 2 is not valid.

~~**C.**~~ It will compile because both short and char can hold 10.

~~**D.**~~ None of the lines 1, 2 and 3 is valid.

Line 1 is valid because 10 is a constant and can fit into a short.

Explanation:

Not all short values are valid char values, and neither are all char values valid short values, therefore compiler complains for both the lines 2 and 3. They will require an explicit cast.

Back to Question without Answer

21. QID - 2.865 : Working with Java Data Types - Variables and Objects

What will the following program print when compiled and run?

```
class Data {
    private int x = 0;
    private String y = "Y";
    public Data(int k){
        this.x = k;
    }
    public Data(String k){
        this.y = k;
    }
    public void showMe(){
        System.out.println(x+y);
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        new Data(10).showMe();
        new Data("Z").showMe();
    }
}
```

Correct Option is : B

~~**A.**~~ 0Z
10Y

B. 10Y
0Z

You are creating two different Data objects in the code. The first one uses a constructor that takes an integer and the second one uses a constructor that takes a String.

Thus, when you call showMe on the first object, it prints 10Y because "Y" is the default value of y as per the given code. When you call showMe on the second object, it prints 0Z because 0 is the default value of x as per the given code.

~~**C.**~~ It will not compile.

~~**D.**~~ It will throws an exception at run time.

Explanation:

Note that + operator is overloaded for String. So if you have a String as any operand for +, a new combined String will be created by concatenating the values of both the operands. Therefore, x+y will result in a String that concatenates integer x and String y.

Back to Question without Answer

22. QID - 2.1077 : Working with Java Data Types - Variables and Objects

What will the following program print?

```
public class TestClass{
  static boolean b;
  static int[] ia = new int[1];
  static char ch;
  static boolean[] ba = new boolean[1];
  public static void main(String args[]) throws Exception{
    boolean x = false;
    if( b ){
      x = ( ch == ia[ch]);
    }
    else x = ( ba[ch] = b );
    System.out.println(x+" "+ba[ch]);
  }
}
```

Correct Option is : D

~~**A.** true true~~

~~**B.** true false~~

~~**C.** false true~~

D. false false

~~**E.** It will not compile.~~

Explanation:

This question tests your knowledge on the default values of uninitialized primitives and object references. `booleans` are initialized to `false`, numeric types to 0 and object references to `null`. Elements of arrays are initialized to the default values of their types. So, elements of a `boolean` array are initialized to `false`. int, char, float to 0 and Objects to null.

In this case, `b` is `false`. So the else part of `if(b)` is executed.
`ch` is a numeric type to its value is `0`. `ba[0] = false` assigns `false` to `ba[0]` and returns `false` which is assigned to `x`.
Finally, `x` and `ba[0]` are printed as `false false`.

Back to Question without Answer

23. QID - 2.1199 : Working with Java Data Types - Variables and Objects

Given:

```
public class TestClass{
  public static int getSwitch(String str){
      return (int) Math.round( Double.parseDouble(str.substring(1, st
  }
  public static void main(String args []){
    switch(getSwitch(args[0])){
      case 0 : System.out.print("Hello ");
      case 1 : System.out.print("World"); break;
      default : System.out.print(" Good Bye");
    }
  }
}
```

What will be printed by the above code if it is run with command line:
`java TestClass --0.50`
(There are two minuses before 0.)

Correct Option is : C

~~**A.**~~ Hello

~~**B.**~~ World

C. Hello World

~~**D.**~~ Hello World Good Bye

~~**E.**~~ Good Bye

Explanation:

`str.substring(1, str.length()-1)` => `"--0.50".substring(1, (6-1) )` => `-0.5`
`Math.round(-0.5) = 0.0`
so `getSwitch(...)` returns 0 if passed an argument of `"--0.50"`.
Now, there is no "break" in case 0 of switch. so the control falls through to the next case (i.e. case 1) after printing Hello. At case 1, it prints World. And since there is a break. default is not executed.

Back to Question without Answer

24. QID - 2.1289 : Working with Java Data Types - Variables and Objects

Given the following statements, what will be the value of k and m after these statements are executed.

will not compile exception at runtime 12

```
int a = 5, b = 7, k = 0;
Integer m = null;
k = new Integer(a) + new Integer(b);
k = new Integer(a) + b;
k = b + new Integer(a);
m = new Integer(a) + new Integer(b);
```

12
12
12
12

Explanation:

In all of these cases, auto-unboxing of integers will occur. For the last statement, after unboxing a and b, the value 12 will be boxed into an Integer object.

Back to Question without Answer

25. QID - 2.1151 : Working with Java Data Types - Variables and Objects

Given that TestClass is a class, how many objects and reference variables are created by the following code?

```
TestClass t1, t2, t3, t4;
t1 = t2 = new TestClass();
t3 = new TestClass();
```

Correct Option is : B

~~**A.**~~ 2 objects, 3 references.

B. 2 objects, 4 references.

two news => two objects. t1, t2, t3, t4 => 4 references.

~~**C.**~~ 3 objects, 2 references.

~~**D.**~~ 2 objects, 2 references.

~~**E.**~~ None of the above.

Explanation:

A declared reference variable exists regardless of whether a reference value (i.e. an object) has been assigned to it or not.

Back to Question without Answer

26. QID - 2.1221 : Working with Java Data Types - Variables and Objects

Which of the following is illegal ?

Correct Option is : D

~~**A.**~~ char c = 320;

> This is valid because 320 is below the maximum value that a char can take, which is 2^16 -1. Remember that char can take only positive values.

~~**B.**~~ float f = 320;

> 320 is an int and so can be assigned to an int variable. f = 320.0 is not valid as 320.0 would be a double.

~~**C.**~~ double d = 320;

> This is valid because an int can be assigned to a double without any cast.

D. byte b = 320;

> 320 cannot fit into a byte so you must cast it.: byte b = (byte) 320;

~~**E.**~~ None of the above is illegal.

Back to Question without Answer

27. QID - 2.1041 : Working with Java Data Types - Variables and Objects

Which of the following is not a primitive data value in Java?

Correct Options are : A D

A. `"x"`

> This is a string containing x. String is not a primitive data type.

~~B.~~ `'x'`

> This is a char.

~~C.~~ `10.2F`

D. `Object`

~~E.~~ `false`

Explanation:

Java has only the following primitive data types:
boolean, byte, short, char, int, long, float and double.

Back to Question without Answer

28. QID - 2.1319 : Working with Java Data Types - Variables and Objects

Given the following code snippet:

```
int rate = 10;
int t = 5;
XXX amount = 1000.0;
for(int i=0; i<t; t++){
   amount = amount*(1 - rate/100);
}
```

What can XXX be?

Correct Option is : C

~~**A.**~~ int

~~**B.**~~ long

C. only double

~~**D.**~~ double or float

~~**E.**~~ float

Explanation:

There is no need for analyzing the whole code. `XXX amount = 1000.0;` will be valid only if XXX is double.

Note that the options do not include wrapper classes. Otherwise, Double is also valid

because of auto boxing.

Back to Question without Answer

29. QID - 2.1146 : Working with Java Data Types - Variables and Objects

What will the following program print?

```
public class TestClass{
  public static void main(String[] args){
    unsigned byte b = 0;
    b--;
    System.out.println(b);
  }
}
```

Correct Option is : E

~~**A.**~~ 0

~~**B.**~~ -1

~~**C.**~~ 255

~~**D.**~~ -128

E. It will not compile.

Explanation:

There no unsigned keyword in java! A `char` can be used as an unsigned integer.

Back to Question without Answer

30. QID - 2.1062 : Working with Java Data Types - Variables and Objects

The following code snippet will print 4.

```
int i1 = 1, i2 = 2, i3 = 3;
int i4 = i1 + (i2=i3 );
System.out.println(i4);
```

Correct Option is : A

A. True

~~**B.**~~ False

Explanation:

First the value of i1 is evaluated (i.e. 1). Now, `i2` is assigned the value of `i3` i.e. `i2` becomes `3`. Finally `i4` gets 1 +3 i.e. 4.

Back to Question without Answer

Working with Methods

Exam Objectives -

Create methods with arguments and return values Apply the static keyword to methods and fields Determine the effect upon object references and primitive values when they are passed into methods that change the values

01. QID - 2.854

What will be printed when the following code is compiled and run?

```
class A {
    public int getCode(){ return 2;}
}

class AA extends A {
  public long getCode(){ return 3;}
}

public class TestClass {

    public static void main(String[] args) throws Exception {
         A a = new A();
        A aa = new AA();
        System.out.println(a.getCode()+" "+aa.getCode());
    }

    public int getCode() {
        return 1;
    }
}
```

Select 1 option

A. 2 3

B. 2 2

C. It will throw an exception at run time.

D. The code will not compile.

Check Answer

02. QID - 2.1283

What is the correct declaration for an abstract method 'add' accessible to any class, takes no arguments and returns nothing?
(Use only one space between words)

Select 1 option

A. `public void add();`

B. `abstract add();`

C. `abstract null add();`

D. `abstract public void add(){ }`

E. `abstract public void add() throws Exception;`

Check Answer

03. QID - 2.912

Which of the following methods does not return any value?

Select 1 option

A.
```
public doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

B.
```
public null doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

C.
```
public doStuff() {
    //valid code not shown
}
```

D.
```
public void doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

E.
```
private doStuff() {
    //valid code not shown
}
```

Check Answer

04. QID - 2.1140

Which of the following are valid declarations?

Select 1 option

A. `abstract int absMethod(int param) throws Exception;`

B. `abstract native int absMethod(int param) throws Exception;`

C. `float native getVariance() throws Exception;`

D. `abstract private int absMethod(int param) throws Exception;`

E. `strictfp float f;`

Check Answer

05. QID - 2.902

Given:

```
class Triangle{
    public int base;
    public int height;
    public double area = 0;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }
    public void updateArea(){
        double a = base*height/2;
        area = a;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

Which variables are not accessible from anywhere within given class code (except from where they are declared)?

Select 1 option

A. base, height, area

B. area, b, h

C. base, height

D. b, h, a

Check Answer

06. QID - 2.957

What will the following program print?

```
public class TestClass{
  static int someInt = 10;
  public static void changeIt(int a){
    a = 20;
  }
  public static void main(String[] args){
    changeIt(someInt);
    System.out.println(someInt);
  }
}
```

Select 1 option

A. 10

B. 20

C. It will not compile.

D. It will throw an exception at runtime.

E. None of the above.

Check Answer

07. QID - 2.1312

Which of the following code fragments are valid method declarations?

Select 1 option

A. void method1{ }

B. void method2( ) { }

C. void method3(void){ }

D. method4{ }

E. method5(void){ }

Check Answer

08. QID - 2.1343

Consider the following class...

```
class TestClass{
   int x;
   public static void main(String[] args){
      // lot of code.
   }
}
```

Select 1 option

A. By declaring x as static, main can access `this.x`

B. By declaring x as public, main can access `this.x`

C. By declaring x as protected, main can access `this.x`

D. main cannot access `this.x` as it is declared now.

E. By declaring x as private, main can access `this.x`

Check Answer

09. QID - 2.1001

Consider the following class:

```
public class Test{
    public int id;
}
```

Which of the following is the correct way to make the variable 'id' read only for any other class?

Select 1 option

A. Make 'id' private.

B. Make 'id' private and provide a public method getId() which will return its value.

C. Make 'id' static and provide a public static method getId() which will return its value.

D. Make id 'protected'

Check Answer

10. QID - 2.1055

Identify valid method declarations.

Select 1 option

A.
```
public void methodX(int... i, String s);
```

B.
```
public void methodX(int... i, String... s);
```

C.
```
public void methodX(int i, int... j);
```

D.
```
public int... methodX(int i);
```

Check Answer

11. QID - 2.904

Given:

```
class Triangle{
    public int base;
    public int height;
    private static double ANGLE;

    public static double getAngle();

    public static void Main(String[] args) {
        System.out.println(getAngle());
    }
}
```

Identify the correct statements:

Select 1 option

A. It will not compile because it does not implement `setAngle` method.

B. It will not compile because `ANGLE` cannot be private.

C. It will not compile because `getAngle()` has no body.

D. It will not compile because `ANGLE` field is not initialized.

E. It will not compile because of the name of the method `Main` instead of `main`.

Check Answer

12. QID - 2.1308

What is the result of compiling and running the following code ?

```
public class TestClass{
   static int si = 10;
   public static void main (String args[]){
      new TestClass();
   }
   public TestClass(){
      System.out.println(this);
   }
   public String toString(){
      return "TestClass.si = "+this.si;
   }
}
```

Select 1 option

A. The class will not compile because you cannot override `toString()` method.

B. The class will not compile as `si` being `static`, `this.si` is not a valid statement.

C. It will print `TestClass@nnnnnnnn`, where `nnnnnnn` is the hash code of the TestClass object referred to by 'this'.

D. It will print `TestClass.si = 10;`

E. None of the above.

Check Answer

13. QID - 2.1252

What will be the output when the following program is run?

```
public class TestClass{
    char c;
    public void m1(){
        char[ ] cA = { 'a' , 'b'};
        m2(c, cA);
        System.out.println( ( (int)c)  + ", " + cA[1] );
    }
    public void m2(char c, char[ ] cA){
        c = 'b';
        cA[1] = cA[0] = 'm';
    }
    public static void main(String args[]){
        new TestClass().m1();
    }
}
```

Select 1 option

A. Compile time error.

B. , m

C. 0 , m

D. b , b

E. b , m

Check Answer

14. QID - 2.829

What will the following program print when compiled and run:

```
public class TestClass {
    public static void main(String[] args) {
            someMethod();
    }

    static void someMethod(Object parameter)  {
          System.out.println("Value is "+parameter);
    }
}
```

Select 1 option

A. It will not compile.

B. `Value is null`

C. `Value is`

D. It will throw a `NullPointerException` at run time.

Check Answer

15. QID - 2.833

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showTwo(200);
        System.out.println(ct.myValue);
        ct.showOne(100);
        System.out.println(ct.myValue);
    }
}
```

Select 1 option

A. 0 followed by 100.

B. 100 followed by 100.

C. 0 followed by 200.

D. 100 followed by 200.

E. 200 followed by 200.

F. 200 followed by 100

Check Answer

16. QID - 2.830

Consider the following method :

```
public void myMethod(int m, Object p, double d){
  ... valid code here
}
```

Assuming that there is no other method with the same name, which of the following options are correct regarding the above method?

Select 1 option

A. If this method is called with two parameters, the value of d in the method will be 0.0.

B. If this method is called with one parameter, the value of p and d in the method will be null and 0.0 respectively.

C. If this method is called with one parameter, the call will throw a NullPointerException.

D. If this method is called with one parameter, the call will throw a NullPointerException only if the code in the method tries to access p.

E. If this method is called with two parameters, the code will not compile.

Check Answer

17. QID - 2.1218

Complete the code using blue labels on the right so that the output will 210.

(You may leave some blanks empty.)

public int a = a + offset;

return a; u.update(a, 111);

a = u.update(a, 111);

return; public void

```
3  public class Updater
4  {
5   [          ] update(int a, int offset)
6      {
7          [          ]
8          [          ]
9      }
10
11     public static void main(String[] args)
12     {
13         Updater u = new Updater();
14
15         int a = 99;
16
17         [          ]
18
19         System.out.println(a);
20
21     }
22 }
23
```

Check Answer

18. QID - 2.903

A java source file contains the following code:

```
interface I {
  int getI(int a, int b);
}

interface J{
    int getJ(int a, int b, int c);
}

abstract class MyIJ implements J , I { }

class MyI{
    int getI(int x, int y){ return x+y; }
}

interface K extends J{
    int getJ(int a, int b, int c, int d);
}
```

Identify the correct statements:

Select 1 option

A. It will fail to compile because of `MyIJ`

B. It will fail to compile because of `MyIJ` and `K`

C. It will fail to compile because of `K`

D. It will fail to compile because of `MyI` and `K`

E. It will fail to compile because of `MyIJ, K,` and `MyI`

F. It will compile without any error.

Check Answer

19. QID - 2.1092

What will the following class print when compiled and run?

```
class Holder{
   int value = 1;
   Holder link;
   public Holder(int val){ this.value = val; }
   public static void main(String[] args){
 final Holder a = new Holder(5);
 Holder b = new Holder(10);
 a.link = b;
 b.link = setIt(a, b);
 System.out.println(a.link.value+" "+b.link.value);
   }

   public static Holder setIt(final Holder x, final Holder y){
       x.link = y.link;
       return x;
   }

}
```

Select 1 option

A. It will not compile because 'a' is final.

B. It will not compile because method setIt() cannot change x.link.

C. It will print 5, 10.

D. It will print 10, 10.

E. It will throw an exception when run.

Check Answer

20. QID - 2.1134

Which of the following are valid at line 1?

```
public class X{
   //line 1: insert code here.
}
```

Select 2 options

A. String s;

B. String s = 'asdf';

C. String s = 'a';

D. String s = this.toString();

E. String s = asdf;

Check Answer

21. QID - 2.1228

What will be the result of attempting to compile and run the following class?

```
public class InitTest{
   static String s1 = sM1("a");{
      s1 = sM1("b");
   }
   static{
      s1 = sM1("c");
   }
   public static void main(String args[]){
      InitTest it = new InitTest();
   }
   private static String sM1(String s){
      System.out.println(s);  return s;
   }
}
```

Select 1 option

A. The program will fail to compile.

B. The program will compile without error and will print `a, c` and `b` in that order when run.

C. The program will compile without error and will print `a, b` and `c` in that order when run.

D. The program will compile without error and will print `c, a` and `b` in that order when run.

E. The program will compile without error and will print `b, c` and `a` in that order when run.

Check Answer

22. QID - 2.906

Given:

```
class Triangle{
    public int base;
    public int height;
    private final double ANGLE;

    public  void setAngle(double a){  ANGLE = a;  }

    public static void main(String[] args) {
        Triangle t = new Triangle();
        t.setAngle(90);
    }
}
```

Select 1 option

A. the value of `ANGLE` will not be set to `90` by the `setAngle` method.

B. An exception will be thrown at run time.

C. The code will work as expected setting the value of `ANGLE` to `90`.

D. The code will not compile.

Check Answer

23. QID - 2.893

Given:

```
interface Worker {
   void  performWork();
}

class FastWorker  implements Worker {
   public void  performWork(){ }
}
```

You are creating a class that follows "program to an interface" principle. Which of the following line of code will you most likely be using?

Select 1 option

A.
```
public FastWorker getWorker(){
return new Worker();
}
```

B.
```
public FastWorker getWorker(){
return new FastWorker();
}
```

C.
```
public Worker getWorker(){
return new FastWorker();
}
```

D.
```
public Worker getWorker(){
return new Worker();
}
```

Check Answer

24. QID - 2.1204

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
  public static void main(String args[ ] ){
    Object a, b, c ;
    a = new String("A");
    b = new String("B");
    c = a;
    a = b;
    System.out.println(""+c);
  }
}
```

Select 1 option

A. The program will print `java.lang.String@XXX`, where XXX is the memory location of the object a.

B. The program will print A

C. The program will print B

D. The program will not compile as a,b and c are of type Object.

E. The program will print `java.lang.String@XXX`, where XXX is the hash code of the object a.

Check Answer

25. QID - 2.1188

What should be the return type of the following method?

```
public RETURNTYPE methodX( byte by){
    double d = 10.0;
    return (long) by/d*3;
}
```

Select 1 option

A. int

B. long

C. double

D. float

E. byte

Check Answer

26. QID - 2.1293

Which of the following correctly defines a method named stringProcessor that can be called by other programmers as follows: `stringProcessor(str1)` or `stringProcessor(str1, str2)` or `stringProcessor(str1, str2, str3)`, where `str1`, `str2`, and `str3` are references to Strings.

Select 1 option

A.
```
public void stringProcessor(...String){
}
```

B.
```
public void stringProcessor(String... strs){
}
```

C.
```
public void stringProcessor(String[] strs){
}
```

D.
```
public void stringProcessor(String a, String b, String c){
}
```

E. Three separate methods need to be written.

Check Answer

27. QID - 2.1362

Which of the following statements are true?

Select 2 options

A. private keyword can never be applied to a class.

B. synchronized keyword can never be applied to a class.

C. synchronized keyword may be applied to a non-primitive variable.

D. final keyword can never be applied to a class.

E. A final variable can be shadowed in a subclass.

Check Answer

28. QID - 2.1178

Which one of the following class definitions is/are a legal definition of a class that cannot be instantiated?

```
class Automobile{
   abstract void honk();  //(1)
}

abstract class Automobile{
   void honk();   //(2)
}

abstract class Automobile{
   void honk(){};   //(3)
}

abstract class Automobile{
   abstract void honk(){}   //(4)
}

abstract class Automobile{
   abstract void honk();   //(5)
}
```

Select 2 options

A. 1

B. 2

C. 3

D. 4

E. 5

Check Answer

29. QID - 2.1074

Consider the following class definition:

```
public class TestClass{
   public static void main(){  new TestClass().sayHello(); }   //1
   public static void sayHello(){ System.out.println("Static Hello W
   public void sayHello() { System.out.println("Hello World "); }  /
}
```

What will be the result of compiling and running the class?

Select 1 option

A. It will print `Hello World`.

B. It will print `Static Hello World`.

C. Compilation error at line 2.

D. Compilation error at line 3.

E. Runtime Error.

Check Answer

30. QID - 2.974

What will be the contents of s1 and s2 at the time of the println statement in the main method of the following program?

```
import java.util.*;
public class TestClass{
   public static void main(String args[]){
      Stack s1 = new Stack ();
      Stack s2 = new Stack ();
      processStacks (s1,s2);
      System.out.println (s1 + "    "+ s2);
   }
   public static void processStacks(Stack x1, Stack x2){
      x1.push (new Integer ("100")); //assume that the method push a
      x2 = x1;
   }
 }
```

Select 1 option

A. [100] [100]

B. [100] []

C. [] [100]

D. [] []

Check Answer

31. QID - 2.863

Consider the following code appearing in the same file:

```
class Data {
    private int x = 0, y = 0;
    public Data(int x, int y){
        this.x = x; this.y = y;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d = new Data(1, 1);
        //add code here
    }
}
```

Which of the following options when applied individually will change the Data object currently referred to by the variable d to contain 2, 2 as values for its data fields?

Select 1 option

A. Add the following two statements :

```
d.x = 2;
d.y = 2;
```

B. Add the following statement:

```
d = new Data(2, 2);
```

C. Add the following two statements:

```
d.x += 1;
d.y += 1;
```

D. Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x.setInt(x);   this.y.setInt(y);
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

E. Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x = x;   this.y = y;
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

Check Answer

32. QID - 2.831

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
        System.out.println(this.myValue);
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
        System.out.println(this.myValue);
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showOne(100);
        ct.showTwo(200);
    }
}
```

Select 1 option

A. 0 followed by 100.

B. 100 followed by 100.

C. 0 followed by 200.

D. 100 followed by 200.

Check Answer

33. QID - 2.1294

What will the code shown below print when run?

```
public class TestClass{
    static class Wrapper{
        int w = 10;
    }

    static Wrapper changeWrapper(Wrapper w){
        w = new Wrapper();
        w.w += 9;
        return w;
    }


    public static void main(String[] args){
        Wrapper w = new Wrapper();
        w.w = 20;
        changeWrapper(w);
        w.w += 30;
        System.out.println(w.w);
        w = changeWrapper(w);
        System.out.println(w.w);
     }
}
```

Select 2 options

A. 9

B. 19

C. 30

D. 20

E. 29

F. 50

Check Answer

Working with Methods (Answered)

01. QID - 2.854 : Working with Methods

What will be printed when the following code is compiled and run?

```
class A {
    public int getCode(){ return 2;}
}

class AA extends A {
  public long getCode(){ return 3;}
}

public class TestClass {

    public static void main(String[] args) throws Exception {
         A a = new A();
        A aa = new AA();
        System.out.println(a.getCode()+" "+aa.getCode());
    }

    public int getCode() {
        return 1;
    }
}
```

Correct Option is : D

~~**A.**~~ 2 3

~~**B.**~~ 2 2

~~**C.**~~ It will throw an exception at run time.

D. The code will not compile.

Class AA is trying to override `getCode()` method of class A but its return type is incompatible with the A's `getCode`.

When the return type of the overridden method (i.e. the method in the base/super class) is a primitive, the return type of the overriding method (i.e. the method in the sub class) must match the return type of the overridden method.

In case of Objects, the base class method can have a covariant return type, which means, it can return either return the same class or a sub class object. For example, if base class method is:

```
public A getA(){ ... }
```

a subclass can override it with:

`public AA getA(){ ... }` because AA is a subclass of A.

Back to Question without Answer

02. QID - 2.1283 : Working with Methods

What is the correct declaration for an abstract method 'add' accessible to any class, takes no arguments and returns nothing?
(Use only one space between words)

Correct Option is : E

~~**A.**~~ `public void add();`

> An abstract method must have the `abstract` keyword and must not have a method body i.e. { }.

~~**B.**~~ `abstract add();`

> A method that is not supposed to return anything must specify `void` as its return type.

~~**C.**~~ `abstract null add();`

> A method that is not supposed to return anything must specify `void` as its return type. `null` is not a type, though it is a valid return value for any type.

~~**D.**~~ `abstract public void add(){ }`

> It is invalid because has a method body i.e. { }.

E. `abstract public void add() throws Exception;`

Back to Question without Answer

03. QID - 2.912 : Working with Methods

Which of the following methods does not return any value?

Correct Option is : D

~~**A.**~~
```
public doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

> It is missing the return type. Every method must have a return type specified in its declaration.
> It could be a valid constructor though if the class is named `doStuff` because the constructors don't return anything, not even `void`.

~~**B.**~~
```
public null doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

> `null` can be a return value not a return type because null is not a type.

~~**C.**~~
```
public doStuff() {
    //valid code not shown
}
```

> This is not a valid method because there is no return type declared. Although it can be a valid constructor if the name of the class is `doStuff`.

D.
```
public void doStuff() throws FileNotFoundException,
IllegalArgumentException{
    //valid code not shown
}
```

A method that does not return anything should declare its return type as `void`. Note that this is different from constructors. A constructor also doesn't return anything but for a constructor, you don't specify any return type at all. That is how a constructor is differentiated from a regular method.

E.
```
private doStuff() {
    //valid code not shown
}
```

This is not a valid method because there is no return type declared. Although it can be a valid constructor if the name of the class is `doStuff`.

Back to Question without Answer

04. QID - 2.1140 : Working with Methods

Which of the following are valid declarations?

Correct Option is : A

A. `abstract int absMethod(int param) throws Exception;`

~~B.~~ `abstract native int absMethod(int param) throws Exception;`

> `native` method cannot be `abstract`.

~~C.~~ `float native getVariance() throws Exception;`

> return type should always be on the immediate left of method name.

~~D.~~ `abstract private int absMethod(int param) throws Exception;`

> `private` method cannot be `abstract`. A `private` method is not inherited so how can a subclass implement it?

~~E.~~ `strictfp float f;`

> The keyword `strictfp` can only be applied to class or method declarations.

Back to Question without Answer

05. QID - 2.902 : Working with Methods

Given:

```
class Triangle{
    public int base;
    public int height;
    public double area = 0;

    public Triangle(int base, int height){
        this.base = base; this.height = height;
        updateArea();
    }
    public void updateArea(){
        double a = base*height/2;
        area = a;
    }
    public void setBase(int b){ base  = b; updateArea(); }
    public void setHeight(int h){ height  = h; updateArea(); }
}
```

Which variables are not accessible from anywhere within given class code (except from where they are declared)?

Correct Option is : D

~~**A.**~~ base, height, area

~~**B.**~~ area, b, h

~~**C.**~~ base, height

D. b, h, a

b and h are method parameters and are only accessible in the method setBase and setHeight respectively.
a is a local variable and is accessible only in updateArea method.

base, height, and area are instance level and can be accessed from anywhere within the class where "this" is accessible.

Explanation:

"class level" means static fields and they can be accessed from anywhere (i.e. static as well as non-static methods) in the class (and from outside the class depending on their accessibility).
"instance level" means the instance fields and they can be accessed only from instance methods in the class.

Back to Question without Answer

06. QID - 2.957 : Working with Methods

What will the following program print?

```
public class TestClass{
  static int someInt = 10;
  public static void changeIt(int a){
    a = 20;
  }
  public static void main(String[] args){
    changeIt(someInt);
    System.out.println(someInt);
  }
}
```

Correct Option is : A

A. 10

~~**B.**~~ 20

~~**C.**~~ It will not compile.

~~**D.**~~ It will throw an exception at runtime.

~~**E.**~~ None of the above.

Explanation:

In case of primitives such an an int, it is the value of the primitive that is passed. For

example, in this question, when you pass `someInt` to `changeIt` method, you are actually passing the value `10` to the method, which is then assigned to method variable '`a`'. In the method, you assign `20` to '`a`'. However, this does not change the value contained in `someInt`. `someInt` still contains `10`. Therefore, `10` is printed.

Theoretically, java supports Pass by Value for everything (i.e. primitives as well as Objects).

. Primitives are always passed by value.
. Object "references" are passed by value. So it looks like the object is passed by reference but actually it is the value of the reference that is passed.

An example:

```
Object o1 = new Object(); //Let us say, the object is stored at memory location 15000.
        //Since o1 actually stores the address of the memory location where the object is stored, it contains 15000.
```

Now, when you call someMethod(o1); the value 15000 is passed to the method.

Inside the method someMethod():

```
someMethod( Object localVar) {
            /*localVar now contains 15000, which means it also points to the same memory location where the object is stored.
            Therefore, when you call a method on localVar, it will be executed on the same object.
            However, when you change the value of localVar itself, for example if you do localVar=null,
            it then it starts pointing to a different memory location. But the original variable o1 still
            contains 15000 so it still points to the same object. */
        }
```

If you need even more detailed explanation, please check

http://www.javaranch.com/campfire/StoryPassBy.jsp

Back to Question without Answer

07. QID - 2.1312 : Working with Methods

Which of the following code fragments are valid method declarations?

Correct Option is : B

~~**A.**~~ void method1{ }

It does not have () after method1.

B. void method2( ) { }

~~**C.**~~ void method3(void){ }

The keyword void is not a valid type for a parameter.

~~**D.**~~ method4{ }

Methods must specify a return type and '( )'. If the method does not want to return a value, it should specify void.

~~**E.**~~ method5(void){ }

If the method does not take any parameter, it should have empty brackets instead of void.

Explanation:

A valid method declaration MUST specify a return type, all other things are optional.

Back to Question without Answer

08. QID - 2.1343 : Working with Methods

Consider the following class...

```
class TestClass{
   int x;
   public static void main(String[] args){
      // lot of code.
   }
}
```

Correct Option is : D

~~**A.**~~ By declaring x as static, main can access `this.x`

~~**B.**~~ By declaring x as public, main can access `this.x`

~~**C.**~~ By declaring x as protected, main can access `this.x`

D. main cannot access `this.x` as it is declared now.

Because main() is a static method. It does not have 'this'!

~~**E.**~~ By declaring x as private, main can access `this.x`

Explanation:

It is not possible to access x from main without making it static. Because main is a static method and only static members are accessible from static methods. There is no 'this' available in main so none of the this.x are valid.

Back to Question without Answer

09. QID - 2.1001 : Working with Methods

Consider the following class:

```
public class Test{
    public int id;
}
```

Which of the following is the correct way to make the variable 'id' read only for any other class?

Correct Option is : B

~~**A.**~~ Make 'id' private.

> This will not allow others to read or write.

B. Make 'id' private and provide a public method getId() which will return its value.

~~**C.**~~ Make 'id' static and provide a public static method getId() which will return its value.

~~**D.**~~ Make id 'protected'

Explanation:

This is a standard way of providing read only access to internal variables.

Back to Question without Answer

10. QID - 2.1055 : Working with Methods

Identify valid method declarations.

Correct Option is : C

~~A.~~ `public void methodX(int... i, String s);`

A var-args parameter must always be the last one.

~~B.~~ `public void methodX(int... i, String... s);`

A method can have only one var-args parameter.

C. `public void methodX(int i, int... j);`

~~D.~~ `public int... methodX(int i);`

The ... syntax is not applicable for return type.

Back to Question without Answer

11. QID - 2.904 : Working with Methods

Given:

```
class Triangle{
    public int base;
    public int height;
    private static double ANGLE;

    public static double getAngle();

    public static void Main(String[] args) {
        System.out.println(getAngle());
    }
}
```

Identify the correct statements:

Correct Option is : C

~~**A.**~~ It will not compile because it does not implement `setAngle` method.

> There is no requirement that a class has to have a setter as well as a getter.

~~**B.**~~ It will not compile because `ANGLE` cannot be private.

> Any field can be made private.

C. It will not compile because `getAngle()` has no body.

~~**D.**~~ It will not compile because `ANGLE` field is not initialized.

> Since it is a static field, it will get a default value of 0.0.

~~**E.**~~ It will not compile because of the name of the method `Main` instead of `main`.

A class can have a method named Main. Although, since it is not same as main, it will not be considered the standard main method that the JVM can invoke when the program is executed.

Back to Question without Answer

12. QID - 2.1308 : Working with Methods

What is the result of compiling and running the following code ?

```
public class TestClass{
   static int si = 10;
   public static void main (String args[]){
      new TestClass();
   }
   public TestClass(){
      System.out.println(this);
   }
   public String toString(){
      return "TestClass.si = "+this.si;
   }
}
```

Correct Option is : D

~~**A.**~~ The class will not compile because you cannot override `toString()` method.

> You sure can. `toString()` is defined as public and non-final method in Object class.

~~**B.**~~ The class will not compile as `si` being `static`, `this.si` is not a valid statement.

> static member can be accessed by static and non-static methods both. Non-static can only be accessed by non-static.

~~**C.**~~ It will print `TestClass@nnnnnnnn`, where `nnnnnnn` is the hash code of the TestClass object referred to by 'this'.

> It would have been correct if `toString()` were not overridden. This is the behavior of the `toString()` provided by Object class.

D. It will print `TestClass.si = 10;`

E. None of the above.

Explanation:

The `toString` method for class Object returns a string consisting of the name of the class of which the object is an instance, the at-sign character `@', and the unsigned hexadecimal representation of the hash code of the object. In other words, this method returns a string equal to the value of:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

Back to Question without Answer

13. QID - 2.1252 : Working with Methods

What will be the output when the following program is run?

```
public class TestClass{
    char c;
    public void m1(){
        char[ ] cA = { 'a' , 'b'};
        m2(c, cA);
        System.out.println( ( (int)c)  + ", " + cA[1] );
    }
    public void m2(char c, char[ ] cA){
        c = 'b';
        cA[1] = cA[0] = 'm';
    }
    public static void main(String args[]){
        new TestClass().m1();
    }
}
```

Correct Option is : C

~~**A.**~~ Compile time error.

c is an instance variable of numeric type so it will be given a default value of 0, which prints as empty space.

~~**B.**~~ , m

Without the cast to int, c would be printed as empty space and cA[1] is 'm'

C. 0 , m

Because of the explicit cast to int in the println() call, c will be printed as 0.

~~**D.**~~ b , b

~~**E.**~~ b , m

Explanation:

Note that Arrays are Objects (i.e. `cA instanceof Object` is `true`) so are effectively passed by reference. So in `m1()` the change in `cA[1]` done by `m2()` is reflected everywhere the array is used.
`c` is a primitive type and is passed by value. In method `m2()` the passed parameter c is different than instance variable 'c' as local variable hides the instance variable. So instance member 'c' keeps its default (i.e. 0) value.

Back to Question without Answer

14. QID - 2.829 : Working with Methods

What will the following program print when compiled and run:

```
public class TestClass {
    public static void main(String[] args) {
            someMethod();
    }

    static void someMethod(Object parameter)  {
          System.out.println("Value is "+parameter);
    }
}
```

Correct Option is : A

A. It will not compile.

To call someMethod(Object parameter), you must pass exactly one parameter. So someMethod() is not a valid call to this method and the code will not compile.
Note that parameter will not be assigned a null or default value.

However, if the method were declared to take variable number of arguments, it would have been valid to call it without any parameters. For example:

```
public static void someMethod(Object... params){
    System.out.println(params.length);
}
```

In this case, calling someMethod() without any parameter will print 0. i.e. the length of params array will be 0. params will NOT be null.

B. ~~`Value is null`~~

~~**C.**~~ `Value is`

~~**D.**~~ It will throw a `NullPointerException` at run time.

Back to Question without Answer

15. QID - 2.833 : Working with Methods

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showTwo(200);
        System.out.println(ct.myValue);
        ct.showOne(100);
        System.out.println(ct.myValue);
    }
}
```

Correct Option is : E

~~**A.**~~ 0 followed by 100.

~~**B.**~~ 100 followed by 100.

~~**C.**~~ 0 followed by 200.

~~**D.**~~ 100 followed by 200.

E. 200 followed by 200.

~~**F.**~~ 200 followed by 100

Explanation:

There are a couple of important concepts in this question:

1. Within an instance method, you can access the current object of the same class using 'this'. Therefore, when you access this.myValue, you are accessing the instance member myValue of the ChangeTest instance.

2. If you declare a local variable (or a method parameter) with the same name as the instance field name, the local variable "shadows" the member field. Ideally, you should be able to access the member field in the method directly by using the name of the member (in this example, myValue). However, because of shadowing, when you use myValue, it refers to the local variable instead of the instance field.

In showTwo method, there are two variables accessible with the same name myValue. One is the method parameter and another is the member field of ChangeTest object. Ideally, you should be able to access the member field in the method directly by using the name myValue but in this case, the method parameter shadows the member field because it has the same name. So by doing this.myValue, you are changing the instance variable myValue by assigning it the value contained in local variable myValue, which is 200. So in the next line when you print ct.myValue, it prints 200.

Now, in the showOne() method also, there are two variables accessible with the same name myValue. One is the method parameter and another is the member field of ChangeTest object. So when you use myValue, you are actually using the method parameter instead of the member field.

Therefore, when you do : myValue = myValue; you are actually assigning the value contained in method parameter myValue to itself. You are not changing the member field myValue. Hence, when you do System.out.println(ct.myValue); in the next line, it still prints 200.

Back to Question without Answer

16. QID - 2.830 : Working with Methods

Consider the following method :

```
public void myMethod(int m, Object p, double d){
  ... valid code here
}
```

Assuming that there is no other method with the same name, which of the following options are correct regarding the above method?

Correct Option is : E

~~**A.**~~ If this method is called with two parameters, the value of d in the method will be 0.0.

~~**B.**~~ If this method is called with one parameter, the value of p and d in the method will be null and 0.0 respectively.

~~**C.**~~ If this method is called with one parameter, the call will throw a NullPointerException.

~~**D.**~~ If this method is called with one parameter, the call will throw a NullPointerException only if the code in the method tries to access p.

E. If this method is called with two parameters, the code will not compile.

Explanation:

To call `myMethod(int m, Object p, double d)`, you must pass exactly three

parameters. If you try to pass less (or more) number of parameters, the code will not compile. Note that method parameters are not assigned default values.

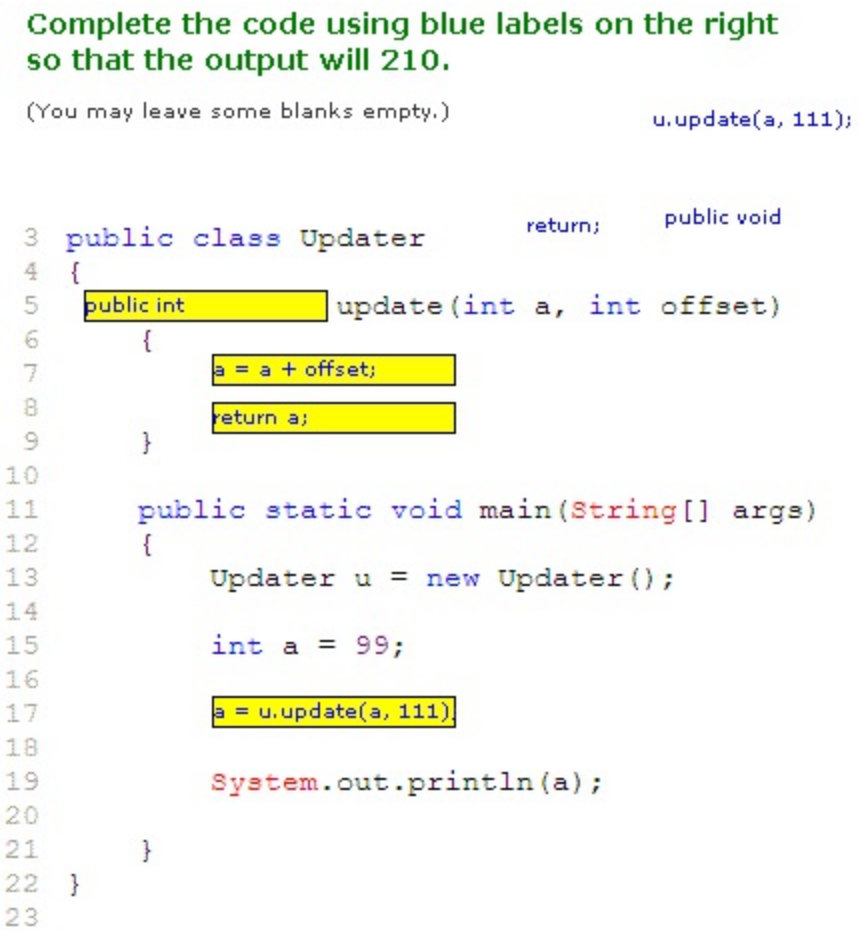
It is possible to declare a method that can take variable number of parameters. For example:

```
public static void someMethod(Object... params){
        System.out.println(params.length);
    }
```

You can call this method by passing any number of parameters. In this case, calling `someMethod()` without any parameter will print 0. i.e. the length of params array will be 0. `params` will NOT be `null`.

Back to Question without Answer

17. QID - 2.1218 : Working with Methods



Explanation:

This question is based on the principle that primitives are always passed by value. Thus, when you pass a to `update()` method, the value of a is passed. The variable `a` in `update()` method is not same as the `a` in `main()`. It is a completely different variable and so updating `update()` method's `a` does not affect `main()'s a`. Therefore, we need to return the new value from `update()` method and assign it to `main()'s a`.

The following is the complete code listing:

```
public class Updater {
  public int update(int a, int offset){
    a = a + offset;
```

```
    return a;
  }

  public static void main(String[] args) {
    Updater u = new Updater();
    int a = 99;
    a = u.update(a, 111);
    System.out.println(a);
  }
}
```

Back to Question without Answer

18. QID - 2.903 : Working with Methods

A java source file contains the following code:

```
interface I {
  int getI(int a, int b);
}

interface J{
    int getJ(int a, int b, int c);
}

abstract class MyIJ implements J , I { }

class MyI{
    int getI(int x, int y){ return x+y; }
}

interface K extends J{
    int getJ(int a, int b, int c, int d);
}
```

Identify the correct statements:

Correct Option is : F

~~**A.**~~ It will fail to compile because of `MyIJ`

> `MyIJ` declares that it implements interfaces I and J, but does not implement the methods declared in these interfaces. However, since `MyIJ` has been declared as `abstract`, it is valid.

~~**B.**~~ It will fail to compile because of `MyIJ` and `K`

~~**C.**~~ It will fail to compile because of `K`

K is a valid interface because an interface is permitted to extend another interface.

D. It will fail to compile because of `MyI` and `K`

Both are valid.

E. It will fail to compile because of `MyIJ, K,` and `MyI`

F. It will compile without any error.

Back to Question without Answer

19. QID - 2.1092 : Working with Methods

What will the following class print when compiled and run?

```
class Holder{
   int value = 1;
   Holder link;
   public Holder(int val){ this.value = val; }
   public static void main(String[] args){
 final Holder a = new Holder(5);
 Holder b = new Holder(10);
 a.link = b;
 b.link = setIt(a, b);
 System.out.println(a.link.value+" "+b.link.value);
   }

   public static Holder setIt(final Holder x, final Holder y){
       x.link = y.link;
       return x;
   }

}
```

Correct Option is : E

~~**A.**~~ It will not compile because 'a' is final.

> 'a' is final is true, but that only means that a will keep pointing to the same object for the entire life of the program. The object's internal fields, however, can change.

~~**B.**~~ It will not compile because method setIt() cannot change x.link.

> Since x and y are final, the method cannot change what x and y to point to some other object but it can change the objects' internal fields.

~~**C.**~~ It will print 5, 10.

~~**D.**~~ It will print 10, 10.

E. It will throw an exception when run.

When method setIt() executes, x.link = y.link, x.link becomes null because y.link is null so a.link.value throws NullPointerException.

Back to Question without Answer

20. QID - 2.1134 : Working with Methods

Which of the following are valid at line 1?

```
public class X{
   //line 1: insert code here.
}
```

Correct Options are : A D

A. String s;

~~**B.**~~ String s = 'asdf';

> A string must be enclosed in double quotes ".

~~**C.**~~ String s = 'a';

> 'a' is a char. "a" is a String.

D. String s = this.toString();

> Since every class directly or indirectly extends Object class and since Object class has a toString() method, that toString() method will be invoked and the String that it returns will be assigned to s.

~~**E.**~~ String s = asdf;

> there is no variable asdf defined in the given class.

Back to Question without Answer

21. QID - 2.1228 : Working with Methods

What will be the result of attempting to compile and run the following class?

```
public class InitTest{
   static String s1 = sM1("a");{
      s1 = sM1("b");
   }
   static{
      s1 = sM1("c");
   }
   public static void main(String args[]){
      InitTest it = new InitTest();
   }
   private static String sM1(String s){
      System.out.println(s);  return s;
   }
}
```

Correct Option is : B

~~**A.**~~ The program will fail to compile.

B. The program will compile without error and will print `a, c` and `b` in that order when run.

~~**C.**~~ The program will compile without error and will print `a, b` and `c` in that order when run.

~~**D.**~~ The program will compile without error and will print `c, a` and `b` in that order when run.

~~**E.**~~ The program will compile without error and will print `b, c` and `a` in that order when run.

Explanation:

First, static statements/blocks are called IN THE ORDER they are defined. (Hence, a and c will be printed.)
Next, instance initializer statements/blocks are called IN THE ORDER they are defined. Finally, the constructor is called. So, then it prints b.

Back to Question without Answer

22. QID - 2.906 : Working with Methods

Given:

```
class Triangle{
    public int base;
    public int height;
    private final double ANGLE;

    public  void setAngle(double a){  ANGLE = a;  }

    public static void main(String[] args) {
        Triangle t = new Triangle();
        t.setAngle(90);
    }
}
```

Correct Option is : D

~~**A.**~~ the value of `ANGLE` will not be set to `90` by the `setAngle` method.

~~**B.**~~ An exception will be thrown at run time.

~~**C.**~~ The code will work as expected setting the value of `ANGLE` to `90`.

D. The code will not compile.

Explanation:

The given code has two problems:
1. If you declare a field to be `final`, it must be explicitly initialized by the time the creation of an object of the class is complete. So you can either initialize it

immediately:

```
private final double ANGLE = 0;
```

or you can initialize it in the constructor or an instance block.

2. Since `ANGLE` is `final`, you can't change its value once it is set. Therefore the `setAngle` method will also not compile.

Back to Question without Answer

23. QID - 2.893 : Working with Methods

Given:

```
interface Worker {
   void  performWork();
}

class FastWorker  implements Worker {
   public void  performWork(){ }
}
```

You are creating a class that follows "program to an interface" principle. Which of the following line of code will you most likely be using?

Correct Option is : C

~~A.~~ `public FastWorker getWorker(){ return new Worker(); }`

> This will not compile because `Worker` is an interface and so it cannot be instantiated. Further, a `Worker` is not `FastWorker`. A `FastWorker` is a `Worker`.

~~B.~~ `public FastWorker getWorker(){ return new FastWorker(); }`

C. `public Worker getWorker(){ return new FastWorker(); }`

> This is correct because the caller of this method will not know about the actual class of the object that is returned by this method. It is only aware of the `Worker`

interface. Hence, if you change the implementation of this method to return a different type of Worker, say `SuperFastWorker`, other classes do not have to change their code.

~~D.~~
```
public Worker getWorker(){
return new Worker();
}
```

This will not compile because `Worker` is an interface and so it cannot be instantiated.

Explanation:

Although not mentioned explicitly in the exam objectives, there are a few questions on this topic in the exam.

Back to Question without Answer

24. QID - 2.1204 : Working with Methods

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
  public static void main(String args[ ] ){
    Object a, b, c ;
    a = new String("A");
    b = new String("B");
    c = a;
    a = b;
    System.out.println(""+c);
  }
}
```

Correct Option is : B

~~**A.**~~ The program will print `java.lang.String@XXX`, where XXX is the memory location of the object a.

B. The program will print A

~~**C.**~~ The program will print B

~~**D.**~~ The program will not compile as a,b and c are of type Object.

> String also IS an Object ! You can always assign a subclass object to a super class reference without a cast.

~~**E.**~~ The program will print `java.lang.String@XXX`, where XXX is the hash code of the object a.

Explanation:

The variables a, b and c contain references to actual objects. Assigning to a reference only changes the reference value, and not the object pointed to by the reference. So, when `c = a` is executed c starts pointing to the string object containing A. and when `a = b` is executed, a starts pointing to the string object containing B but the object containing A still remains same and c still points to it. So the program prints A and not B.

The Object class's toString generates a string using: `getClass().getName() + '@' + Integer.toHexString(hashCode())`
But in this case, String class overrides the `toString()` method that returns just the actual string value.

Back to Question without Answer

25. QID - 2.1188 : Working with Methods

What should be the return type of the following method?

```
public RETURNTYPE methodX( byte by){
    double d = 10.0;
    return (long) by/d*3;
}
```

Correct Option is : C

~~**A.**~~ int

~~**B.**~~ long

C. double

~~**D.**~~ float

~~**E.**~~ byte

Explanation:

Note that the cast (long) applies to '`by`' not to the whole expression.
`( (long) by ) / d * 3;`
Now, division operation on long gives you a `double`. So the return type should be `double`.

Back to Question without Answer

26. QID - 2.1293 : Working with Methods

Which of the following correctly defines a method named stringProcessor that can be called by other programmers as follows: `stringProcessor(str1)` or `stringProcessor(str1, str2)` or `stringProcessor(str1, str2, str3)`, where `str1`, `str2`, and `str3` are references to Strings.

Correct Option is : B

~~**A.**~~
```
public void stringProcessor(...String){
}
```

B.
```
public void stringProcessor(String... strs){
}
```

~~**C.**~~
```
public void stringProcessor(String[] strs){
}
```

~~**D.**~~
```
public void stringProcessor(String a, String b, String c){
}
```

~~**E.**~~ Three separate methods need to be written.

Explanation:

To allow a method to take variable arguments of a type, you must use the . . . syntax:
`methodName( <type>... variableName);`
Remember that there can be only one vararg argument in a method. Further, the vararg argument must be the last argument.
So this is invalid: `stringProcessor( String... variableName, int age);`

but this is valid: `stringProcessor(int age, String... variableName);`

Though not important for this exam, it is good to know that within the method, the vararg argument is treated like an array:

```
public void stringProcessor(String... names){
    for (String n : names) {
        System.out.println("Hello " + n);
    }
}
```

Back to Question without Answer

27. QID - 2.1362 : Working with Methods

Which of the following statements are true?

Correct Options are : B E

~~**A.**~~ private keyword can never be applied to a class.

> private, protected and public can be applied to an Inner class.

B. synchronized keyword can never be applied to a class.

~~**C.**~~ synchronized keyword may be applied to a non-primitive variable.

> It can only be applied to a method or a block.

~~**D.**~~ final keyword can never be applied to a class.

> It can be applied to class, variable and methods.

E. A final variable can be shadowed in a subclass.

Explanation:

final keyword when applied to a class means the class cannot be subclassed, when applied to a method means the method cannot be overridden (it can be overloaded though) and when applied to a variable means that the variable is a constant.

Back to Question without Answer

28. QID - 2.1178 : Working with Methods

Which one of the following class definitions is/are a legal definition of a class that cannot be instantiated?

```
class Automobile{
   abstract void honk();  //(1)
}

abstract class Automobile{
   void honk();   //(2)
}

abstract class Automobile{
   void honk(){};   //(3)
}

abstract class Automobile{
   abstract void honk(){}   //(4)
}

abstract class Automobile{
   abstract void honk();   //(5)
}
```

Correct Options are : C E

~~**A.**~~ 1

> It will not compile as one of its method is abstract but the class itself is not abstract.

~~**B.**~~ 2

> It will not compile as the method doesn't have the body and also is not declared abstract.

C. 3

> This is a valid abstract class although it doesn't have any abstract method.

~~**D.** 4~~

> An abstract method cannot have a method body. {} constitutes a valid method body.

E. 5

> This is a valid abstract class

Explanation:

Here are some points to remember:

A class is uninstantiable if the class is declared `abstract`.
If a method has been declared as `abstract`, it cannot provide an implementation i.e. a method body even if empty (and the class containing that method must be declared `abstract`).
If a method is not declared `abstract`, it must provide a method body (the class can be `abstract` but not necessarily so).
If any method in a class is declared `abstract`, then the whole class must be declared `abstract`.

Back to Question without Answer

29. QID - 2.1074 : Working with Methods

Consider the following class definition:

```
public class TestClass{
   public static void main(){  new TestClass().sayHello(); }   //1
   public static void sayHello(){ System.out.println("Static Hello W
   public void sayHello() { System.out.println("Hello World "); }  /
}
```

What will be the result of compiling and running the class?

Correct Option is : D

~~**A.**~~ It will print `Hello World`.

~~**B.**~~ It will print `Static Hello World`.

~~**C.**~~ Compilation error at line 2.

D. Compilation error at line 3.

It will say, method sayHello() is already defined.

~~**E.**~~ Runtime Error.

Explanation:

You cannot have two methods with the same signature (name and parameter types) in one class.
Also, even if you put one `sayHello()` method in other class which is a subclass of

this class, it won't compile because you cannot override/hide a static method with a non static method and vice versa.

Back to Question without Answer

30. QID - 2.974 : Working with Methods

What will be the contents of s1 and s2 at the time of the println statement in the main method of the following program?

```
import java.util.*;
public class TestClass{
   public static void main(String args[]){
      Stack s1 = new Stack ();
      Stack s2 = new Stack ();
      processStacks (s1,s2);
      System.out.println (s1 + "    "+ s2);
   }
   public static void processStacks(Stack x1, Stack x2){
      x1.push (new Integer ("100")); //assume that the method push a
      x2 = x1;
   }
 }
```

Correct Option is : B

~~**A.**~~ [100] [100]

B. [100] []

~~**C.**~~ [] [100]

~~**D.**~~ [] []

Explanation:

. Primitives are always passed by value.

. Object "references" are passed by value. So it looks like the object is passed by reference but actually it is the value of the reference that is passed.

An example:

```
    Object o1 = new Object(); //Let us say, the object is stored at
memory location 15000.
        //Since o1 actually stores the address of the memory
location where the object is stored, it contains 15000.
```

Now, when you call someMethod(o1); the value 15000 is passed to the method.

Inside the method someMethod():

```
    someMethod( Object localVar) {
            /*localVar now contains 15000, which means it also
points to the same memory location where the object is stored.
            Therefore, when you call a method on localVar, it will
be executed on the same object.
            However, when you change the value of localVar itself,
for example if you do localVar=null,
            it then it starts pointing to a different memory
location. But the original variable o1 still
            contains 15000 so it still points to the same object. */
        }
```

If you need even more detailed explanation, please check
http://www.javaranch.com/campfire/StoryPassBy.jsp

This is what happens in this question.

You created two objects in main method:

```
s1 ------------> [ EMPTY ] STACK 1 OBJECT
```

s1 actually contains 15000 (say)

```
s2 ------------> [ EMPTY ] STACK 2 OBJECT
```

s2 actually contains 25000 (say)

inside the method `processStacks()` :

Step 1:

```
s1 ----> [ EMPTY ] STACK 1 OBJECT <----x1 Local variable
```

s1 and x1 both contain 15000 (say)

```
s2 ----> [ EMPTY ] STACK 2 OBJECT <----x2 Local variable
```

s2 and x2 both contain 25000 (say)

Step 2;

```
s1 -----> [ 100 ] STACK 1 OBJECT <----x1 Local variable
```

Because x1 is referring to the same memory location.

```
s2 -----> [ EMPTY ] STACK 2 OBJECT <---x2 Local variable
```

Step 3: After doing `x2 = x1`

```
s1 ---> [ 100 ] STACK 1 OBJECT <---- x1 and x2 Local variables
```

s1 and x1 both contain 15000 (say) and x2 now also contains 15000.

```
s2 ------------> [ EMPTY ] STACK 2 OBJECT
```

But s2 still contains 25000.

Note that it is the local variable x2 that is pointing to the same object as x1, which is s1 stack object. The original s2 (of the main method) is still pointing to the same object which is empty.

So when you come back to the main method, you print s1 (which has now 100) and s2 (which is still empty).

Back to Question without Answer

31. QID - 2.863 : Working with Methods

Consider the following code appearing in the same file:

```
class Data {
    private int x = 0, y = 0;
    public Data(int x, int y){
        this.x = x; this.y = y;
    }
}
public class TestClass {
    public static void main(String[] args) throws Exception {
        Data d = new Data(1, 1);
        //add code here
    }
}
```

Which of the following options when applied individually will change the Data object currently referred to by the variable d to contain 2, 2 as values for its data fields?

Correct Option is : E

~~**A.**~~ Add the following two statements :

```
d.x = 2;
d.y = 2;
```

> Note that x and y are private in class Data. Therefore, you cannot access these members from any other class.

~~**B.**~~ Add the following statement:

```
d = new Data(2, 2);
```

> This will create a new Data object and will not change the original Data object referred to be d.

~~C.~~ Add the following two statements:

```
d.x += 1;
d.y += 1;
```

> Note that x and y are private in class Data. Therefore, you cannot access these members from any other class.

~~D.~~ Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x.setInt(x);   this.y.setInt(y);
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

> x is primitive int.You cannot call any methods on a primitive. so this.x.setInt(...) or this.y.setInt(...) don't make any sense.

E. Add the following method to Data class:

```
public void setValues(int x, int y){
  this.x = x;   this.y = y;
}
```

Then add the following statement:

```
d.setValues(2, 2);
```

> This is a good example of encapsulation where the data members of Data class are private and there is a method in Data class to manipulate its data. Compare this approach to making x and y as public and letting other classes directly modify the values.

Back to Question without Answer

32. QID - 2.831 : Working with Methods

What will the following program print when run?

```
public class ChangeTest {

    private int myValue = 0;

    public void showOne(int myValue){
        myValue = myValue;
        System.out.println(this.myValue);
    }

    public void showTwo(int myValue){
        this.myValue = myValue;
        System.out.println(this.myValue);
    }
    public static void main(String[] args) {
        ChangeTest ct = new ChangeTest();
        ct.showOne(100);
        ct.showTwo(200);
    }
}
```

Correct Option is : C

~~**A.**~~ 0 followed by 100.

~~**B.**~~ 100 followed by 100.

C. 0 followed by 200.

~~**D.**~~ 100 followed by 200.

Explanation:

There are a couple of important concepts in this question:

1. Within an instance method, you can access the current object of the same class using '`this`'. Therefore, when you access `this.myValue`, you are accessing the instance member `myValue` of the `ChangeTest` instance.

2. Within the `showOne()` method, there are two variables accessible with the same name myValue. If you declare a local variable (or a method parameter) with the same name as the instance field name, the local variable "shadows" the member field. One is the method parameter and another is the member field of ChangeTest object. Ideally, you should be able to access the member field in the method directly by using the name of the member (in this example, `myValue`). However, because of shadowing, when you use myValue, it refers to the local variable instead of the instance field.

Therefore, when you do : `myValue = myValue;` you are actually assigning the value contained in method parameter myValue to itself. You are not changing the member field myValue. Hence, when you do `System.out.println(this.myValue);` in the next line, it prints 0.

Now, in `showTwo()`, you are assigning the value contained in `myValue` (i.e. 200) to `this.myValue`, which is the instance member. Therefore, in the next line, when you print `this.myValue`, it prints 200.

Back to Question without Answer

33. QID - 2.1294 : Working with Methods

What will the code shown below print when run?

```
public class TestClass{
    static class Wrapper{
        int w = 10;
    }

    static Wrapper changeWrapper(Wrapper w){
        w = new Wrapper();
        w.w += 9;
        return w;
    }


    public static void main(String[] args){
        Wrapper w = new Wrapper();
        w.w = 20;
        changeWrapper(w);
        w.w += 30;
        System.out.println(w.w);
        w = changeWrapper(w);
        System.out.println(w.w);
     }
}
```

Correct Options are : B F

~~**A.** 9~~

B. 19

~~**C.** 30~~

~~**D.**~~ 20

~~**E.**~~ 29

F. 50

Explanation:

Remember that when you pass an object in a method, only its reference is passed by value. So when `changeWrapper()` does `w = new Wrapper();` and then `w.w +=9;` it does not affect the original wrapper object that was passed to this method. Therefore, it prints `50`.

Calling `w = changeWrapper(w);` replaces the original Wrapper object with the one created in the `changeWrapper(w);` method. Therefore, in the second print statement, it prints `19`.

Back to Question without Answer

Working with Methods - Access Modifiers

Exam Objectives -

Apply access modifiers

01. QID - 2.966

Select the correct order of restrictiveness for access modifiers...
(First one should be least restrictive)

Select 1 option

A. public < protected < package (i.e. no modifier) < private

B. public < package (i.e. no modifier) < protected < private

C. public < protected < private < package (i.e. no modifier)

D. protected < package (i.e. no modifier) < private < public

E. depends on the implementation of the class or method.

Check Answer

02. QID - 2.981

For object o1 of class A to access a member(field or method) of object o2 of class B, when the member has no access modifier, class B must be...

Select 1 option

A. a subclass of A

B. in the same package as A is in.

C. a Super class of A

D. a subclass but may not be in the same package.

E. in the same package and must be a Subclass of A.

Check Answer

03. QID - 2.1217

Consider the following classes in one file named A.java...

```
abstract class A{
   protected int m1(){ return 0; }
}
class B extends A{
   int m1(){ return 1; }
}
```

Which of the following statements are correct...

Select 1 option

A. The code will not compile as you cannot have more than 1 class in 1 file.

B. The code will not compile because class B does not override the method m1() correctly.

C. The code will not compile as A is an abstract class.

D. The code will not compile as A does not have any abstract method.

E. The code will compile fine.

Check Answer

04. QID - 2.1053

Compared to public, protected and private accessibility, default accessibility is....

Select 1 option

A. Less restrictive than public

B. More restrictive than public, but less restrictive than protected.

C. More restrictive than protected, but less restrictive than private.

D. More restrictive than private.

E. Less restrictive than protected from within a package, and more restrictive than protected from outside a package.

Check Answer

05. QID - 2.1330

Which of the following access control keywords can be used to enable all the subclasses to access a method defined in the base class?

Select 2 options

A. public

B. private

C. protected

D. No keyword is needed.

Check Answer

06. QID - 2.1275

How can you declare 'i' so that it is not visible outside the package `test`.

```
package test;
public class Test{
   XXX int i;
   /*  irrelevant code */
}
```

Select 2 options

A. private

B. public

C. protected

D. No access modifier

E. friend

Check Answer

07. QID - 2.1263

Consider the following code in TestClass.java file:

```
package p;
private class TC extends java.util.HashMap{
   public TC(){
      super(100);
      System.out.println("TC created");
   }
}
public class TestClass extends TC{
   public TestClass(){
      System.out.println("TestClass created");
   }
   public static void main(String[] args){ new TestClass(); }
}
```

What will be the output when TestClass is run?

Select 1 option

A. `"TestClass created"` as well as `"TC created"`.

B. It will not compile because `HashMap` is a final class.

C. Only `"TestClass created"` will be printed.

D. Only `"TC created"` will be printed.

E. None of the above are correct.

Check Answer

08. QID - 2.978

Given the following definition of class, which member variables are accessible from OUTSIDE the package com.enthu.qb?

```
package com.enthu.qb;
public class TestClass{
   int i;
   public int j;
   protected int k;
   private int l;
}
```

Select 2 options

A. Member variable i.

B. Member variable j.

C. Member variable k.

D. Member variable k, but only for subclasses.

E. Member variable l.

Check Answer

Working with Methods - Access Modifiers (Answered)

01. QID - 2.966 : Working with Methods - Access Modifiers

Select the correct order of restrictiveness for access modifiers...
(First one should be least restrictive)

Correct Option is : A

A. public < protected < package (i.e. no modifier) < private

> That's right, protected is less restrictive than package.

~~**B.**~~ public < package (i.e. no modifier) < protected < private

~~**C.**~~ public < protected < private < package (i.e. no modifier)

> The default accessibility is more restrictive than protected, but less restrictive than private.

~~**D.**~~ protected < package (i.e. no modifier) < private < public

~~**E.**~~ depends on the implementation of the class or method.

Explanation:

Members with default accessibility are only accessible within the class itself and from classes in the same package.
Protected members are in addition accessible from subclasses. Members with private accessibility are only accessible
within the class itself.

Back to Question without Answer

02. QID - 2.981 : Working with Methods - Access Modifiers

For object o1 of class A to access a member(field or method) of object o2 of class B, when the member has no access modifier, class B must be...

Correct Option is : B

~~**A.**~~ a subclass of A

> It cannot access the member if it is not in the same package.

B. in the same package as A is in.

> This is correct. B may or may not be a Subclass of A.

~~**C.**~~ a Super class of A

~~**D.**~~ a subclass but may not be in the same package.

> Has to be in the same package.

~~**E.**~~ in the same package and must be a Subclass of A.

Explanation:

No access modifier means "default" access which means only classes of the same package can access it.
Note that "default" is not a valid access specifier. The absence of any access modifier means default access.

Back to Question without Answer

03. QID - 2.1217 : Working with Methods - Access Modifiers

Consider the following classes in one file named A.java...

```
abstract class A{
   protected int m1(){ return 0; }
}
class B extends A{
   int m1(){ return 1; }
}
```

Which of the following statements are correct...

Correct Option is : B

~~**A.** The code will not compile as you cannot have more than 1 class in 1 file.~~

> You can. But only one class can be public.

B. The code will not compile because class B does not override the method m1() correctly.

> The overriding method cannot decrease the accessibility.

~~**C.** The code will not compile as A is an abstract class.~~

~~**D.** The code will not compile as A does not have any abstract method.~~

> You need not have any 'abstract' method to make a class abstract. Putting 'abstract' keyword is enough.

~~**E.** The code will compile fine.~~

Explanation:

The concept here is that an overriding method cannot make the overridden method more private.
The access hierarchy in increasing levels of accessibility is:
private->'no modifier'->protected->public (public is accessible to all and private is accessible to none except itself.)
Here, class B has no modifier for m1() so it is trying to reduce the accessibility of protected to default.
'protected' means the method will be accessible to all the classes in the same package and all the subclasses (even if the subclass is in a different package).
No modifier (which is the default level) means the method will be accessible only to all the classes in the same package. (i.e. not even to the subclass if the subclass is in a different package.)

Back to Question without Answer

04. QID - 2.1053 : Working with Methods - Access Modifiers

Compared to public, protected and private accessibility, default accessibility is....

Correct Option is : C

~~**A.**~~ Less restrictive than public

> public is least restrictive.

~~**B.**~~ More restrictive than public, but less restrictive than protected.

C. More restrictive than protected, but less restrictive than private.

> The default accessibility is more restrictive than `protected`, but less restrictive than `private`. Members with default accessibility are only accessible within the class itself and from other classes in the same package. `protected` members are in addition accessible from subclasses in any other package as well. Members with `private` accessibility are only accessible within the class itself.

~~**D.**~~ More restrictive than private.

> private is most restrictive.

~~**E.**~~ Less restrictive than protected from within a package, and more restrictive than protected from outside a package.

Explanation:

The correct order :
public < protected < package (or default) < private

(here, public is least restrictive and private is most restrictive.)

Back to Question without Answer

05. QID - 2.1330 : Working with Methods - Access Modifiers

Which of the following access control keywords can be used to enable all the subclasses to access a method defined in the base class?

Correct Options are : A C

A. public

> It will allow everybody to access the method.

~~**B.**~~ private

> It will not allow any other class to access the method.

C. protected

> It will allow the subclasses and the classes in same package to access the method.

~~**D.**~~ No keyword is needed.

> It will allow only the classes in same package to access the method. So Subclasses outside the package will not have access.

Back to Question without Answer

06. QID - 2.1275 : Working with Methods - Access Modifiers

How can you declare 'i' so that it is not visible outside the package `test`.

```
package test;
public class Test{
   XXX int i;
   /*  irrelevant code */
}
```

Correct Options are : A D

A. private

> Note that the question does not require that 'x' should be accessible from test package. So private is fine.

~~**B.**~~ public

> Marking it public will make it accessible from all classes in all packages.

~~**C.**~~ protected

> It will make it available to a subclass even if the subclass is in a different package.

D. No access modifier

~~**E.**~~ friend

> There is no such modifier in Java.

Back to Question without Answer

07. QID - 2.1263 : Working with Methods - Access Modifiers

Consider the following code in TestClass.java file:

```
package p;
private class TC extends java.util.HashMap{
   public TC(){
      super(100);
      System.out.println("TC created");
   }
}
public class TestClass extends TC{
   public TestClass(){
      System.out.println("TestClass created");
   }
   public static void main(String[] args){ new TestClass(); }
}
```

What will be the output when TestClass is run?

Correct Option is : E

~~**A.**~~ `"TestClass created"` as well as `"TC created"`.

~~**B.**~~ It will not compile because `HashMap` is a final class.

> `HashMap` is not a final class.

~~**C.**~~ Only `"TestClass created"` will be printed.

~~**D.**~~ Only `"TC created"` will be printed.

E. None of the above are correct.

Explanation:

The file will not compile because `TC` is a top level class and `private` is not a valid access modifier for a top level class. `private` can be applied to an inner class.

Back to Question without Answer

08. QID - 2.978 : Working with Methods - Access Modifiers

Given the following definition of class, which member variables are accessible from OUTSIDE the package com.enthu.qb?

```
package com.enthu.qb;
public class TestClass{
   int i;
   public int j;
   protected int k;
   private int l;
}
```

Correct Options are : B D

~~**A.**~~ Member variable i.

> No modifier means package(or default access) and is only accessible inside the package.

B. Member variable j.

> public things (classes, methods and fields) are accessible from anywhere.

~~**C.**~~ Member variable k.

> Only if the accessing class is a subclass of TestClass.

D. Member variable k, but only for subclasses.

> protected things (methods and fields) can be accessed from within the package and from subclasses

~~**E.**~~ Member variable l.

private things are accessible only from the class that has it.

Explanation:

`public > protected > package (i.e. no modifier) > private`
where `public` is least restrictive and `private` is most restrictive.

Remember:
`protected` is less restrictive than package access. So a method(or field) declared as protected will be accessible from a subclass even if the subclass is not in the same package.
The same is not true for package access.
A top level class can only have either `public` or no access modifier but a method or field can have all the four. Note that `static`, `final`, `native` and `synchronized` are not considered as access modifiers.

Back to Question without Answer

Exam Refresher

01 - Java Basics

1. The standard main method that is used by the JVM to execute a class must be public and static. It must have the following signature:
   `public static void main(String[] args)`
   It may throw any exception.
2. When a program is called with no arguments, the args array will **NOT** be null. It will be of length zero. `args[0]` is the first argument. It is **NOT** the name of the program or class.
3. A valid java identifier is composed of a sequence of java letters (including _ and $) and digits, the first of which must be a letter. ~~`8val`~~is invalid. `_val8`is valid
4. The order of keywords for a static import must be **`import static...`** .
   You can either import all the static member using `import static java.lang.Integer.*` or one specific member using `import static java.lang.Integer.MAX_VALUE;`
   You must specify the full package name of the class that your are importing (just like the regular import statement). So, `import static Integer.*;`is wrong.
5. You cannot use `this` within a static method because static methods do not belong to any instance of the class but to the class itself.
6. Arrays are indexed from 0. So, for `int[] ia = { 1, 2, 3 }; ia[0] is 1.`

---

01 - Java Basics - Garbage Collection

Although not mentioned explicitly in the objectives, the exam has a few basic questions on garbage collection. All you need to know is:

1. An object can be made eligible for garbage collection by making sure there are no references pointing to that object.

2. You cannot directly invoke the garbage collector. You can suggest the JVM to perform garbage collection by calling System.gc();

---

02 - Working with Java Data Types - String, StringBuilder

1. String is immutable while StringBuilder and StringBuffer are not. So when you call String.replace() or String.concat(), it returns a new String object. The original String remains the same. However, if there is no change, then the same String object is returned.
2. StringBuilder has methods named insert, delete, and append. These methods modify the StringBuilder itself. Further, these methods return the same StringBuilder object so that multiple calls can be chained:
   `sb.append("a").append("asdf").insert(2, "asdf")`
   is equivalant to:
   ```
   sb.append("a");
   sb.append("asdf");
   sb.insert(2, "asdf");
   ```
3. The method `substring()` in StringBuilder/StringBuffer returns a String (and not a reference to itself, unlike append, insert, and delete). So another StringBuilder method cannot be chained to it. For example, the following is **not** valid:
   ~~`sb.append("a").substring(0, 4).insert(2, "asdf");`~~
4. substring methods of String work as follows:

   `public String substring(int beginIndex)` - Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.
   Examples:
   ```
   "unhappy".substring(2) returns "happy"
   "Harbison".substring(3) returns "bison"
   "emptiness".substring(9) returns "" (an empty string)
   ```

`public String substring(int beginIndex, int endIndex)` - Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex-beginIndex.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
"smiles".substring(1, 7) will throw
java.lang.IndexOutOfBoundsException (in practice, it throws
StringIndexOutOfBoundsException, which is a subclass of
IndexOutOfBoundsException)
```

5. `public StringBuilder delete(int start, int end)`
   Removes the characters in a substring of this sequence. The substring begins at the specified start and extends to the character at index end - 1 or to the end of the sequence if no such character exists. If start is equal to end, no changes are made.
6. + operator is overloaded for String. Therefore,

   ```
   "" + 5 + 6 => "5"+6 => "56"
   5 + "" +6 => "5"+6 => "56"
   5 + 6 +"" => 11+"" => "11"
   5 + 6 => 11 => "11"
   ```

   and also for primitive wrappers. Therefore,

   ```
   Integer x = 10;
   Integer y = 10;
   //Valid. Prints 21
   System.out.println(y+1+x);

   Object o1 = new Object();
   //Invalid. Will not compile.
   System.out.println(x+o1);
   ```
7. `StringBuilder.ensureCapacity(int minimumCapacity)` - Ensures that the capacity is at least equal to the specified minimum.

02 - Working with Java Data Types - Variables and Objects

1. ~~`bool`~~ is not a valid keyword. `boolean` is.
2. `String, StringBuilder, and StringBuffer`are final classes. `java.lang.System`is final as well.
3. Wrapper classes for primitives (`java.lang.Boolean, java.lang.Integer, java.lang.Long, java.lang.Short`etc.) are also final and so they cannot be extended.
4. `java.lang.Number`, however, is not final. Byte, Short, Integer, Long, Float, and Double extend Number. Character does not.
   Therefore, the declaration `class ImaginaryNumber extends Number`, would be valid. But `class ReverseString extends String` is not.

---

03 - Using Operators and Decision Constructs

1. Only String, byte, char, short, int, and enum values can be used as types of a switch variable. float, double, and boolean are not valid.
2. Strings in switch statement are case sensitive. Therefore, `case "A":`is not same as `case "a":`.
3. `switch(str)`will throw `NullPointerException`if str is `null`. It is ok if str is "".
4. + operator is overloaded for String. It concatenates two Strings. `String x = "1"; String y = "2"; x+y` returns a new String instance.
5. `||`and `&&`perform short circuit evaluation, while `&`and `|`do not. Which means, if you use the || and && forms, Java will not bother to evaluate the right-hand operand if the result of the expression can be known by just evaluating the left hand operand.
   Example 1:

```
boolean b = true;
if(b || foo.timeConsumingCall()) {
//entered here without calling timeConsumingCall()
}
```

Example 2:

```
String s = null;
/*No NullPointerException because string.isEmpty() is not
called.
If you use & instead of && , s.isEmpty will be called and a
NullPointerException will be thrown.*/
if(s != null && s.isEmpty()) {
...
}
```

6. You may get a few very simple questions about operator preference. Simple school math trick of BODMAS can be used to evaluate the expressions.

   B Brackets first
   O Orders (ie Powers and Square Roots, etc.)
   DM Division and Multiplication (left-to-right)
   AS Addition and Subtraction (left-to-right)

   Therefore, `1 + 2 + 3 * 4; is 15 and 2 * 3 + 4 is 10.`

---

04 - Creating and Using Arrays

1. The correct syntax to access any element within an array is to use the square brackets - [ ]. Thus, to access the first element in an array, you would use array[0].

   `int[][] iaa = { { 1, 2 }, { 1, 2, 3}, null } iaa[0] is {1, 2}. iaa[0][0] is 1. iaa[1][3] is ArrayIndexOutOfBoundsException, iaa[2][0] is NullPointerException.`

2. Arrays are just like regular Objects and arrays of different types have different class names. For example, the class name of an int array is **[I** and the class name for an array of int array is **[[I**.
3. `int[] ia = { 1, 2 }; ia.getClass().isArray()` will return true;
4. `System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length)` method copies an array from the specified source array, beginning at the specified position, to the specified position of the destination array. The last parameter is the number of elements that you want to copy.
5. An ArrayList resized dynamically at run time as per the situation. An array cannot be resized once created. This reduces the amount of boiler plate code that is required to do the same task using an array.
6. ArrayList is a subclass of AbstractList.

---

05 - Using Loop Constructs

1. A continue statement can occur in and only in a for, while or do-while loop. A continue statement means - Forget about the rest of the statements in the loop and start the next iteration.
   So, `for (int i=1 ; i<5 ; i++) continue;` just increments the value of i up to 5 because of i++.
   `for (int i=0 ; ; i++) break;` iterates only once because of the break so the value of i remains 0.

---

06 - Constructors

1. **Default Constructor**:
   If a class contains no constructor declarations, then a default constructor that takes no parameters is automatically provided. The default constructor takes no parameters and simply invokes the superclass constructor with no arguments. A compile-time error occurs if a default constructor is provided by the compiler but

the superclass does not have an accessible constructor that takes no arguments. A default constructor has no throws clause.
2. Constructors are not inherited.
3. If a derived class constructor doesn't explicitly call the super class's constructor, the compiler automatically inserts super();

---

06 - Encapsulation

1. Data fields should be private and there should be public methods to manipulate them.
2. When a class is properly encapsulated, only the members that are part of its public API are publicly accessible to other classes. Rest of the class members are private or protected.
3. An invariant means a certain condition that constrains the state stored in the object. Area and side fields in the example below form an invariant. The invariant enforces that the value of area must not be anything other than side*side.
4. For example:

```
public class Square {
 private double side = 0;
 private double area;

 public Square(double length){
 this.side = length;
 //calculate area and keep it
 calculateArea();
 }

 public double getSide() {
 return side;
 }

 public void setSide(double side) {
 this.side = side;
 /* make sure area is consistent with the side
 so recalculate area */
```

```
 calculateArea();
 }

 private void calculateArea(){
 this.area = this.side*this.side;
 }

 public double getArea() {
 return area;
 }

 /*
 Get rid of setArea because that would make it
 inconsistent with side. area and side form an invariant.
 public void setArea(double d){
 this.area = d;
 }
 */

}
```

---

06 - Overloading methods

1. Overloading means method name is same but method parameters are different. Thus, overloaded methods have different method signatures.
2. When you have multiple overloaded methods that can be applicable for a method call, the most specific one is use.
   Example:

```
void m1(int x) {
  System.out.println(" m1 int");
}

void m1(double x){
  System.out.println(" m1 double");
}

void m1(String x){
```

```
  System.out.println(" m1 String");
}
```

When you call m1(1.0), m1(double x) will be used and not m1(int).

---

06 - Working with Methods

1. Although not mentioned in exam objectives, you may get a question that talks about Program to an interface concept in the exam. It means, the caller of this method does not know about the actual class of the object that it uses. It is only aware of the interface that the object implements. Hence, if you give it a different object that implements the same interface, you do not have to change its code.

```
interface Worker {
   void performWork();
}

class FastWorker implements Worker {
   public void performWork(){ }
}

public Worker getWorker(){
   return new FastWorker();
}
```

   The caller of `getWorker`will not know about the actual class of the object that is returned by this method. It is only aware of the Worker interface. Hence, if you change the implementation of `getWorker` method to return a different type of Worker, say `SuperFastWorker`, other classes do not have to change there code
2. Within an instance method, you can access the current object of the same class

using '`this`'.

3. If you declare a local variable (or a method parameter) with the same name as the instance field name, the local variable "shadows" the member field.
   Example:

```
class X{
 int a = 10;
 void m1(){
   int a = 20;
   // Will print 20
   System.out.println( a);
   // Will print 10
   System.out.println( this.a);
   }
}
```

4. To call `myMethod(int m, Object p, double d)`, you must pass exactly three parameters. If you try to pass less (or more) number of parameters, the code will not compile. Method parameters are not assigned default values.
   It is possible to declare a method that can take variable number of parameters.
   Example:

```
public static void someMethod(Object... params){
   System.out.println(params.length);
}
```

   You can call this method by passing any number of parameters. In this case, calling someMethod() without any parameter will print 0. i.e. the length of params array will be 0. params will NOT be null.
5. class level fields means static fields. They can be accessed from anywhere in the class.
   instance level fields means the instance fields and they can be accessed only from instance methods in the class.
6. If you declare a field to be final, it must be explicitly initialized by the time the creation of an object of the class is complete. So you can either initialize it immediately:

`private final double ANGLE = 0;` or you can initialize it in the constructor or an instance block.

---

06 - Working with Methods - Access Modifiers

1. The order of access modifiers is:
   public < protected < no access modifier (aka package access or default access) < private
   (here, public is least restrictive and private is most restrictive.)
2. Members with private accessibility are only accessible within the class itself. Members with default accessibility are only accessible within the class itself and from classes in the same package.Protected members are, in addition to default access, accessible from any subclass in any package .
3. A method(or field) declared as protected will be accessible from a subclass even if the subclass is not in the same package.
4. A class can only have either public or no access modifier but a method or field can have all the four.
5. static, final, native and synchronized are not considered as access modifiers.

---

07 - Working with Inheritance

1. Polymorphism means that it is always the class of the actual object at run time (and not the class of the reference variable that a variable points to) that determines which method will be called at run time. The concept of polymorphism doesn't apply to private methods or static methods because these methods are never inherited.
   Example:

```
class A{
 public void m1(){
   System.out.println(" A");
```

```
 }
}

class B extends A{
  Cat c;
  public void m1(){
    System.out.println(" B");
  }
}

B b = new B();
A a = new B();

// will print B
b.m1();

// will also print B because a refers to an object of class B.
a.m1();
```

2. Inheritance defines an is-a relation , so B is-a A because B extends A.
3. If B implements I, then B is-a I.
4. Aggregation and Composition define has-a relationship. In the example above, B has-a Cat;
5. An overriding method cannot throw an exception that is a super class of the exception thrown by the overridden method.
   FileNotFoundException is a subclass of IOException. Therefore, sub class's doStuff() cannot throw IOException if the base class's doStuff throws only FileNotFoundException.
   Think of it this way:

```
/*Will this work? No, because an IOException is NOT a
FileNotFoundException. */
FileNotFoundException fne = new IOException();



/* Will this work? Yes, because a FileNotFoundException is an
```

```
IOException. */
IOException ioe = new FileNotFoundException();
```

Therefore, overriding method must not throw an exception that cannot be assigned to a variable whose class is the class of the overridden method's exception.

6. Having ambiguous fields or methods does not cause any problems by themselves but referring to such fields/methods in an ambiguous way will cause a compile time error.

   Example:

```
interface T1{
 int T = 10;
 void m1();
}

interface T2{
 int T = 20;
 void m1();
}

class X implements T1, T2{
  public void m1() {
    System.out.println("x");
  }
}

X x = new X();
x.m1(); //This is valid. prints x
x.T; //This will NOT compile because x.T is ambiguous.
System.out.println((T1) x). T); //valid. prints 10
System.out.println((T2) x). T); // valid. prints 20
```

7. **Covariant Returns:** In case of overriding, the return type of the overriding method may be a subclass of the return type of the overridden method. However, the return type of the overriding method must match exactly to the return type of

the overridden method if the return type is a primitive.

8. While accessing a method or variable, the compiler will only allow you to access a method or variable that is visible through the class of the reference.
   Example:

```
interface Flyer{
 String getName();
}

class Bird implements Flyer{
  ...
  public String getName(){
    return name;
  }
  public String getNativeOf(){
    return nativeOf;
  }
}

Flyer f = new Bird();
f.getName(); //valid
f.getNativeOf(); //will not compile because the class of referen

((Bird) f).getNativeOf(); //valid because class of the object re
```

---

08 - Handling Exceptions

1. The base class of all exceptions is `java.lang.Throwable`. `java.lang.Error`and `java.lang.Exception`are the only two subclasses of Throwable.
2. Note: The terminology "thrown by the JVM" and "thrown by the application/programatically" is imprecise but is used by popular books. If it helps, you can think of the exception categories as "thrown implicitly" and "thrown explicitly". An exception is that is thrown even when there is no `throws` statement, is said to be thrown implicitly. For example, calling a method on null

will cause a `NullPointerException` to be thrown automatically, even through there is no throws statement. On the other hand, a code may throw an exception explicily by using the `throw` statement. For example, a method code might check an argument for validity and if it finds the argument inappropriate, it may throw an exception by executing `throw new IllegalArgumentException();`.

A quick way to determine who should throw an exception is to see if the exception extends `java.lang.Error`. Instances of `java.lang.Error` are thrown only by the JVM. Some common Errors are `java.lang.AssertionError, java.lang.OutOfMemoryError, and java.lang.NoClassDefFoundError`

Generally, `RuntimeException`s are also thrown by the JVM. However, it is ok for an application code to throw a `RuntimeException` if it makes sense for the application to throw a `RuntimeException` in a given situation.

You should know about the following common exception classes:

1. **`IndexOutOfBoundsException extends RuntimeException:`** Usually thrown by the JVM. Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range. Applications can subclass this class to indicate similar exceptions. `ArrayIndexOutOfBoundsException and StringIndexOutOfBoundsException` both extend `IndexOutOfBoundsException`.
2. **`SecurityException extends RuntimeException:`** Usually thrown by the JVM. It is thrown by the security manager upon security violation. For example, when a java program runs in a sandbox (such as an applet) and it tries to use prohibited APIs such as File I/O, the security manager throws this exception.

3. **`ClassCastException extends RuntimeException:`** Usually thrown by the JVM. Thrown to indicate that the code has attempted to cast an object to a subclass of which it is not an instance. For example, the following code

generates a `ClassCastException`:

```
Object x = new Integer(0);
System.out.println((String)x);
```

4. **`NullPointerException extends RuntimeException:`** Usually thrown by the JVM. Thrown when an application attempts to use `null` in a case where an object is required. These include-
   1. Calling the instance method of a null object.
   2. Accessing or modifying the field of a null object.
   3. Taking the length of null as if it were an array.
   4. Accessing or modifying the slots of null as if it were an array.
   5. Throwing null as if it were a Throwable value.

   Applications should throw instances of this class to indicate other illegal uses of the null object.
5. **`IllegalArgumentException`** extends **`RuntimeException:`** Usually thrown by the application if a parameter passed to a method is not valid.
6. **`IllegalStateException extends RuntimeException:`** Usually thrown by the application. Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation.

3. A try without resources must have either a catch or a finally. It may have both as well. A try with resources may avoid catch or finally altogether.
   Thus, the following constructs are valid:

1.

```
try{
}
catch(Exception e){ }// no finally
```

2.

```
try{
}
finally{ } // no catch
```

3.

```
try{
}
catch(Exception e){ }
finally{ }
```

4. A catch can catch multiple exceptions:

```
try{
}
catch(Exception1|Exception2|Exception3 e){
/*
Invalid because when you catch multiple exceptions
in the same catch, e is implicitly final
*/
e = new Exception();
}
```

5. `try(FileInputStream fis = new FileInputStream("a.txt") ){`

`} //No catch or finally necessary.`

4. `System.out.println(exception);` Prints only what is returned by `exception.getMessage()`. Doesn't print the stack trace.
   `exception.printStackTrace();` prints the stack trace to std err

About the Author

Hanumant Deshmukh has over 15 years of professional Java experience as an architect, developer, and teacher. Besides developing enterprise level projects for several wall street firms over the years, he has helped Enthuware in developing thousands of mock exam questions for various Java related certifications such as JSP/Servlet, EJB, and JPA. **Hanumant** is also very active as a Public Interest Lawyer and maintains hanumant.com for the benefit of law students in India.