Question 1.
Which of the following are valid code snippets appearing in a method:

You had to select 3 options.

1. int a = b = c = 100;
2. int a, b, c; a = b = c = 100;
3. int a, b, c=100;
4. int a=100, b, c;
5. int a= 100 = b = c;

Question 2.
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  );
   }
}
```

1. It will not compile.
2. It will print 10 and then 10
3. It will print 20 and then 20
4. It will print 10 and then 20
5. It will print 20 and then 10

Question 3.
What will the following code print when compiled and run?

```
public class DaysTest{
    static String[] days = {"monday", "tuesday", "wednesday", "thursday", "friday",
"saturday", "sunday" };

    public static void main(String[] args) {

        int index = 0;
        for(String day : days){

            if(index == 3){
                break;
            }else {
                 continue;
            }
            index++;
            if(days[index].length()>3){
                days[index] = day.substring(0,3);
            }
        }
        System.out.println(days[index]);
    }
}
```

1. wed
2. thu
3. fri
4. It will not compile.
5. It will throw an exception at run time.

Question 4.
The following code snippet will print 4.

```
int i1 = 1, i2 = 2, i3 = 3;
int i4 = i1 + (i2=i3 );
System.out.println(i4);
```

1. True
2. False

Question 5.
Which of these expressions will return true?

1. "hello world".equals("hello world")
2. "HELLO world".equalsIgnoreCase("hello world")
3. "hello".concat(" world").trim().equals("hello world")
4. "hello world".compareTo("Hello world") < 0
5. "Hello world".toLowerCase( ).equals("hello world")

Question 6.
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 be the value of 'harry' if the program is run from the command line:
`java TestClass 111 222 333`

1. 111
2. 222
3. 333
4. It will throw an ArrayIndexOutOfBoundsException
5. None of the above.

Question 7.
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") ;
      }
   }
}
```

1. The program will fail to compile.
2. The program will throw a NullPointerException when run with zero arguments.
3. The program will print no arguments when called with zero arguments and 1 arguments when called with one argument.
4. The program will print no arguments and 2 arguments when called with zero and one arguments.
5. The program will print no arguments and 3 arguments when called with zero and one arguments.

Question 8.
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);
    }
}
```

1. 3
2. 4
3. 5
4. 6
5. 7

Question 9.
Consider the following class definition:

```
public class TestClass{
   public static void main(String[] args){  new TestClass().sayHello(); }   //1
   public static void sayHello(){ System.out.println("Static Hello World"); }  //2
   public void sayHello() { System.out.println("Hello World "); }  //3
}
```

What will be the result of compiling and running the class?

1. It will print Hello World.
2. It will print Static Hello World.
3. Compilation error at line 2.
4. Compilation error at line 3.
5. Runtime Error.

Question 10.
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");
     }
  }
}
```

1. It will print hello 6 times.
2. It will not compile.
3. It will print hello 2 times.
4. It will print hello 5 times.
5. It will print hello 4 times.

Question 11.
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 ?

1. It will not compile.
2. It will throw ArrayIndexOutOfBoundsException when run.
3. It will print 1.
4. It will print 3.
5. It will print 4

Question 12.
Identify the correct statements about ArrayList?

You had to select 3 options.

1. ArrayList extends java.util.AbstractList.
2. It allows you to access its elements in random order.
3. You must specify the class of objects you want to store in ArrayList when you declare a variable of type ArrayList.
4. ArrayList does not implement RandomAccess.
5. You can sort its elements using Collections.sort() method.

Question 13.
What will the following code print when compiled and run?

```
public class TestClass{
    public static void main(String[] args) {

        System.out.println(getMsg((char)10));

    }

    public static String getMsg(char x){
        String msg = "Input value must be ";
        msg = msg.concat("smaller than X");
        msg.replace('X', x);
        String rest = " and larger than 0";
        msg.concat(rest);
        return msg;
    }
}
```

1. Input value must be smaller than X and larger than 0
2. Input value must be smaller than 10 and larger than 0
3. Input value must be smaller than X
4. Input value must be smaller than 10

Question 14.
Which statements about the following code contained in BankAccount.java are correct?

```
interface Account{
  public default String getId(){
     return "0000";
  }
}

interface PremiumAccount extends Account{
  public String getId();
}

public class BankAccount implements PremiumAccount{
  public static void main(String[] args) {
      Account acct = new BankAccount();
      System.out.println(acct.getId());
  }
}
```

1. It will print 0000 when run.
2. It will compile if class BankAccount provides an implementation for getId method.
3. It will not compile unless interface PremiumAccount is marked abstract.
4. It will compile if getId method in PremiumAccount is replaced with:
   public String getId(){ super.getId(); }
5. It will compile if getId method in PremiumAccount is replaced with:
   public default String getId(){ super.getId(); }

Question 15.
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();
        }
    }
}
```

You had to select 2 options.

1. add throws IOException to the main method.
2. add throws IOException to PortConnector constructor.
3. add throws IOException to the main method as well as to PortConnector constructor.
4. Change RuntimeException to java.io.IOException.
5. add throws Exception to PortConnector constructor and change catch(RuntimeException re) to catch(Exception re) in the main method.

Question 16.
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;
    }

}
```

1. 13.0
2. 4.0
3. 4
4. The code will not compile.

Question 17.
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.print("catch ");
      }
      finally   {
         System.out.print("finally ");
      }
      System.out.print("out ");
   }
   public static void amethod(){ }
}
```

1. try finally
2. try finally out
3. try out
4. catch finally out
5. It will not compile because amethod() does not throw any exception.

Question 18.
What will the following code print when compiled and run?

```
import java.util.*;
public class TestClass {
    public static void main(String[] args) throws Exception {
        List  al = new ArrayList(); //1
        al.add(111); //2
        System.out.println(al.get(al.size()));  //3
     }
}
```

1. It will not compile.
2. It will throw an exception at run time because of line //1
3. It will throw an exception at run time because of line //2
4. It will throw an exception at run time because of line //3
5. null.

Question 19.
Which of these statements are valid when occurring by themselves in a method?

You had to select 3 options.

1. while ( ) break ;
2. do { break ; } while (true) ;
3. if (true) { break ; } (When not inside a switch block or a loop)
4. switch (1) { default : break; }
5. for (; true ;) break ;

Question 20.
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);
       }
}
```

1. It will not compile.
2. It will throw an exception at runtime.
3. 800
4. 200
5. 400

Question 21.
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 NewException(); }

    public static void m3() throws AnotherException{ throw new AnotherException(); }

}
```

1. It will compile but will throw AnotherException when run.
2. It will compile but will throw NewException when run.
3. It will compile and run without throwing any exceptions.
4. It will not compile.
5. None of the above.

Question 22.
Which of these are NOT legal declarations within a class?

1. static int sa ;
2. final Object[ ] objArr = { null } ;
3. abstract int t ;
4. abstract void format( ) ;
5. final static private double PI = 3.14159265358979323846 ;

Question 23.
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 { }
```

1. At Line 1.
2. At Line 2.
3. At Line 3.
4. At Line 4.
5. None of the above.

Question 24.
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);
    }
}
```

1. 2
2. 3
3. 6
4. 7
5. 9

Question 25.
Given a class named Test, which of these would be valid definitions for a constructor for the class?

1. Test(Test b) { }
2. Test Test( ) { }
3. private final Test( ) { }
4. void Test( ) { }
5. public static void Test(String args[ ]) { }

Question 26.
What can be inserted in the following code so that it will print exactly 2345 when compiled and run?

```
public class FlowTest {

    static int[] data = {1, 2, 3, 4, 5};

    public static void main(String[] args) {
        for (int i : data) {
            if (i < 2) {
                //insert code1 here
            }
            System.out.print(i);
            if (i == 3) {
                //insert code2 here
            }
        }
    }
}
```

1. **break;** and **//nothing is required**
2. **continue;** and **//nothing is required**
3. **continue;** and **continue;**
4. **break;** and **continue;**
5. **break;** and **break;**

Question 27.
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() );
  }
}
```

1. 43
2. 7
3. It will not compile.
4. It will throw an exception at runtime.
5. None of the above.

Question 28.
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);
   }
}
```

1. It will not compile because line //1 cannot be parsed correctly.
2. It will not compile because x is used before initialization.
3. It will not compile because there is an ambiguous reference to x.
4. It will print 12.
5. It will print 3.

Question 29.
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");
      }
   }
}
```

1. It will not compile.
2. It will print E1 and Finally.
3. It will print E1, E and Finally.
4. It will print E and Finally.
5. It will print Finally.

Question 30.
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);
}
}
```

1. 13432
2. 13423
3. 14324
4. 12434
5. 12342

Question 31.
The following code snippet will not compile:

```
int i = 10;
System.out.println( i<20 ? out1() : out2() );
```

Assume that out1 and out2 methods have the following signatures: public void out1(); and public void out2();

1. True
2. False

Question 32.
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);
     }
}
```

1. true, false, true
2. true, true, false
3. true, false, false
4. true, true, true

Question 33.
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 RuntimeException.
    }
}
```

Which of the following statements are true?
You had to select 2 options.

1. If IOException gets thrown at line1, then the whole method will end up throwing SQLException.
2. If IOException gets thrown at line1, then the whole method will end up throwing CloneNotSupportedException.
3. If IOException gets thrown at line1, then the whole method will end up throwing InstantiationException.
4. If no exception is thrown at line1, then the whole method will end up throwing CloneNotSupportedException.
5. If SQLException gets thrown at line1, then the whole method will end up throwing InstantiationException.

Question 34.
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`

1. It will print Problem in 0
2. It will throw an exception and end without printing anything.
3. It will not even compile.
4. It will not print anything if the argument is '1' instead of 'one'.
5. None of the above.

Question 35.
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 );
      }
   }
}
```

1. It will print 0, 1 and 2.
2. It will not print anything.
3. Compilation error.
4. Runtime error.
5. None of the above.

Question 36.
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?

1. In Number
2. In Object
3. In Long
4. In Integer
5. It will not compile.

Question 37.
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" ?

1. It will not compile.
2. It will return 10.0
3. It will return Float.Nan
4. It will return 0.0
5. None of the above.

Question 38.
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]);
  }
}
```

1. true true
2. true false
3. false true
4. false false
5. It will not compile.

Question 39.
Given:

```
abstract class Vehicle{ }

interface Drivable{ }

class Car extends Vehicle implements Drivable{ }

class SUV extends Car { }
```

Which of the following options will compile?
You had to select 2 options.

1. ArrayList<Vehicle> al1 = new ArrayList<>();
   SUV s = al1.get(0);
2. ArrayList<Drivable> al2 = new ArrayList<>();
   Car c1 = al2.get(0);
3. ArrayList<SUV> al3 = new ArrayList<>();
   Drivable d1 = al3.get(0);
4. ArrayList<SUV> al4 = new ArrayList<>();
   Car c2 = al4.get(0);
5. ArrayList<Vehicle> al5 = new ArrayList<>();
   Drivable d2 = al5.get(0);

Question 40.
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;
       }
   }
}
```

1. int
2. float
3. long
4. double
5. char
6. byte
7. short