FIRST TEST

EXPLANATION:

Option A is correct. We can import static members of the Arrays class for making this code works. In Option A we have followed the correct way of import statically.

Options B and E are incorrect since they are invalid syntaxes.

Options C and D are incorrect, as we need a static import because we haven't used full qualified names in the class.

QUESTION 1/76:

Given the following method from the java.util.Arrays

Method Summary: static void sort(long[] a)

And given,

```
1.  public class Sorter{
2.      public static void main(String []args){
3.          int[] a = {3,1,22,5,11,2};
4.          sort(a);
5.      }
6.  }
```

Which of the following will make this code works?

A. By adding the import statements "import static java.util.Arrays.sort;"

B. By adding the import statement "static import java.util.Arrays.*;"

C. By adding the import statement "import java.util.Arrays.*;"

D. By adding the import statement "import java.util.Arrays.sort;"

E. By adding the import statement "import static java.util.*;"

EXPLANATION:

When creating a java source code file, one should follow following order.

package statement - comes first and only one such a statement is allowed.

import statements - comes after the package statement and multiple import statements are allowed.

class statements - comes after the import statements.

We can use comments anywhere in the source code files they are ignored by the compiler in the compile time.

As explained above option A is correct.

QUESTION 2/76:

Which is of the following can appear before the package statements in a java file?

A. Comments.

B. Another package statement.

C. Class statements and/or comments

D. Lambda expression.

E. Any of above

EXPLANATION:

Option A is correct as only the statement I is incorrect.

Statement I is correct as the instance variables have wider scope than the local variables. The scope of a local variable is limited to the method in which it is declared and scope of an instance variable spread over the class.

Statement II is incorrect since the variables the instance variables are not declared inside methods. They are declared inside the class without using static modifier.

Statement III is incorrect as we can't say anything about the variable "i" as it could be a local variable or instance variable because it doesn't use any modifier. If It has used a modifier except "final" then it should be an instance variable.

QUESTION 3/76:

Which is/are true?

I. Scope of a local variable is lesser than scope of an instance variable.

II. The instance variables are only visible to the methods in which they are declared.

III. The variable "i" in the declaration "int i = 12" should be an instance variable.

A. Only I.

B. Only II.

C. Only I and II.

D. Only I and III.

E. None

EXPLANATION:

We can use "*" for importing all the classes of a package. So, the given imported statement will import only "Abc" and "Two" class. Packages appear to be hierarchical, but they are not. For example, the Java API includes a java.awt package, a java.awt.color package, a java.awt.font package, and many others that begin with java.awt. However, the java.awt.color package, the java.awt.font package, and other java.awt.xxxx packages are not included in the java.awt package. So, here, the classes of "bar" package are not imported. Therefore, option E is correct.

Other options are incorrect as explained above.

QUESTION 4/76:

Consider following directory structure. (Assume that current directory is same directory where "com" is located.

com - - -|

|- Abc.class

|- -foo - -|

|- One.class

|- Two.class

|- bar - -|

|- Test.class

Now consider the following import statement;

import com.*;

Which of the following set shows the set of the classes which will be imported when used above statement?

A. Abc, One, Two, Test

B. One, Two, Test
C. One, Two
D. Abc, One, Two
E. None of above.

EXPLANATION:

The signature of the main method must take one form of the following two forms;

public static void main(String[] args) or public static void main(String… args)

And it can also, be a final.

Starting point of a program is the main method; simply means that the program starts to execute statements which are located inside the main method. So, here "BC" will be printed. Therefore, option B is correct.

Options A and C are incorrect as they are not the main method. They are just overloaded versions of the main method and it is legal.

Option D is incorrect as there will be no error thrown as the code has the main method.

Option E is incorrect as the code compiles fine.

QUESTION 5/76:

Given

```
1.  public class Program{
2.      public static void main(int []i){
3.          System.out.print("AB");
4.      }
5.      public static void main(String... c){
6.          System.out.print("BC");
7.      }
8.      public static void main(String [][]c){
9.          System.out.print("CD");
10.     }
11. }
```

What is the result?

A. AB

B. BC

C. CD

D. An Error is thrown at the runtime, stating that, Main method not found in class Pro.

E. Compilation fails.

EXPLANATION:

Option C is correct since inside the call method only the instance variable "y" and the method argument "x" is changed. Here the static variable is shadowed by the argument variable "x". Shadowing occurs when two variables are located in two different scopes with same name. In such case the closet scope variable is chosen. For example here, the call method

changes the value of the argument “x” from 5 to 10 since the argument is in the local variable scope. So, the static variable remains unchanged. Therefore, the output will be 11 11 12.

Other options are incorrect as explained above.

QUESTION 6/76:

Given

```
1.  public class Program{
2.      static int x = 11;
3.      private int y = 33;
4.      public static void main(String args[]){
5.          Program pro = new Program();
6.          pro.method(5);
7.          System.out.print(Program.x);
8.          System.out.print(" " + pro.x);
9.          System.out.print(" " + pro.y);
10.     }
11.     public void method(int x){
12.         x = 10;
13.         y = 12;
14.     }
15. }
```

What is the result?

A. 10 10 33

B. 11 11 33

C. 11 11 12

D. 11 22 12

E. Compilation fails.

EXPLANATION:

Option D is correct as the code fails to compile. In the main method, we have declared the argument as “String[] a” and inside the main method we have tried to create a Integer type array with the name “a”. This will cause a compile time error since we can't declare two variables with same name inside a same scope. Here both method argument and the Integer array belong to the same local scope.

Other options are incorrect as the code fails to compile.

QUESTION 7/76:

Given

```
1.  public class Program{
2.      static int x = 2;
3.      public final static void main(String[] a){
4.          int []a = new int[x];
```

```
5.                          a[2] = 2;
6.                          for (int i:a)
7.                                  System.out.print(i);
8.                  }
9.          }
```

What is the result?

A. 0 2

B. 2

C. An exception is thrown.

D. Compilation fails.

E. None of above.

EXPLANATION:

Option E is correct as we can use int compatible literal for char. So, here assigning 65 to “c” is legal. Here the real value of the char c is equal to “A” because ASCII value of A is 65.

When using a float, we should add the suffix “f” or “F” to the end of the literal if the literal is a fraction number. If the literal is not a fraction then suffix is optional. So, option A is incorrect as we didn’t use suffix there.

Option B is incorrect as we can only use int compatible literal for char. Here trying to assign a String value to the variable “ch” is illegal.

Option C is incorrect as we can only use “false” or “true” for a boolean.

QUESTION 8/76:

Which of the following lines will compile without warning or error?

A. float f=1.3;

B. char ch="a";

C. boolean b=0;

D. int y = 2.3;

E. char c=65;

EXPLANATION:

When invoking methods, we should pass suitable arguments. So, code fails to compile as we have tried to invoke the “print()” method without passing any parameters but there is no such overloaded version of the print method of the PrintStream class. Therefore, option E is correct.

Options A, B, C and D are incorrect as the code fails to compile.

QUESTION 9/76:

Given

```
1.          public class Program{
2.                  static int x = 20;
3.                  public static void main(String args[]){
4.                          Program pr = new Program();
```

```
5.                          pr.x = 5;
6.                          int y = x/pr.x;
7.                          System.out.print("y =");
8.                          System.out.print();
9.                          System.out.print(y);
10.                 }
11.         }
```

What will be the result?

A. y = 1

B. y = 2

C. y = 3

D. Compilation fails due to an error on line 6.

E. Compilation fails due to an error on line 8.

EXPLANATION:

At line 2, we have defined static variable x with value 20, since it is static all instance of the class share same variable. So, if any instance change value of x, that change will reflect on all instances. In this case changing value of x at line 5, So, value of the variable x will be 5 for any instance. Hence, the output would be y=1. So, option C is correct.

QUESTION 10/76:

Given

```
1.          public class Program{
2.                  static int x = 20;
3.                  public static void main(String args[]){
4.                          Program pr = new Program();
5.                          pr.x = 5;
6.                          int y = x/pr.x;
7.                          System.out.print("y = ");
8.                          System.out.print(y);
9.                  }
10.         }
```

What will be the result?

A. y = 4

B. y = 2

C. y = 1

D. Compilation fails due to an error on line 6.

E. Compilation fails due to an error on line 8.

EXPLANATION:

The interface variables are implicitly static. So, we can access them using interface name and also, we can access them directly from static context. So, this code will compile fine and produce the output as "Value is: 11". Therefore, option A is correct.

Option B is incorrect as at line 3, trying to access variable x without interface name or reference variable won't cause any problem because variable "s" is inherited from interface Face.

Option C is incorrect as we can also, access interface variable using interface name.

Option D is incorrect since there are no compile time errors.

QUESTION 11/76:

Given

```
1.    class Program implements Face {
2.        public static void main(String args[]){
3.        System.out.print(s);
4.        System.out.print(Face.value);
5.        }
6.    }
7.    interface Face {
8.        static int value = 11;
9.        String s = "Value is: ";
10.   }
```

What will be the result?

A. Value is: 11

B. Compilation fails due to an error on line 3.

C. Compilation fails due to an error on line 4.

D. Compilation fails due to multiple errors.

E. An exception will be thrown at runtime.

EXPLANATION:

Option D is correct since all are primitive literals, they are double, float and char.

Option A is incorrect as "a" is a String literal.

Option B is incorrect as True is incorrect literal is should be true.

Option C is incorrect as 'BF' is illegal; char literal can only has one letter.

QUESTION 12/76:

Which of the following set contains only primitive literals?

A. 1, 'c', "a"

B. 1, 1.5f, True

C. 'BF', 10, "Sure"

D. 1.2D, 1f, 'c'

E. None of above.

EXPLANATION:

Option E is correct since no method in String class effect on the string object which they are invoked on. For example consider following code fragment;

String s = “abc”;

s.concat(“def”); // after executing this the value of the variable s, still remain as “abc”.

What happens is, concat method returns the string “abcdef” without changing the value of the variable s.

QUESTION 13/76:

Which methods from the String class modify the object on which they are called?

A. charAt(int index)

B. concat(String str)

C. toUpperCase()

D. split(String regex)

E. None of above.

EXPLANATION:

Option D is correct as among the given methods, only the charAt(int index) method can be found in both classes.

Options A, B and E are incorrect as these methods are defined in the String class. Append method is defined only in StringBuilder class.

QUESTION 14/76:

Which of the following method can be found in both StringBuilder and String classes?

A. concat(String str)

B. isEmpty()

C. append(String str)

D. charAt(int index)

E. toUpperCase()

EXPLANATION:

Option E is correct as code fails to compile due to an error on line 4. StringBuilder class doesn’t have a method called “concat()”. The “concat() method is a method belong to the String class. We should have used the “append()” method there.

If we change line 4 to “s.append(“SE 8”)”, then the option C would be correct because the length of the String “Java SE 8” is nine, 9 will be printed when line 5 executed. And we have used StringBuilder class’ “public StringBuilder(String str)” constructor, when we construct a string builder initialized to the contents of the specified string. The initial capacity of the string builder is 16 plus the length of the string argument. So, here we have initially passed “Java” to constructor and length of it; is 4. Therefore, the initial capacity is 20. So, 20 will be printed when line 6 executed.

QUESTION 15/76:

Given

1. class Program{

```
2.      public static void main(String [] args){
3.          StringBuilder s =new StringBuilder("Java");
4.          s.concat(" SE 8");
5.          System.out.print(s.length() + " ");
6.          System.out.print(s.capacity());
7.      }
8.  }
```

What is the result?

A. 9 9

B. 9 16

C. 9 20

D. 20 20

E. Compilation fails.

EXPLANATION:

Option A is correct as the variable “cd” is equal to “ef”, So, “s1+cd” returns “abcef”.

Option D is incorrect as we can use a String in that manner. At line 5, calling substring method on the “abcdef” will return a string with value “ef” because the “4” is the index position of e and we haven’t specify the last index So, every letter after the index position 4 will be return.

Option E is incorrect as we can create a string from invoking the string class constructor. There is a constructor which takes a char array So, line 6 is completely legal.

QUESTION 16/76:

Given

```
1.  public class Pro{
2.
3.      public static void main(String args[]){
4.          char c[] = new char[]{'a','b','c'};
5.          String cd = "abcdef".substring(4);
6.          String s1 = new String(c);
7.          s1 += cd;
8.          System.out.print(s1);
9.      }
10. }
```

What is the result?

A. abcef

B. abcdef

C. An exception will be thrown.

D. Compilation fails due to error at line 5.

E. Compilation fails due to error at line 6.

EXPLANATION:

Option E is correct as the code fails to compile. There are three types of variables class, instance and local. Local variables (variables which are declared inside a method) must be initialized before using them. But at line 4 the statement "int i, j = 2", only initialized the variable j. Therefore, at line 5, trying to use the variable "i" before initialize, causes a compile time error.

Other options are incorrect as the code fails to compile. However, if we could initialize the variable "i" then the code would compile and output will be the value of the local variable "i". This concept is called as variable shadowing.

QUESTION 17/76:

Given

```
1.  public class Program{
2.      static int i = 1;
3.      public final static void main(String[] args){
4.          int i, j = 2;
5.          System.out.print(i);
6.      }
7.  }
```

What is the result?

A. 1

B. 2

C. 0

D. An Exception.

E. Compilation fails.

EXPLANATION:

When class is loaded the static fields are initialized and also, the static blocks are executed. So, when the class Program is loaded, j is initialized to 1 but then the static field at line 11 changes the value of j to 2. So, the last value of the variable j is 2. Not like static variables, the instance variables initialized when objects are created. So, here the value of i is also, 2. Therefore, the output is 2 2. So, option C is correct.

Options A and B are incorrect as explained above.

Option D is incorrect as there is no issue at line 3, because the variable is j is initialized before i get initialized.

QUESTION 18/76:

Given

```
1.  public class Program{
2.      int i = j;
3.      static int j = 1;
4.      public static void main(String args[]){
5.          System.out.print(new Program().i + " ");
6.          System.out.print(j);
```

```
7.                  }
8.                  static{   j = 2;    }
9.          }
```

What is the result?

A. 1 2

B. 1 1

C. 2 2

D. Compilation fails due to error on line 2.

E. Compilation fails due to multiple errors.

EXPLANATION:

Option B is correct since an object is eligible for GC when there is no reference to an object in a currently live thread.

Like other any program, java applications can run out of memory. So, option A is incorrect.

Option C is incorrect as the purpose of the GC is to remove the objects which have no reference in a currently live thread.

Option D is incorrect as the JVM decide when to run GC whether we request or not.

QUESTION 19/76:

Which is true?

A. Java applications never run out of memory as Garbage Collector manages the memory.

B. An object is eligible for GC when there is no live thread can reach it.

C. The purpose of GC is to delete objects that there is no use at the moment.

D. When you request GC to run, it will start to run immediately.

EXPLANATION:

Option E is correct as calling “toString()” method on String builder objects returns strings accordingly. So, comparison is equal to following;

“OCAJP”.equals(“OCAJP”);

Option A is incorrect as it causes a compile time error because String and StringBuilder are incomparable types.

Option B is incorrect as “==” only check the both reference refer to same object, So, here sb and sb0 refer to two different objects Therefore, this will return false.

Options C and D are incorrect as both result false because the StringBuilder class doesn’t override the “equals()” method.

QUESTION 20/76:

Consider following three statements.

```
I.      StringBuilder sb =new StringBuilder("OCAJP");
II.     StringBuilder sb0 =new StringBuilder("OCAJP ");
III.    String s =new String ("OCAJP");
```

Which of the following will result true?

A. s == sb

B. sb == sb0

C. sb.equals(sb0)

D. s.equals(sb)

E. sb.toString().equals(s.toString())

EXPLANATION:

Option A is correct as the output is 1.

Option D is correct as it is legal to use underscore at the beginning or end of a number literal.

At line 4 we have created an int variable with the name "_5" and at line 5 we have assigned the value of the variable "_5" to x. So, "_5" is not a literal with underscore. If there is no variable declared as "_5" then line 5 would cause a compile time error.

QUESTION 21/76:

Given

```
1.    class Program{
2.        public static void main(String args[]){
3.            int _5 = 10;
4.            int x  = _5;
5.            System.out.print(_5/x);
6.        }
7.    }
```

Which is the output?

A. 1

B. 2

C. We can't use underscore at the beginning or end of a number literal.

D. Compilation fails due to error at line 3.

E. Compilation fails due to error at line 4.

EXPLANATION:

Option D is correct. This code fails to compile due to line 5, since when using binary literals we can use only '1' and '0's.

QUESTION 22/76:

Given

```
1.    class Program{
2.    public static void main(String args[]){
3.        int x = 011;
4.        int y  = 0b2;
5.        System.out.print(x+y);
6.        }
```

```
7.      }
```

Which is the output?

A. 13

B. 10

C. 1

D. Compilation fails.

EXPLANATION:

Option E is correct as the code fails to compile. When using a primitive or reference type variables as case constants, we should keep in mind that they should be constant. Basically they should mark as final. But here all chars are not final, therefore, code fails to compile.

Option D is incorrect as from java se 7, it is legal to strings as case constants.

Other options are incorrect as the code fails to compile.

QUESTION 23/76:

Given

```
1.      class Program{
2.      public static void main(String args[]){
3.          char s = ‘a’;
4.          char c1 = ‘A’;
5.          char c2 = ‘b’;
6.          char c3 = ‘B’;
7.          switch(s){
8.              case c1 : {System.out.print("A");};
9.              default  : {System.out.print("default ");};
10.             case c2 : {System.out.print("B");}; break;
11.             case c3 : {System.out.print("C");};
12.             }
13.         }
14.     }
```

Which is the output?

A. default

B. defaultB

C. AdefaultB

D. Compilation fails as Strings can’t be used as the case constant.

E. Compilation fails due to multiple errors.

EXPLANATION:

Here we have evaluate same expression “true |false ^ true” by applying parentheses in different positions.

If we consider the default precedence, the expression “false ^ true” will be evaluated before evaluating “true | false” which will produce true as the output.

Option D is correct as parentheses change the default precedence. Here “true | false” will evaluated before evaluating with “^” So, with this the output will be false since “true ^ true” results false.

Options A, B and C are incorrect as the default precedence has not changed there.

QUESTION 24/76:

Which of the following will result false?

A. true | (false ^ true);

B. (true | false ^ true);

C. true | false ^ true;

D. (true | false) ^ true;

E. None of above.

EXPLANATION:

Here we have combined two ternary operators. first one check local variable x is greater than static variable x if not then it do another comparison to check if local variable x is lesser than static variable x, otherwise it will print “<”,. If second condition met then it will print >, if not then <> will be printed, So, in this case both variables have same value So, <> will be printed. Hence, option C is correct.

QUESTION 25/76:

Given

```
1.    class OCAJP {
2.        static int x = 10;
3.        public static void main(String[] args){
4.            int x = 10;
5.            System.out.println(x>OCAJP.x?">":x<OCAJP.x?"<":"<>");
6.        }
7.    }
```

What is the result?

A. <

B. >

C. <>

D. An exception is thrown at the runtime.

E. Compilation fails.

EXPLANATION:

Option B is correct. Here we have defined double wrapper with value 12.85, invoking intValue on that double will retrun only the integer part of the number and it will simply drop the fraction part, Hence, output will be 12.

QUESTION 26/76:

Given

```
1.  public class OCAJP{
2.      public static void main(String[] args){
3.          Double d = new Double(12.85);
4.          int i = d.intValue();
5.          System.out.println(i);
6.      }
7.  }
```

What is the result?

A. 12.45

B. 12

C. 13

D. An exception is thrown at the runtime.

E. Compilation fails.

EXPLANATION:

Two wrappers Will always be == when their primitive values are the same:

- Boolean
- Byte
- Character from \u0000 to \u007f (7f is 127 in decimal)
- Short and Integer from -128 to 127

Note: When == is used to compare a primitive to a wrapper, the wrapper will be unwrapped and the comparison will be primitive to primitive.

So, in this case i3 == i4 will be true and other will be false becouse 160 in not in range -128 to 127. Hence, option D is correct.

QUESTION 27/76:

Given

```
1.  public class OCAJP{
2.      public static void main(String[] args){
3.          Integer i1 = 160;
4.          Integer i2 = 160;
5.          Integer i3 = 10;
6.          Integer i4 = 10;
7.          System.out.print((i3 == i4) + " " + (i1 == i2));
8.      }
9.  }
```

What is the result?

A. false true

B. true true

C. false false

D. true false

E. Compilation fails.

EXPLANATION:

Java predicate is a function interface and its' functional method is test(Object). It is method signature is

Boolean test(Object c);

So, here we use test method to check the condition, that the price of each object greater than 180 or not. For that we have to override the test method of predicate object like follows;

```
Predicate <Certification>  filter = new Predicate <Certification>  (){
        public boolean test(Certification c){
        return c.getPrice() > 180;
}
};
```

But since Predicate is java functional interface we can use lambda expression as given in option D.

Option A is incorrect as we haven't used the braces.According to the given code we can't use Consumer or Supplier functional interfaces for the task.

QUESTION 28/76:

Given

```
1.      public class Certification{
2.          private String name;
3.          private double price;
4.
5.          public Certification(String s,double d){
6.              name = s;
7.              price = d;
8.          }
9.
10.         public double getPrice(){
11.             return price;
12.         }
13.
14.         public String getName(){
15.             return name;
16.         }
17.     }
18.
19.     //Assume necessary imports have done
```

```
20.
21.         public class CertifyTest {
22.
23.             public static void main(String[] args) {
24.
25.                 List<Certification> certs = new ArrayList<Certification>();
26.
27.                 certs.add(new Certification("1Z0-803",120));
28.                 certs.add(new Certification("1Z0-804",250));
29.                 certs.add(new Certification("1Z0-805",175));
30.                 certs.add(new Certification("1Z0-808",150));
31.                 certs.add(new Certification("1Z0-810",225));
32.
33.                 //insert here
34.
35.                 for(Certification c : certs){
36.                     if(filter.test(c)){
37.                         System.out.println(c.getName());
38.                     }
39.                 }
40.             }
41.
42.         }
```

Suppose we have to print only the certifications where price are higher than 180, which of the following insert at line 33 will achieve that?

A. Predicate<Certification> filter = (c) -> return c.getPrice() > 180;

B. Consumer <Certification> filter = (c) -> {return c.getPrice() > 180;};

C. Supplier <Certification> filter = (c) -> {return c.getPrice() > 180;};

D. Predicate <Certification> filter = (c) -> {return c.getPrice() > 180;};

E. Consumer <Certification> filter = () -> {return c.getPrice() > 180;};

EXPLANATION:

At line 15, we can see we have invoked test method, which is functional method of Predicate interface. So, options B and C are incorrect. Since we have declared list as double type, we should use double version of predicate, Hence, option E produce expected result.

Option D is incorrect since we haven't given generic type there, but if we used it as follows then it would work

```
Predicate<Double> ip = (i) -> {return i > 21.4;};
```

Option A is incorrect since there we have used Integer version of predicate interface. So, we can't pass double value to test method.

QUESTION 29/76:

Given

```
1.      //Assume all necessary importing have done
2.
3.      public class EPractizeLab{
4.          public static void main(String[] args){
5.
6.              ArrayList<Double> list = new ArrayList<>();
7.              list.add(21.6);
8.              list.add(21.39);
9.              list.add(21.5d);
10.             list.add(21.41);
11.
12.             //insert here
13.
14.             for(Double d : list){
15.                 if(ip.test(d)){
16.                     System.out.println(d);
17.                 }
18.             }
19.         }
20.     }
```

Which insert at line 12, will provide following output?

21.6

21.5

21.41

A. IntPredicate ip = i -> i > 21.4;

B. UnaryOperator<Double> ip = (i) ->{return i > 21.4;};

C. Consumer<Double> ip = (i) ->{return i > 21.4;};

D. Predicate ip = (i) -> {return i > 21.4;};

E. DoublePredicate ip = i -> i > 21.4;

EXPLANATION:

Option C is correct. Statement I is correct as the compiler provides a default constructor when we fail to specify a constructor.

Statement III is correct as when we specify a constructor then there will be no default constructor created.

Statement II is incorrect as default constructor has only “super()” call.

QUESTION 30/76:

Which is/are correct?

I. When we failed to provide a constructor to a class, a default constructor is created.

II. The default constructor has only the statement “this()”.

III. If we created a constructor, then the default constructor will be vanished.

A. Only I.

B. Only II.

C. Only I and III

D. Only II and III.

E. All

EXPLANATION:

Option A is correct as correct because a void method is allowed to have a return statement as long as it doesn’t try to return a value. So, option B is incorrect.

Option C is incorrect as there is no return type or void in the method signature.

Option D is incorrect as we can’t return double value there as return type should be int.

Option E is incorrect as there is no return statement in the method, where it should return int.

QUESTION 31/76:

Which of the following method compiles?

A. public void methodA() { return;}

B. public void methodB() { return null;}

C. ublic static methodC() {}

D. public int methodD() { return 9.0;}

E. public int methodE() { }

EXPLANATION:

Option E is correct since there are two errors in this code. Fist error is at line 5. Because we have defined a constructor, the default constructor is not created So, now there is only one constructor which can take int, So, trying to invoke the constructor without any argument causes a compile time error. Second error is at line 15. When we define a constructor a super or this call come before any statement. Therefore, there are multiple errors in this code at line 5 and 13.

Options A and B are incorrect as code fails to compile.

QUESTION 32/76:

Given

```
1.  class Pro{
2.
3.      public static void main(String args[]){
4.          System.out.print(new A(5).y);
5.          System.out.print(new A().x);
6.      }
7.  }
8.
```

```
9.      class A{
10.         A(int i){
11.             y = i;
12.             System.out.print("A");
13.             super();
14.         }
15.          int y;
16.          int x = 10;
17.     }
18.
```

Which is the output?

A. A510

B. A5 followed by an exception.

C. Compilation fails due to an error on line 5.

D. Compilation fails due to an error on line 13.

E. Compilation fails due to multiple errors.

EXPLANATION:

Option A is correct as a constructor can have any access modifier.

Option B is incorrect as constructors should have same name as class name.

Option C is incorrect as constructors must not have any return type. Not even “void”.

Option D is incorrect as it is illegal to use final with Constructors.

Option E is correct as constructors have same name as class name.

QUESTION 33/76: Select Multiple Answers

Given

```
1.      class Person{
2.          //What would be correct constructor for this class
3.          }
```

What would be correct constructor for Person class?

A. private Person(int x){ }

B. Persons(){ }

C. void Person(String s){ }

D. final Person(){ System.out.print(“Person”); //new return line }

E. Person(){ System.out.print(“Person”); } // new return line super();

EXPLANATION:

Option D is correct as class Bird has only one constructor and it is marked as private, So, it is illegal to try to create a Bird object from outside class as from outside class we can't invoke private constructor, simply a Bird instance can be created only inside Bird class. So, line 18 causes a compile time error.

Options A, B and C are incorrect since code fails to compile.

Option D is incorrect as there is no error in line 3.

QUESTION 34/76:

Given

```
1.  class Animal{
2.      Animal(){
3.          super();
4.      }
5.  }
6.
7.  class Bird extends Animal{
8.      private Bird(String name){
9.          System.out.print(name);
10.     }
11. }
12.
13. class Test{
14.     public static void main(String args[]){
15.         new Bird("parrot");
16.     }
17. }
```

Which is the output?

A. Unknown

B. parrot

C. unknown parrot

D. Compilation fails due to an error on line 15.

E. Compilation fails due to an error on line 3.

EXPLANATION:

According to the method call on line 5, we could assume following things about the method we are trying to invoke;

- method name is "charToInt"
- method should be static since it doesn't use any object reference.
- method should return int, because if method returns nothing then a compile time error occurs.
- method should be able to take char value.
- method can have any access level since they are in the same class.

Option A is correct since it is the only method which satisfies all above requirements.

Options B, D and E are incorrect as those methods are not static.

Option C is incorrect since it doesn't return anything.

QUESTION 35/76:

Given

```
1.      class Program{
2.
3.          public static void main(String args[]){
4.              char c = 'A'; //ASCII value of 'A'is 65.
5.              System.out.print( charToInt(c) );
6.          }
7.
8.          //insert here
9.              int x = c;
10.             return x;
11.         }
12.
13.     }
```

Choose the correct method signature for compilation to succeed?

A. static int charToInt(char c){

B. public int charToInt(char c){

C. public static void charToInt(char c){

D. public char charToInt(char c){

E. public char charToInt(){

EXPLANATION:

Option B is correct. According to the given description, method should take double array as argument and return int which represent the max values' index position. Since we are asked to create it as non instance method, we have to make it static, So, the correct method signature is;

static int max (double []c)

QUESTION 36/76:

You are asked to create a method which should satisfy the following requirements.

Method should be in package access level. It should take a double array. It should return the position index of the max value from the double array. Name of the method should be "max". And it shouldn't be an instance Method.

Which is the correct method signature?

A. private static double[] max(int c)

B. static int max (double []c)

C. double[]max (int c)

D. public double max(double []c)

E. int max(double []c)

EXPLANATION:

Give method signature has following characteristic;

- It returns nothing since it has “void” as the return type.
- It takes String as argument.
- It has defaults access level.

Option A is correct since protected access level is less restrictive than default access level.

Option B is incorrect as even this method has “main” as its name, it is not main method because the main method signature is unique.

Option C is incorrect as method returns nothing.

Option D is incorrect since this method signature is completely legal.

QUESTION 37/76:

Consider following method signature,

void main(String s)

Which of the following is\are true about above method signature?

A. Access level of this method is more restrictive than protected access level.

B. This is the main method.

C. It returns a String.

D. It’s invalid method signature.

E. None of above.

EXPLANATION:

Option A is correct as instance methods can access instance variables and instance methods directly.

Option B is incorrect as instance methods can access class variables and class methods directly.

Option C is incorrect as class methods can’t access instance variables and instance methods directly.

Option D is incorrect as static content can never use this since static content is not belongs to any instance, it belongs to class.

QUESTION 38/76:

Which are true?

A. Instance methods can access instance variables and instance methods directly.

B. Instance methods can’t access class variables and class methods directly.

C. Class methods can access Instance variables and Instance methods directly.

D. Both class methods and instance methods can use the keyword “this”.

E. None of above.

EXPLANATION:

Static variable “j” at line 3 has default value “0” as we have not given any specific value. So, when the “method()” invokes first time using “p1” object reference, while loop executes and will print “A” twice. Since the variable “j” is static “p3” and “p3” objects sees variable “i” with value larger than 3, So, when using “p2” and “p3” for invoking the “method()”, while loop will not execute.

The output only contains two “A”s as explained above, So, option B is correct.

Option A is incorrect, it would be correct if the variable “j” was a instance variable.

QUESTION 39/76:

Given

```
1.    public class Program {
2.
3.        static int j;
4.
5.        public static void main(String[] args){
6.            Program p1 = new Program();
7.            Program p2 = new Program();
8.            Program p3 = new Program();
9.            p1.method();
10.           p2.method();
11.           p3.method();
12.       }
13.
14.       public void method(){
15.           while(++j<3){
16.           System.out.print("A");
17.           }
18.       }
19.   }
```

What is the output?

A. AAAAAA

B. AA

C. No output.

D. Compilation fails due to error on line 15.

E. Compilation fails due to multiple errors.

EXPLANATION:

Option A is correct as the default constructor only contains the super call and it is in package access level since the class is in package access level.

Options B and C are incorrect as default constructor doesn’t have this call.

Option D is incorrect as there is no user defined constructor in OCAJP class, there is only a method with name “OCAJP”.

QUESTION 40/76:

Consider the following class.

```
1.    class OCAJP{
2.        void OCAJP(){}
3.    }
```

Which of the following is above class' default constructor?

A. OCAJP(){ super();}

B. OCAJP(){this();}

C. public OCAJP(){this();}

D. No default constructor will be created since there is a user defined constructor.

EXPLANATION:

Option B is correct. Statement III is correct as static classes are just like any top-level class.

Statement I is incorrect as inner classes have access to private members of the enclosing class.

Statement II is incorrect as any access modifier can be used with inner class.

QUESTION 41/76:

Which is correct?

I. Inner classes don't have access to private members of the enclosing class.

II. Inner class can't be declared with the "private" access modifier.

III. A static nested class is just like any other top-level class.

A. Only I.

B. Only III.

C. Only I and III.

D. Only II and III.

E. All

EXPLANATION:

Option E is correct. Statements I is incorrect as only the access modifiers public and default can be used with top level classes.

Statements III is incorrect as no access modifier can be used inside a method.

Statements II is incorrect as default access level is more restrictive than the protected access level.

QUESTION 42/76:

Which of the following statement/s are correct?

I. The access modifier "private" can be used with every class.

II. Default access level is less restrictive than the protected access level.

III. The only access modifier which can be used inside a method is public.

A. Only II.

B. Only III.

C. Only I and II.

D. Only I and III.

E. None of above

EXPLANATION:

LocalDate is an interface, So, we can't use new keyword with them. To get current time we can call now method on LocalDate interface. So, option B is correct.

QUESTION 43/76:

Which of the following will print only the date without time?

A. System.out.println(new LocalTime());

B. System.out.println(LocalDate.now());

C. System.out.println(LocalDate.today());

D. System.out.println(new Date());

EXPLANATION:

Option C is correct as interfaces a 100% abstract So, they can be marked with the abstract keyword but it is optional.

Option A is incorrect as the Interface variables are implicitly public, static and final So, the modifier protected is not allowed here.

Option B is incorrect as the interface methods are implicitly abstract. An abstract method can't have a body.

Option D is incorrect as interface methods are implicitly public So, trying to create private method causes a compile time error.

QUESTION 44/76:

Which of following represents a correct interface?

A. interface I{ protected int x = 10; }

B. interface I{int x = 10; void print(){} }

C. abstract interface I{ int x = 10; void print();}

D. interface I{int x = 10; private void print();}

E. None of above.

EXPLANATION:

Option C is correct. Abstract methods are optional for an abstract class, it is enough a class to mark with abstract keyword to class to be abstract.

Option B is incorrect as the abstract methods should be marked using the abstract keyword.

Option D is incorrect as abstract class should be marked with the abstract keyword and also, abstract method should mark with abstract keyword.

Option A is incorrect as the abstract class can't be final.

QUESTION 45/76:

Which of following represents a correct abstract class?

A. final abstract Abst { public abstract void print(); }

B. abstract class Abst { public void print(); }

C. abstract class Abst { public void print() { } }

D. class Abst { public void print(); }

E. None of above.

EXPLANATION:

The following is a list of rules for creating an interface, many of which you should recognize as adoptions of the rules for defining abstract classes.

1. Interfaces cannot be instantiated directly.

2. An interface is not required to have any methods.

3. An interface cannot be marked as final.

4. All top-level interfaces are assumed to have public or default access, and they must include the abstract modifier in their definition. Therefore, marking an interface as private, protected, or final will trigger a compiler error, since this is incompatible with these assumptions.

5. All non default methods in an interface are assumed to have the modifiers abstract and public in their definition. Therefore, marking a method as private, protected, or final will trigger compiler errors as these are incompatible with the abstract and public keywords.

So, option C is correct.

QUESTION 46/76:

Which of the following statement/s are correct?

I. Interfaces cannot be instantiated directly.

II. An interface may not be marked as final.

III. An interface can be declared with any access modifier.

A. Only II.

B. Only III.

C. Only I and II.

D. Only I and III.

E. None of above

EXPLANATION:

From java SE 8, we are allowed to add static and default method which shouldn't be abstract, So, in given code at line 3, trying to add static method without body causes a compile time error.

Option C is correct.

QUESTION 47/76:

Choose the correct statement about the following code:

```
1.  public interface Exam {
2.      int amount = 10;
3.      public static void getPass();
4.      int marks();
```

5. }

Which of the following could be the output?

A. It compiles and runs without issue.

B. The code will not compile because of line 2.

C. The code will not compile because of line 3.

D. The code will not compile because of line 4.

E. The code will not compile due to multiple errors.

EXPLANATION:

Concrete classes are, by definition, not abstract, So, option A is incorrect.

A concrete class must implement all inherited abstract methods, but if the subclass is abstract too then implementing all abstract methods is optional, So, option B is incorrect.

Option C is incorrect; a super class may have already implemented an inherited interface, So, then concrete subclass would not need to implement the method.

Concrete classes can be both final and not final, So, option D is incorrect.SO, Option E is correct.

QUESTION 48/76:

Which of the following is true?

A. A concrete subclass can be declared as abstract.

B. An abstract subclass must implement all inherited abstract methods.

C. A concrete subclass must implement all methods defined in an inherited interface.

D. A concrete subclass cannot be marked as final.

E. None of above.

EXPLANATION:

Option B is correct. Statement I is incorrect as an abstract class can contain zero to more abstract methods.

Statement III is incorrect as an abstract class has constructors like other classes.

Statement II is correct if there is one or more abstract method in a class, enclosing class should be abstract.

QUESTION 49/76:

Which of the following statement/s are correct?

I. An abstract class should have at least one abstract method

II. If there is one or more abstract method then class should be abstract.

III. Abstract class doesn’t have constructor/s.

A. Only I.

B. Only II.

C. Only I and II.

D. Only I and III.

E. None of above

EXPLANATION:

Remember that Strings are immutable and literals are pooled. The JVM created only one literal in memory. x and y both point to the same location in memory; therefore, the following outputs true.

String x = "Hello World";

String y = "Hello World";

System.out.println(x == y);

But in given code, we don't have two of the same String literal. Although x and y happen to evaluate to the same string, one is computed at runtime. Since it isn't the same at compile-time, a new String object is created. So, false will be printed as the output. Option A is correct.

QUESTION 50/76:

Choose the correct statement about the following code:

```
1.  public class OCAJP{
2.      public static void main(String[] args){
3.          String x = "Hello World";
4.          String y = " Hello World".trim();
5.          System.out.println(x == y);
6.      }
7.  }
```

Which of the following could be the output?

A. false

B. true

C. The code will not compile because of line 4.

D. The code will not compile because of line 5.

EXPLANATION:

Option D is correct as Java compiler already knows some inconvertible types. Here we try to convert a String to int , it cause a compile error and not ClassCastException. Code fails only because of this error So, E is incorrect.

This code fails to compile So, it won't produce any output or exception So, A, C and D are incorrect.

QUESTION 51/76:

Given

```
1.  class Program{
2.      public static void main(String args[]){
3.       int x = (int)args[0];
4.       System.out.print(x);
5.       }
6.  }
```

What is true?

A. If we use command line invocation, java Program 10, the output will be 10.

B. If we use command line invocation, java Program abc, An ClassCastException will be thrown.

C. If we use command line invocation, java Ex, a ArrayIndexOutOfBoundsException will be thrown.

D. Compilation fails due to error on line 3.

E. Compilation fails due to multiple errors

EXPLANATION:

Option C is correct as code fails due to error on line 16, as "=>" is not a valid operator. It should be corrected to ">=". If it is corrected then the output would be 24.

Options A and B are incorrect as code fails to compile.

QUESTION 52/76:

Given

```
1.    class Program{
2.
3.      public static void main(String[] args){
4.
5.        int val1 = 1;
6.        int val2 = 2;
7.
8.        if(val1 == val2)
9.          System.out.print("1");
10.       if(val1!= val2)
11.         System.out.print("2");
12.       if(val1 > val2)
13.           System.out.print("3");
14.       if(val1 < val2)
15.           System.out.print(4);
16.       if(val1 => val2)
17.           System.out.print("5");
18.
19.     }
20.   }
```

Which is the output?

A. 245

B. 24

C. Compilation fails due to an error on line 16

D. Compilation fails due to multiple errors

E. None of above.

EXPLANATION:

Option E is correct as the code fails to compile due to error on line 10. We can't use "case" when defining the default case So, at line 10 "case default" will cause a compile time error.

Options A and B are incorrect as the code fails to compile.

Option C is incorrect as using a string as a switch expression is valid from java se 7.

Option D is incorrect as using string as a case constant is valid from java se 7.

QUESTION 53/76:

Given

```
1.      class Program{
2.
3.      public static void main(String args[]){
4.
5.          final String s = "JAVA";
6.
7.          switch(s){
8.              case "JAVA"    : {System.out.print("A");}
9.              case "java"    : {System.out.print("B");}
10.             case default   : System.out.print("default"); break;
11.             }
12.         }
13.     }
```

Which is the output could be?

A. A

B. ABdefault

C. Compilation fails due to error at line 7.

D. Compilation fails due to error at line 8.

E. Compilation fails due to error at line 10.

EXPLANATION:

Option A is incorrect since when creating an array we should use "[]". So, here it should be corrected as "new char[]{'c', 'd', 'e'};.

Option B is incorrect since we should not specify the size of the array when creating anonymous array.

Option C is incorrect as we have missed the keyword new there.

Option D is incorrect as we can create anonymous arrays in java.

So, option C is correct.

QUESTION 54/76:

Which of the following represent the correct way of creating an anonymous one dimensional array?

A. new char{'c', 'd', 'e' };

B. new int[2]{5,6,7};

C. int[]{10, 20, 30 };

D. We can't create anonymous arrays in java

E. None of above.

EXPLANATION:

If the size of the array is n then the last index of the array is n-1.

In this code we have created a integer array which can hold 2 integers. At line 6 we have tried to add third element to the array (we have skip initializing the first element of the array – index position 0). So, tying to add elements more than the 2 cause an ArrayIndexOutOfBoundsException. So, option D is correct.

Other options are incorrect as code throws an exception before producing any output.

QUESTION 55/76:

Given

```
1.    public class Program{
2.        static int i = 2;
3.        public static void main(String[] args) {
4.            int array[] = new int[i];
5.            array[1] = 7;
6.            array[2] = 8;
7.            System.out.print(array[1]);
8.        }
9.    }
```

Which is true?

A. The output will be 7.

B. The output will be 0.

C. The output will be 8.

D. An Exception will be thrown at the runtime.

E. Compilation fails as we can't assign int values as the elements of a char array.

EXPLANATION:

Option C is correct as arrays are objects, and that each array dimension is a separate type. So, for instance, l2d is of type "two dimensional int array", which is a different type than l1d. Line 7 attempts to assign a one-dimensional array into an int. So, compilation fails due to line 7 because they are incompatible types.

A and B are incorrect because lines 5 and 6 perform legal array manipulations

QUESTION 56/76:

Given

```
1.    public class Program {
2.        public static void main(String[] args) {
3.            long [][] l2d;
4.            long [] l1d = {1,2,3};
```

```
5.                          Object o = l1d;
6.                          l2d = new long[3][3];
7.                          l2d[0][0] = (long[])o;
8.                  }
9.          }
```

What is the result?

A. Compilation fails due to an error on line 5.

B. Compilation fails due to an error on line 6.

C. Compilation fails due to an error on line 7.

D. Compilation succeeds and the code runs without exception.

E. Compilation succeeds and an exception is thrown at runtime.

EXPLANATION:

Option E is correct as the code fails to compile due to error on line 8 and 9 because polymorphic assignments can't be applied to the generic type parameter.

Other options are incorrect as explained above.

QUESTION 57/76:

Given

```
1.          import java.util.*;
2.          class Animal { }
3.          class Dog extends Animal { }
4.
5.          public class Pro{
6.                  public static void main(String[] args) {
7.                          List<Dog> c2 = new ArrayList<Dog>();
8.                          List<Animal> c3 = new ArrayList<Dog>();
9.                          ArrayList<Object> c4 = new ArrayList<Animal>();
10.                 }
11.         }
```

What are true?

A. Compilation succeeds.

B. An exception is thrown at the runtime.

C. Compilation fails due to an error on line 7.

D. Compilation fails due to an error on line 9.

E. Compilation fails due to multiple errors.

EXPLANATION:

Option C is correct since it uses correct syntax for creating two dimensional array.

Option A is incorrect as when creating two dimensional arrays we should use two “[]”.

Options B is incorrect as there we create one dimensional array.

Options E is incorrect since we should not specify dimension in variable declaration.

Option D is incorrect as we should specify the first dimension of the two dimensional array when creating a two dimensional array.

QUESTION 58/76:

Which correctly create a two dimensional array of double?

A. double array[][] = new double[10,10];

B. double array [] = new double[10];

C. double array [][] = new double[10][];

D. double [] array [] = new double[][10];

E. double array[10][] = new double[][];

EXPLANATION:

Option B is correct as it creates equal three dimensional array as given in the code fragment.

Option A is incorrect since it is two dimensional array.

Options C is incorrect as it's an invalid syntaxes, we have tried to assign two dimensional array to three dimensional array declaration.

Option E is incorrect since the index positions of number 12, 10, 11 and 21 are mismatched with the given array.

QUESTION 59/76:

Consider following code fragment.

```
int arr[][][] = new int[1][2][3];
arr[0][0][0] = 8;
arr[0][0][2] = 12;
arr[0][0][1] = 10;
arr[0][1][0] = 40;
arr[0][1][2] = 11;
arr[0][1][1] = 21;
```

Which of the following is equal to above code fragment?

A. int array1[][]= {{8,10,12},{40,21,11}};

B. int array[][][] = {{{8,10,12},{40,21,11}}};

C. int array[][][] = {{8,10,12},{40,21,11}};

D. int array[][][] = {{8,12,10},{40,11,21}};

E. int array[][][] = {{{8,12,10},{40,11,21}}};

EXPLANATION:

Option A is correct as the output is “-1 0”. At line 8, -1 will be printed since “D” is not included in the list. At line 10, 0 will be printed because we have sorted the alist using Collections class sort method.

Options B and C are incorrect as explained above.

Option D is incorrect as trying to get index position of an element which is not included in the list doesn't cause an exception.

Option E is incorrect as the code compiles fine.

QUESTION 60/76:

Given

```
1.  import java.util.*;
2.  public class Pro {
3.    public static void main(String[] args) {
4.        List<String> list = new ArrayList<String>();
5.        list.add("B"); list.add("C"); list.add("A");
6.        System.out.print(" " + list.indexOf("D"));
7.            Collections.sort(list);
8.        System.out.println(" " + list.indexOf("A"));
9.        }
10. }
```

What is the result?

A. -1 0

B. -1 1

C. 1 0

D. -1 followed by a runtime exception

E. Compilation fails.

EXPLANATION:

Length of array "a" is 4, So, the value of the variable y is 4. Execution of while loop will print array element reverse as variable "y" has initial value 4 , So, printing of elements start with 0 (a[3]) and runs till the value of variable "i" equal to -1. When it reaches to -1, an ArrayIndexOutOfBoundsException is thrown. So, option D is correct.

Options A, B and C are incorrect as an exception is thrown at the runtime.

Option E is incorrect as the code fails to compile.

QUESTION 61/76:

Consider following code segment

```
1.  import java.util.*;
2.  public class Program{
3.      public static void main(String[] args) {
4.          int []a = {3,2,1,0};
5.          int y =  a.length;
6.          while(y>=0){
7.          System.out.print(a[--y]);
```

```
8.                                  }
9.                          }
10.                 }
```

What would be the output, if it is executed as a program?

A. 0123

B. 3210

C. Never ending loop after printing 3210

D. 0123 followed by an exception.

E. Compilation fails

EXPLANATION:

General format of the for loop is;

for(initialization; Boolean expression; update){ //Statements }

Option B is correct as there we use correct syntax. When creating for loops all three blocks are optional, simply means that we can skip initialization, boolean expression or/and update statements.

Option A is incorrect as defining multiple variables in for loop uses the incorrect syntax.

Option C is incorrect as we should use ";" for separating three statements.

Option D is incorrect we have use boolean expression and update statements wrong places.

QUESTION 62/76:

Which of the following will compile successfully?

A. for(int j = 0,int k=5; j < k; k--) ;

B. for(;;System.out.print("a")) ;

C. for();

D. for(int k = 10; k--; k>0) ;

E. None of above.

EXPLANATION:

Option A is correct as it produce the expected output.

Option B is incorrect since the two for loops work individually So, in second for loop trying to use the variable a causes compile time error.

Option C is incorrect as arr is two dimensional array So, "for(int a: arr)" is invalid syntax.

QUESTION 63/76:

Consider following;

int arr[][] = {{1, 3, 5},{7,8}};

Which of the following print all elements of the array "arr"?

A.

```
for(int []a : arr)
for(int i: a) System.out.print(i + " ");
```

B.

```
for(int []a : arr);
for(int i: a) System.out.print(i + " ");
```

C.

```
for(int a : arr)
System.out.print(a + " ");
```

D. None of above.

EXPLANATION:

Option A is correct as the code produces the output as “1.go 2.go 3.go 4.go 5.go”

Option B is incorrect as the “0.go” will not print because “i++” executes before print statement.

Option C is incorrect as skipping any part of for loop doesn’t case any compile time error, in fact we can skip all three part if necessary.

Option D is incorrect because there is no issue in that for loop, many mistakenly call the third expression of for loop as “increment expression”, but we can put any virtually arbitrary code statements that you want to happen with each iteration of the loop.

QUESTION 64/76:

Given

```
1.  public class Program{
2.      public static void main(String[] args) {
3.
4.          for(int j = 0,k=5; j < k; k--) ;
5.          for(int j = 0; j++ < 3;) ;
6.          for(int i = 0; i < 5; i++, System.out.print(i + ".go ")) ;
7.
8.      }
9.  }
```

What will be the result?

A. 1.go 2.go 3.go 4.go 5.go

B. 0.go 1.go 2.go 3.go 4.go 5.go

C. An exception is thrown at runtime.

D. Compilation fails due to an error on line 6.

E. Compilation fails due to multiple errors.

EXPLANATION:

Option B is correct. We have defined two for loops; loop labeled with L2 is inside the loop L1. We have declared the loop variable j in outer loop, as after the first iteration of the outer loop, the value of variable j is 5. So, inner loop iterates only one time, which is in the first iteration of the outer loop. So, output will be 50 51 52 53 54.

QUESTION 65/76:

Given

```
1.  class OCAJP {
2.
3.      public static void main(String[] args) {
4.          L1: for (int x = 5, y = 0; x > 0; x--) {
5.
6.              L2: for (; y < 5; y++) {
7.
8.                  System.out.print(x + "" + y + " ");
9.                  if (x == 0)
10.                     continue L1;
11.             }
12.         }
13.     }
14. }
```

What is the output?

A. 50 51 52 53 54 40 41 42 43 44 30 31 32 33 34 20 21 22 23 24 10 11 12 13 14

B. 50 51 52 53 54

C. 50 40 30 20 10

D. 51 41 31 21 11

E. None of above.

EXPLANATION:

Option A is correct as ClassNotFoundException is a checked exception and it is thrown when a required class is not found in the compile time.

Options B, C and D are incorrect as they are unchecked exception.

NullPointerException	:	Is thrown when Invalid use of a null reference.
NumberFormatException	:	Is thrown when Invalid conversion of a string to a numeric format.
ClassCastException	:	Is thrown when Invalid cast.

QUESTION 66/76:

Which of the following is checked exception?

A. ClassNotFoundException

B. NullPointerException

C. NumberFormatException

D. ClassCastException

E. None of above.

EXPLANATION:

Option D is correct as all Exceptions except Errors and RuntimeException are subjected to catch or specify requirement.

Options A and C are incorrect as both the Errors and RuntimeException are not subjected to catch or specify requirement.

Option B is incorrect as Errors are not subjected to catch or specify requirement.

QUESTION 67/76:

Which of the following kind is subjected to catch or specify Requirement?

A. Errors.

B. All Exceptions except RuntimeException

C. RuntimeException

D. All Exceptions except Errors and RuntimeException

E. None of above.

EXPLANATION:

Option C is correct as at line 6, an ArithmeticException is thrown but it couldn't be caught by the multiple exception catch box So, the exception is passed to the catch box at line 11, So, "UnKnown" will be printed.

Option A is incorrect as an ArithmeticException is thrown at the runtime.

Option B is incorrect as the ArithmeticException couldn't be caught by the multiple exception catch box.

Option D is incorrect as there is no uncaught exception

Option E is incorrect as code compiles fine.

QUESTION 68/76:

Given

```
1.  class Pro{
2.
3.      public static void main(String args[]){
4.          int i = 10, j = 0;
5.          try{
6.              int k = i/j;
7.          }
8.          catch(IllegalArgumentException | NullPointerException e){
9.              System.out.print("Known");
10.         }
11.         catch(Exception e){
12.             System.out.print("UnKnown ");
13.         }
14.     }
15. }
```

Which is the output?

A. No output

B. Known

C. UnKnown

D. An uncaught exception will be thrown.

E. Compilation fails.

EXPLANATION:

When method is declared to throw a checked exception, then the compiler check whether the calling method has handled or declared the unchecked exception or not, if not then a compile time error will be produced.

Option A is correct since the both IOException and ClassNotFoundException are checked exceptions So, they must be handled or declared in the main method. Using this declaration, we have declared the both checked exceptions to be thrown.

Option B is incorrect as the ClassNotFoundException is not handled or declared. and SecurityException is not a checked exception.

Option C is incorrect as the ClassNotFoundException is not handled or declared and also, "|" is not valid there.

Option D is incorrect as the IOException is not handled or declared.

QUESTION 69/76:

Given

```
1.  public class Program{
2.      //insert here
3.          new Program().doIt();
4.          new Program().didIt();
5.      }
6.
7.      static void doIt()throws java.io.IOException {
8.          throw new java.io.IOException();
9.      }
10.
11.     static void didIt()throws ClassNotFoundException{
12.         throw new SecurityException();
13.     }
14. }
```

Which insert individually at line 2, will compile?

A. public static void main(String[] args)throws java.io.IOException, ClassNotFoundException {

B. public static void main(String[] args)throws java.io.IOException, SecurityException {

C. public static void main(String[] args)throws java.io.IOException|SecurityException {

D. public static void main(String[] args)throws ClassNotFoundException {

E. None.

EXPLANATION:

Option C is correct as we passed invalid formatted string for an int when passing strings to main method. Number format exception is thrown when invalid conversion of a string to a numeric format.

Option D is incorrect as we have passed one argument So, array Index Out Of Bounds Exception wouldn't be thrown.

QUESTION 70/76:

Consider following code fragment.

```
class OCAJP {
    public static void main(String[] args) {
        int i = Integer.parseInt(args[0]);
    }
}
```

What is the result if we run above code using "java OCAJP 11d"?

A. ArithmeticException

B. NullPointerException

C. NumberFormatException

D. ArrayIndexOutOfBoundsException

E. IllegalArgumentException

EXPLANATION:

You can't have multiple finally clauses for one try clause. In this code first finally clause causes the end of the try clause. So, other catch clause is like a catch clause without a try clause, So, it is illegal. So, compilation fails. So, the answer is C.

A, B, C, and E are incorrect as code fails to compile So, we can't talk anything about runtime.

QUESTION 71/76:

Given

```
1.  class Test{
2.      public static void main(String args[]){
3.          try{
4.              new Test().meth();
5.          }catch(ArithmeticException e){
6.              System.out.print("Arithmetic");
7.          }finally{
8.              System.out.print("final 1");
9.          }catch(Exception e){
10.             System.out.print("Exception");
11.         }finally{
12.             System.out.print("final 2");
13.         }
14.     }
```

```
15.
16.                 public void meth()throws  ArithmeticException{
17.                         for(int x=0;x<5;x++){
18.                                 int y = (int)5/x;
19.                                 System.out.print(x);
20.                         }
21.                 }
22.         }
```

What is the output?

A. Arithmetic final 1

B. Exception final 2

C. Arithmetic final 2

D. Compilation fails.

E. Arithmetic

EXPLANATION:

Option B is incorrect since we cannot consider Animal as a Bird because Bird class has extended the Animal class.

Option C is incorrect as Bird class doesn't have a reference to an Animal object.

Option D is incorrect as Animal class doesn't have a reference to an Bird object. So, option A is correct.

QUESTION 72/76:

Consider following class code fragment.

```
class Animal{}
class Bird extends Animal {}
```

Which of the following is true?

A. Bird is an Animal.

B. Animal is a Bird.

C. Bird has an Animal.

D. Animal has a Bird.

E. None of above.

EXPLANATION:

Option C is correct as we can hide static methods of super class by defining a methods with the same signature as in super class.

Options A and B are incorrect as sub class can't only use the private methods and the private fields of the super class directly.

QUESTION 73/76:

When a class is extended by another class then the subclass can,

A. All methods of super class can be used directly as they are located in subclass.

B. All fields of super class can be used directly as they are located in subclass.

C. We can hide the static methods of super class by defining a methods in the subclass with the same signature as in super class.

D. All.

E. None.

EXPLANATION:

Option D is correct as with down casting, a ClassCastException is possible.

Option A is incorrect as the when we casting the object becomes more general So, it will remove object's unique actions. Therefore, object capabilities are reduced.

Option B is incorrect as up casting can be occurred implicitly but down casting should be done explicitly.

Option C is incorrect as the casting subclass object to super class object is known as up casting.

QUESTION 74/76:

Which of following statements is true?

A. The up casting increases the capabilities of the object.

B. The up casting can't be occurred implicitly.

C. Casting subclass object to super class object is known as down casting.

D. With down casting, a ClassCastException is possible.

E. None of above.

EXPLANATION:

When creating a time, you can choose how detailed you want to be. You can specify just he hour and minute, or you can add the number of seconds. For that we can use any of following methods;

public static LocalTime of(int hour, int minute)

public static LocalTime of(int hour, int minute, int second)

public static LocalTime of(int hour, int minute, int second, int nanos)

So, option D is correct.

QUESTION 75/76:

Which of the following create LocalTime object to represent time 6.15 AM?

A. LocalDate time = LocalDate.of(6, 15);

B. LocalTime time = new LocalTime(6, 15);

C. LocalTime time = LocalTime.getTime(6, 15);

D. LocalTime time = LocalTime.of(6, 15);

E. None of above

EXPLANATION:

We can use of method from LocalDate class to create LocalDate object for specific date, following you can see the method signature, and note that month is not zero based, for instance 1 means January.

public static LocalDate of(int year, int month, int dayOfMonth)

So, in this case 2015-01-20 is printed. Option A is correct.

QUESTION 76/76:

Given

```
1.  import java.time.LocalDate;
2.
3.  public class OCAJP{
4.      public static void main(String[] args){
5.          LocalDate date = LocalDate.of(2015, 1, 20);
6.          System.out.println(date);
7.      }
8.  }
```

Which of the following could be the output?

A. 2015-01-20

B. 2015-12-20

C. 2014-12-20

D. Compilation fails.

E. DateTimeException will be thrown.

SECOND TEST

EXPLANATION:

We can import all classes and interfaces using import statement with wild card. But there is no way to import only classes without other members, So, option E is correct.

QUESTION 1/77:

What is the correct import statement to import only all classes from java.util package?

A. import static java.util.*;

B. import java.util.*.class;

C. import java.util.*;

D. import java.util;

E. None of above

EXPLANATION:

When using static imports we are allowed to import all static members of a class. Syntax for importing all static members statically

```
import static [class name.*]
```

Only Option A follows the correct syntax for static import of Arrays class all static members.

Option B is incorrect as it is not static import.

Options C and D are invalid important statements.

QUESTION 2/77:

Which of the following import statement will import all static members of java.util.Arrays class?

A. import static java.util.Arrays.*;

B. import java.util.Arrays.*;

C. import static java.util.Arrays;

D. static import java.util.Arrays.*;

E. None of above

EXPLANATION:

We can have following comments in java

/* text */ - The compiler ignores everything from /* to */.

// text - The compiler ignores everything from // to the end of the line.

/** documentation */

This is a documentation comment and in general its called doc comment. The JDK javadoc tool uses doc comments when preparing automatically generated documentation. Option C is correct.

QUESTION 3/77:

Which of the following we can use for java doc comments?

A. /* text */

B. //text

C. /** text */

D. # text

E. None of above

EXPLANATION:

When creating a java source code file, one should follow following order.

package statement - comes first and only one such a statement is allowed.

import statements - comes after the package statement and multiple import statements are allowed.

class statements - comes after the import statements.

We can use comments anywhere in the source code files they are ignored by the compiler in the compile time.

As explained above option B is correct.

QUESTION 4/77:

Which is the correct order of following statements should appear?

I. class Statement.

II. Package statement.

III. import statement

A. III, II and I.

B. I, III and I.

C. I, II and III.

D. II, I and III.

E. III, I and II.

EXPLANATION:

The main method is the execution point of a java program. Main method signature is;

```
public static void main(String[] args)
```

In above code, we haven't defined a method with main method signature, at line 2, we have defined a method with name main but it is not the main method since it doesn't follow the main method signature. So, when we try to run the code we get following error, Hence, option D is correct.

Main method not found in class Program, please define the main method as:

```
public static void main(String[] args)
```

QUESTION 5/77:

Given

```
1.  public class Program{
2.      public static void main(){
3.          System.out.print("Program");
4.      }
5.  }
```

What is the output?

A. Program.

B. Nothing will be printed.

C. An Exception is thrown.

D. An Error is thrown.

E. Compilation fails.

EXPLANATION:

This is simple java program, and it has no errors. Program compile fine, and executes statements in the main method. So, Program will be printed, Hence, option A is correct.

QUESTION 6/77:

Given

```
1.  package test;
2.
3.  public class Program{
4.      public static void main(String[] args){
5.          System.out.print("Program");
6.      }
```

7. }

Which is true about above code?

A. Program will be printed.

B. Nothing will be printed.

C. An Exception is thrown.

D. We have imported test package.

E. Code doesn't has a main method.

EXPLANATION:

Option D is correct since the Java EE platform contains the specifications related to dynamic web content solutions, including servlets, JavaServer Pages, and JavaServer Faces.

Option A is incorrect because Java ME is the Java Micro Edition used for Mobile solutions.

Option B is incorrect because Java SE is the Java 2 Standard Edition used for basic application development. Other options are incorrect as explained above.

QUESTION 7/77:

Which of the following Java platform contains the specifications for servlets, JavaServer Pages, and JavaServer Faces?

A. Java ME

B. Java SE

C. Java EA

D. Java EE

E. Java BE

EXPLANATION:

The java.lang package is the only package that has all of its classes imported by default. So, option A is correct.

Other options are incorrect because we have to import those manually using import statements.

QUESTION 8/77:

Which of the following is the only Java package that is imported by default?

A. java.lang

B. java.awt

C. java.util

D. java.io

E. None of above.

EXPLANATION:

In a class that stores information about items, you would want to store the price of the item in a variable that will remain in scope for the life of the object. Hence, we instance variable is the most suitable. So, option E is correct.

QUESTION 9/77:

You need to create a class to store information about Items contained in a shop. What variable scope is best suited for the variable that will store the price of an item?

A. Method parameter

B. Static variable

C. Global variable

D. Local variable

E. Instance variable

EXPLANATION:

As calculations are performed on a variable, it should be stored as a local variable. In this scenario, the variable is only needed for this one method. So, option C is correct.

QUESTION 10/77:

You need to create a method that has two parameters to perform many complex steps and then return a result. What variable scope is best suited to store the value to be returned after the calculations are carried out upon it?

A. Static variable

B. Method parameter

C. Local variable

D. Instance variable

EXPLANATION:

At line 5, we have declared a variable and at line 10 we have tried to access that variable, but the scope of that variable limited to try block, So, compilation fails. Hence, option E is correct.

QUESTION 11/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          try{
5.              String name = null;
6.              System.out.println(name.length());
7.          }catch(NullPointerException e){
8.              System.out.println("Error");
9.          }
10.         System.out.print(name);
11.     }
12. }
```

What will be the result?

A. 0

B. null

C. Error.

D. An uncaught exception is thrown.

E. Compilation fails.

EXPLANATION:

Scope of the static variable include the main method too, but at line 6 we have declare a local variable with same name, which makes static variable is shadowed by the local variable. So, at line 7, what we change the value of that local variable, not the static (if it were static then code should fail to compile since static variable y is final). So, at line 8, 12 will be printed. Hence, option C is correct.

QUESTION 12/77:

Given

```
1.    public class Program{
2.
3.        final static int y = 10;
4.
5.        public static void main(String[] args){
6.            int y = 20;
7.            y = 12;
8.            System.out.print(y);
9.        }
10.   }
```

What will be the result?

A. 10

B. 20

C. 12

D. Compilation fails due to an error on line 6.

E. Compilation fails due to an error on line 7.

EXPLANATION:

Option D is correct since all are primitive literals, they are double, float and char.

Option A is incorrect as “a” is a String literal.

Option B is incorrect as True is incorrect literal is should be true.

Option C is incorrect as ‘BF’ is illegal; char literal can only has one letter.

QUESTION 13/77:

Which of the following set contains only primitive literals?

A. 1, ‘c’, “a”

B. 1, 1.5f, True

C. ‘BF’, 10, “Sure”

D. 1.2D, 1f, ‘c’

E. None of above.

EXPLANATION:

We can't invoke methods on primitive types, since they don't have any method. Only reference types can have methods. String is not a primitive, it is reference variable. String has many methods such as length(), So, option C is correct.

QUESTION 14/77:

On which of following variable type we can invoke methods?

A. double

B. float

C. String

D. char

E. Any of above.

EXPLANATION:

Local variables must be initialized before using them, at line 6 even if declared variable we haven't initialize it before using it at line 7, So, compilation fails. Hence, option E is correct.

QUESTION 15/77:

Given

```
1.  public class Program{
2.
3.      final static int x = 10;
4.
5.      public static void main(String[] args){
6.          int x;
7.          System.out.print(x);
8.      }
9.  }
```

What is the result?

A. 10

B. 0

C. null

D. An exception is thrown.

E. Compilation fails.

EXPLANATION:

When we initialize array, elements of array gets their default values. In above code at line 3, we have initialized a double array. Since we haven't assigned values for elements, they will take their default values, in this case default value is 0.0. So, option B is correct.

QUESTION 16/77:

Given

```
1.  public class Pro{
2.      public static void main(String args[]){
3.          double d[] = new double[5];
4.          System.out.print(d[1]);
5.      }
6.  }
```

What is the result?

A. 1

B. 0.0

C. null

D. An exception will be thrown.

E. Compilation fails due to error at line 4.

EXPLANATION:

The variable x is a short, at line 5 we have used increment operator. When using increment operator on a primitive type will not change. But at line 6, we have tried to add int to short, So, the result is in int format, trying to assign it to short causes compile time error.

So, option E is correct.

QUESTION 17/77:

Given

```
1.  public class Program{
2.
3.      public static void main(String args[]){
4.          short x = 2;
5.          x++;
6.          short y = 3 + x++;
7.          System.out.print(y);
8.      }
9.  }
```

What is the result?

A. 7

B. 6

C. 5

D. An Exception.

E. Compilation fails.

EXPLANATION:

Some time we have to use explicit casting, where compiler refuses to do implicit casting. The correct syntax for casting element is;

[type] element = (type)[variable]

Option A follows the correct syntax, Hence, option A is correct.

QUESTION 18/77:

Consider following statement,

int y = 20;

Which of the following can be used to assign value of y in to a short variable?

A. short s = (short)y;

B. short s = short(y);

C. short s = y;

D. short s = y(short);

EXPLANATION:

Option B is correct since an object is eligible for GC when there is no reference to an object in a currently live thread.

Like other any program, java applications can run out of memory. So, option A is incorrect.

Option C is incorrect as the purpose of the GC is to remove the objects which have no reference in a currently live thread.

Option D is incorrect as the JVM decide when to run GC whether we request or not.

QUESTION 19/77:

Which is true?

A. Java applications never run out of memory as Garbage Collector manages the memory.

B. An object is eligible for GC when there is no live thread can reach it.

C. The purpose of GC is to delete objects that there is no use at the moment.

D. When you request GC to run, it will start to run immediately.

E. None of above.

EXPLANATION:

To get expected output 30, we have to add the static variable y and the local variable y. We can refer to the local variable y, just using the variable name. To refer static variable we can either use, object reference p or class name. It is not good practice to use instance reference for referring static members though. So, we can refer static variable y, using Program.y.

As explained above we can get 30 by “y+Program.y” ,So, option C is correct.

QUESTION 20/77:

Given

```
1.  public class Program{
2.      static int y = 20;
3.      public static void main(String args[]){
4.          int y = 10;
5.          Program p = new Program();
```

```
6.                          // insert here
7.                  }
8.          }
```

Which of the following insert at line 6, will produce output as 30?

A. System.out.print(y+y);

B. System.out.print(p.y+Program.y);

C. System.out.print(y+Program.y);

D. System.out.print(this.y+Program.y);

E. System.out.print(p.y+super.y);

EXPLANATION:

Option A is correct because you can use the dollar sign in identifiers.

Options B and C is incorrect identifier because true and new are Java reserved words.

Option D is not valid because the dot (.) is not allowed in identifiers.

Option E is incorrect since we can't start with identifier with a number.

QUESTION 21/77:

Which of the following is valid Java identifiers?

A. $one

B. new

C. true

D. my.Age

E. 2s

EXPLANATION:

Integer class has parse methods, such as parseInt() , return a primitive, and the valueOf() method returns a wrapper class. In given code, we have used value of method and parseInt method at line 6, value of method will parse wrapper with value 3, and parseInt will parse primitive with value 3. At line 6, parsed value 3 pass both conditions, So, 1 to 4 will be printed, Hence, the option A is correct.

QUESTION 22/77:

Given

```
1.          public class Program {
2.                  public static void main(String[] args) {
3.                          int i = 4;
4.                          int j = 1;
5.
6.                          if(i > Integer.valueOf(args[0]) && j < Integer.parseInt(args[0])){
7.                                  System.out.println("1 to 4.");
8.                          }else{
```

```
9.                                  System.out.println("Not 1 to 4.");
10.                             }
11.                         }
12.                 }
```

And, if the code compiles, given the following two command-line invocations:

I. java Program 3

Which is the output?

A. 1 to 4.

B. Not 1 to 4.

C. An exception is thrown.

D. Compilation fails.

EXPLANATION:

Static variables take their default values when declaring, So, in given code the value i will be null. At line 7, trying to print the variable I (which will try to invoke toString() method of the Integer class) results a null pointer exception. Hence, option C is correct.

QUESTION 23/77:

Given

```
1.          public class Program {
2.
3.              static Integer i;
4.              public static void main(String[] args) {
5.                  int i = 4;
6.                  System.out.println(i + Program.i);
7.              }
8.          }
```

Which is the output?

A. 4.

B. 0.

C. An Exception is thrown.

D. Compilation fails due to error at line 3.

E. Compilation fails due to error at line 6.

EXPLANATION:

Option A is correct since we can use xxxValue() method to convert a Wrapper to a primitive. For example;

```
int pri = new Integer(100).intValue();
```

Option B is incorrect since primitive parseXxx(String) method to convert a String to a primitive.

Option C is incorrect wrapper valueOf(String) method to convert a String to a Wrapper.

Option D is incorrect since there is no method called value.

QUESTION 24/77:

Which of the following Wrapper method can be used to convert a Wrapper to a primitive?

A. xxxValue

B. parseXxx

C. valueOf

D. Value

E. None of above.

EXPLANATION:

Here we have combined two ternary operators. First one check local variable x is greater than 10 if not then it do another comparison to check if local variable x is lesser than 10, otherwise it will assign 1 to the variable y. If second condition met then it will assign -1 to the variable y, if not then 0 will be assigned to the variable y, So, in this case 0 will be assigned to the variable y. Hence, option D is correct.

QUESTION 25/77:

Given

```
1.    class OCAJP {
2.        public static void main(String[] args){
3.          int x = 10;
4.          int y = x>10?1:x<10?-1:0;
5.       System.out.println(y);
6.          }
7.    }
```

What is the result?

A. 1

B. -1

C. 10

D. 0

E. Compilation fails.

EXPLANATION:

When using if-else or if-else if, we don't need to use any condition check for else, as it is to execute when no above if or else if conditions met. So, due to line 10, code fails to compile. Hence, option E is correct.

QUESTION 26/77:

Given

```
1.    public class Program {
2.        public static void main(String[] args){
3.            int x = 10;
4.            int y = 12;
```

```
5.
6.                  if(x > y){
7.                          System.out.println("x > y");
8.                  }else if(x < y){
9.                          System.out.println("x < y");
10.                 }else(x == y){
11.                         System.out.println("x == y");
12.                 }
13.         }
14.     }
```

What is the result?

A. x > y

B. x < y

C. x == y

D. An exception is thrown at the runtime.

E. Compilation fails.

EXPLANATION:

One common mistake programmers make (and that can be difficult to spot), is assigning a boolean variable when you meant to test a boolean variable. Look out for code like the following:

```
boolean boo = false;
if (boo = true) { }
```

Above code compiles and runs fine and the if test succeeds because boo is SET to true (rather than TESTED for true) in the if argument.

In this code at line 6 instead using == we had used assignment operator =, which assign the false to the Boolean f, So, if statement will not be executed, when it reaches to else if, f stays false So, that statements won't execute too, So, option C is correct.

QUESTION 27/77:

Given

```
1.      public class Program {
2.
3.          static boolean f = true;
4.
5.          public static void main(String[] args){
6.              if(f = false){
7.                  System.out.println("false");
8.              }else if(f){
9.                  System.out.println("true");
10.             }
```

```
11.         }
12.     }
```

What is the result?

A. false

B. true

C. Nothing will be printed.

D. Compilation fails.

EXPLANATION:

It is illegal to have statements between if and else. At line 9, there is a printing statement, So, compiler can't understand where else clause belongs. So, compilation fails, Hence, option D is correct.

To avoid about error we could use curly braces as follows;

```
if(age > 18){
                legal = true;
                 System.out.println("Legal");
        }else
                System.out.println("Not Legal");
        }
```

QUESTION 28/77:

Given

```
1.      public class Program {
2.
3.          public static void main(String[] args){
4.              boolean legal = false;
5.              int age = 12;
6.
7.              if(age > 18)
8.                  legal = true;
9.                  System.out.println("Legal");
10.             else
11.                 System.out.println("Not Legal");
12.         }
13.     }
```

What is the result?

A. Legal

B. Not Legal

C. Nothing will be printed.

D. Compilation fails.

EXPLANATION:

At line 4, first evaluate the 2 * 5 and 3 * 4, which reduces the expression to the following:

int x = 10 + 12 - 8;

Then, evaluate the remaining terms in left-to-right order, resulting in a value of x of 14.

At line 5, we have values in the same order, but with set of parentheses:

```
int y = 2 * (5 + 3) * 4 - 8;
```

This time you would evaluate the addition operator 5 + 3, which reduces the expression to the following:

```
int y = 2 * 8 * 4 - 8;
```

So, the value of y would be 56.

Final value will be 42. Hence, option B is correct.

QUESTION 29/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          int x = 2 * 5 + 3 * 4 - 8;
5.          int y = 2 * (5 + 3) * 4 - 8;
6.
7.          System.out.println(y-x);
8.      }
9.  }
```

What is the result?

A. 0

B. 42

C. -24

D. Compilation fails.

EXPLANATION:

This example is tricky because of the second assignment operator embedded in line 5. The expression (t=false) assigns the value false to t and returns false for the entire expression. Since y does not equal 10, the left-hand side returns true; therefore, the exclusive or (^) of the entire expression assigned to b is true. The output reflects these assignments, with no change to y, So, option B is the only correct answer.

The code compiles and runs without issue, So, option E is not correct.

QUESTION 30/77:

Consider given code fragment.

```
1.  boolean b = true, t = true;
2.  int y = 20;
```

3. b = (y != 10) ^ (t=false);

4. System.out.println(b+", "+y+", "+t);

Which is the output?

A. true, 10, true

B. true, 20, false

C. false, 20, true

D. false, 20, false

E. The code will not compile.

EXPLANATION:

Statement I and II are incorrect, since == checks if the reference points to same object in case of a reference type, otherwise it will check primitive values. But equals is different it can be used only reference type, and it check meaningfully equality.

Statement III is incorrect as we can't use equals method on primitives.

So, option D is correct, since all three statements are incorrect.

QUESTION 31/77:

Which is/are correct?

I. Both == and equals are same thing.

II. We can use == with primitives only.

III. We can use equals method for both primitive and reference types.

A. Only I.

B. Only II.

C. Only I and III

D. None

E. All

EXPLANATION:

Strings are immutable and literals are pooled. The JVM created only one literal in memory. x and y both point to the same location in memory; therefore, the statement outputs true. The code compiles and runs without issue, So, at line 3, true will be printed.

String class equal method returns true when both strings have same values and it is case sensitive. So, it will also, return true. Hence, option A is correct.

QUESTION 32/77:

Consider given code fragment.

1. String x = "OCAJP";

2. String y = "OCAJP";

3. System.out.print((x == y) + " ");

4. System.out.print(x.equals(y));

Which is the output?

A. true true

B. true false

C. false true

D. false false

E. The code will not compile.

EXPLANATION:

In a switch, the case constant must be a compile time constant! Since the case argument has to be resolved at compile time, that means you can use only a constant or final variable that is assigned a literal value. It is not enough to be final; it must be a compile time constant. For example:

```
final int a = 1;
final int b;
b = 2;
int x = 0;
switch (x) {
 case a: // ok
 case b: // compiler error
}
```

So, at line 11 using the non final variable cs result a compile time error.

So, option E is correct.

QUESTION 33/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          String in = "OCAJP";
5.          String cs = "ocajp";
6.
7.          switch(in){
8.              default : System.out.println("default");
9.              case "OCAJP" : System.out.println("OCAJP");break;
10.             case "OCPJP" : System.out.println("OCPJP");break;
11.             case cs : System.out.println("ocajp");break;
12.         }
13.     }
14. }
```

What is the output?

A. default

B. OCAJP

C. OCPJP

D. ocajp

E. Compilation fails.

EXPLANATION:

In above code, we haven't used break for first set of odd numbers and at the final odd number break will stop moving to other cases, So, input is odd number only odd will be printed, and if it is even number then Even will be printed. So, here input is 7 which is an odd number.

Hence, option A is correct.

QUESTION 34/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          int num = 7;
5.
6.          switch(num){
7.              case 1 :
8.              case 3 :
9.              case 5 :
10.             case 7 :
11.             case 9 : System.out.print("Odd");break;
12.             case 2 :
13.             case 4 :
14.             case 6 :
15.             case 8 : System.out.print("Even");break;
16.         }
17.     }
18. }
```

Which is the output?

A. Odd

B. Even

C. OddEven

D. Nothing will be printed.

E. Compilation fails.

EXPLANATION:

In given statement we have defined a array with size 3. So, it can store up to three integers. Hence, option B is incorrect.

Array indexes are zero based, So, last element index is 2, Hence, option A is incorrect, Option D is correct since index position refers to actual second element of the array.

Option C is incorrect as array is not primitive.

QUESTION 35/77:

Given

int[] numbers1 = new int[3];

Which of the following is true about above statement?

A. Index of last element of above array is 3.

B. We can store four elements in this array.

C. The numbers1 variable is primitive.

D. We can access second element of above array using numbers1[1].

E. None of above.

EXPLANATION:

We have set of methods in java.util.Arrays that simplify many things when working with arrays. Such a method is sort method, which sort passed array in order. We can also, provide our own comparison method by passing comparator to the sort method otherwise it will sort in generic order. In above code array will be sorted at line 7, So, third element is 7, Hence, option C is correct.

QUESTION 36/77:

Given

```
1.      import java.util.Arrays;
2.
3.      public class Program{
4.          public static void main(String[] args){
5.
6.              int[] ints = { 7, 9, 4,2 };
7.              Arrays.sort(ints);
8.              System.out.println(ints[2]);
9.          }
10.     }
```

What is the output?

A. 2

B. 4

C. 7

D. An exception is thrown.

E. Compilation fails due to error at line 7.

EXPLANATION:

Option B is correct, as it follows correct syntax for defining anonymous array.

When defining anonymous array we don't need to provide size of the array, So, option A is incorrect.

Option C is incorrect because there we missed [].

Option D is incorrect as we need to use curly braces instead parentheses.

QUESTION 37/77:

Which of the following will create anonymous array of doubles?

A. new double[3]{1.2,3.2,3.1};

B. new double[]{1.2,3.2,3.1};

C. new double{1.2,3.2,3.1};

D. new double[](1.2,3.2,3.1);

E. None of above.

EXPLANATION:

Option A is correct as "[[]]" at left hand side is invalid.

When initializing a two dimensional array we need to provide first dimension size, So, option D is correct and options B and C are incorrect.

Option E is incorrect as there we defined variable two as one dimensional array, and trying to assign two dimensional array which makes it invalid.

QUESTION 38/77:

Which of the following is correct way to create a two dimensional array?

A. String[[]] two = new String[2][3];

B. String[][] two = new String[][];

C. String[][] two = new String[][3];

D. String[][] two = new String[2][];

E. String[] two = new String[2][3];

EXPLANATION:

Here we have two dimensional array, since indexes of array are zero based, ints[2][1] means the second element of third array and ints[1][1] means the second element of second array. So, the output is 2 + 2, which makes 4. So, option B is correct.

QUESTION 39/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.          int[][] ints = { {1,3}, {2,2},{4,2}};
4.          System.out.println(ints[2][1] + ints[1][1]);
5.      }
6.  }
```

What is the output?

A. 3

B. 4

C. 5

D. ArrayIndexOutOfBoundsException

E. Compilation fails.

EXPLANATION:

Here we need create three dimensional arrays in one statement to replace given array. From given array we can see that ints hold array of two, 2 dimensional arrays, and each of 2 dimensional arrays contain three, 1 dimensional arrays. In each of these 1 dimensional arrays, size is 1, So, they can only hold one value.

So, for there will be two , 2 dimensional arrays with three one dimensional arrays inside as follows

{{1},{2},{3}}

{{4},{5},{6}}

Now we need to put those in to an array which is three dimensional, like follows which matches option A.

{{{1},{2},{3}},{{4},{5},{6}}};

So, option A is correct.

QUESTION 40/77:

Consider;

```
1.      int[][][] ints = new int[2][3][1];
2.
3.      ints[0][0][0] = 1;
4.      ints[0][1][0] = 2;
5.      ints[0][2][0] = 3;
6.      ints[1][0][0] = 4;
7.      ints[1][1][0] = 5;
8.      ints[1][2][0] = 6;
```

Which of the following can replace above?

A. int[][][] ints = {{{1},{2},{3}},{{4},{5},{6}}};

B. int[][][] ints = {{1},{2},{3},{4},{5},{6}};

C. int[][][] ints = {{{1},{2},{3},{4},{5},{6}}};

D. int[][][] ints = {1,2,3,4,5,6};

E. int[][][] ints = {{1,2,3},{4,5,6}};

EXPLANATION:

Only while, do while, for, switch structures allow break, but if doesn't allow for break statement. So, option C is correct.

QUESTION 41/77:

Which of the following flow control structure/s allows break statement?

I. do while

II. if

III. for

A. Only I.

B. Only II.

C. Only I and III.

D. Only II and III.

E. All

EXPLANATION:

Only while, do while and for structures allow continue, but if and switch don't allow for continue statement. So, option B is correct.

QUESTION 42/77:

Which of the following flow control structure/s allows continue statement?

I. if

II. switch

III. while

A. Only II.

B. Only III.

C. Only I and II.

D. Only II and III.

E. None of above

EXPLANATION:

Option A is correct.

At line 4, while loop keep decrease the value of the variable y until the max int 2147483646.

QUESTION 43/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.          int y = 0;
4.          while(y-- < 10) { continue;}
5.          String message = y > 10 ? "Greater than" : "Less than";
6.          System.out.println(message+"  10");
7.      }
8.  }
```

What is the output?

A. Greater than 10

B. Less than 10

C. Infinite loop

D. Compilation fails due to error at line 4.

E. Compilation fails due to error at line 5.

EXPLANATION:

When using continue or break, we need to make sure that there are no statements to executes after them, if there are then complier has no way to reach them. At line 4, we have continue statement and after that we have another statement, which never going to executes because of the continue, in compile time compiler notice this and produce a compile time error, So, option D is correct.

QUESTION 44/77:

Given

```
1.    public class Program{
2.        public static void main(String[] args){
3.            int y = 0;
4.            while(y-- < 10) { continue; y +=2;}
5.            String message = y > 10 ? "Greater than" : "Less than";
6.            System.out.println(message+"  10");
7.        }
8.    }
```

What is the output?

A. Greater than 10

B. Less than 10

C. Infinite loop

D. Compilation fails due to error at line 4.

E. Compilation fails due to error at line 5.

EXPLANATION:

The general syntax for the for loop is

```
    for(initialization; Boolean_expression; update)
{
    //Statements
}
```

So, here for update part we have used both increment and printing part which is completely legal So, option B is correct.

Option A is incorrect since it will print from 0 to 3.

Option C is incorrect as it will result never ending loop since there is no incrementing on variable x.

Option D is incorrect as for each iteration incrementing will be done twice So, it will skip some values between 0 to 5.

QUESTION 45/77:

Which of the following will print numbers from 0 to 5(excluding)?

A. for(int x = 0;x<4;System.out.print(x++));

B. for(int x = 0;x<5;System.out.print(x++));

C. for(int x = 0;x<4;System.out.print(x));

D. for(int x = 0;x++<4;System.out.print(x++));

E. None of above

EXPLANATION:

Option A is correct. Statement I is correct since when using do while loop, statements in do block executes at least one time.

Statement II is incorrect as there we haven't use semicolon at the end, which is a common mistake.

Statement III is incorrect since statements in do block executes at least one time, even the while condition not met.

QUESTION 46/77:

Which of the following statement/s are correct?

I. The do while loop executes at least one time even the loop condition not met.

II. do{ }while(Boolean condition) represents correct syntax structure for a do while loop.

III. Statements in do block only executes if while condition met.

A. Only I.

B. Only II.

C. Only III.

D. Only I and III.

E. None of above

EXPLANATION:

The syntax for the "for" loop is as follows:

for(initialization; Boolean_expression; update){

//Statements

}

None of these 3 elements not a must for a for loop. If we omit all three, then infinite loop will be created, So, in this case code will keep print 10, since there is no condition to meet. Hence, option B is correct.

QUESTION 47/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.          for(;;)
4.          System.out.println("10");
5.      }
6.  }
```

What is the output?

A. 10.

B. 10 will be printed infinitely.

C. Nothing will be printed.

D. An exception is thrown.

E. Compilation fails due to error at line 3.

EXPLANATION:

The syntax for the “for” loop is as follows:

```
for(initialization; Boolean_expression; update){
//Statements
}
```

When you are aware about that a task needs to be repeated several times, you can use the “for” loop. It follows a structure where an integer is initialized, a condition is set with that integer, and then the integer is updated to a new value. Now, after initialization, the integer is checked for the Boolean expression. If found true, the code statements are executed. After this, the loop goes to the update statement and then again checks if the integer value satisfies the Boolean expression. If yes, the steps for loop execution are repeated until the Boolean expression becomes false.

It is important to note that, update statement can be any statement. So, in above code we have put the printing statment there, and inside the code block we increment the variable x, So, first value that will be printed is 1, and when x reach 10, that will be printed as final value, then loops ends. Hence, option A is correct.

QUESTION 48/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.          for(int x = 0;x < 10;System.out.print(x))
4.              x++;
5.      }
6.  }
```

What is the output?

A. 12345678910

B. 0123456789

C. 123456789.

D. Compilation fails.

EXPLANATION:

Syntax for the enhanced-for loop

```
for(declaration : expression){
    //Statements
}
```

Here, the declaration should be of the type of variable that you plan on accessing in the arrays. This variable will be used throughout the "for" block of code, and will be replaced each time with the array value that is being processed. The expression is nothing but evaluation of the array that we need to loop through. This expression generally is either an array variable or a method to return an array.

Option E is correct, as it has the correct syntax for accessing all elements in order, So, it can print values as expected.

QUESTION 49/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          int[] numbers = {1,2,3,4,5};
5.          // insert here
6.              System.out.print(i);
7.      }
8.  }
```

Which insert at line 5, will provide following output?

12345

A. for(int i : numbers);

B. for(int i ; numbers)

C. for(i : numbers)

D. for(numbers : int i)

E. for(int i : numbers)

EXPLANATION:

It is a common mistake that missing the semicolon when using do while loop. At line 8, we have missed the semicolon So, compilation fails.

Hence, option E is correct.

QUESTION 50/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          int i = 10;
5.              do{
6.                  i++;
7.                  System.out.println(i);
8.              }while(i-- > 10)
9.          }
10. }
```

What is the output?

A. 10

B. 11

C. 11 will be printed infinitely.

D. An exception is thrown.

E. Compilation fails.

EXPLANATION:

This code expects input from java console, So, if we couldn't provide argument, then at line 3, it will throw ArrayIndexOutOfBoundsException. Even if we pass value if it is not convertible to a number then it will result a NumberFormatException, So, options A and C are correct.

QUESTION 51/77: Select Multiple Answers

Given

```
1.    class Program{
2.        public static void main(String args[]){
3.            int x = Integer.parseInt(args[0]);
4.             System.out.print(x);
5.        }
6.    }
```

Which of the following exceptions possible in above code? (Choose 2)

A. NumberFormatException

B. NullPointerException

C. ArrayIndexOutOfBoundsException

D. ClassCastException

E. IllegalArgumentException

EXPLANATION:

From all given exceptions only the FileNotFoundException is a checked exception, it is a subclass of IOException.

Hence, option B is correct.

All other given exceptions are sub classes of RuntimeException, So, they are unchecked exception.

QUESTION 52/77:

Which of the following is a checked exception?

A. NoClassDefFoundError

B. FileNotFoundException

C. ExceptionInInitializerError

D. ClassCastException

E. IllegalArgumentException

EXPLANATION:

It is illegal to add any other code between the try, catch, or finally blocks. So, at line 7 adding print statement result compilation error. Hence, option E is correct.

QUESTION 53/77:

Given

```
1.  public class Program{
2.      public static void main(String args[]){
3.          double x =  0;
4.          try{
5.              x = Double.parseDouble("6d");
6.          }
7.          System.out.print(x);
8.          catch(NumberFormatException nfe){
9.              System.out.print("Number Format Exception");
10.         }
11.     }
12. }
```

Which is the output?

A. 6

B. 6d

C. 6.0

D. Number Format Exception

E. Compilation fails.

EXPLANATION:

At line 2, we have declared wrapper, since we haven't given any value to the variable x, it will take its' default value which is null. So, at line 5 trying to invoke the doubleValue method on null object result a NullPointerException. But finally block executes every time, no matter there is exception or not. So, "Done" will be printed and then NullPointerException is thrown. Hence, option D is correct.

QUESTION 54/77:

Given

```
1.  public class Program{
2.      static Double x;
3.      public static void main(String args[]){
4.          try{
5.              System.out.println(x.doubleValue());
6.          }finally{
```

```
7.                                  System.out.print("Done");
8.                          }
9.                  }
10.         }
```

What is the output?

A. 0.0 Done

B. Done

C. null Done

D. Done followed by a NullPointerException

E. Compilation fails.

EXPLANATION:

It is not necessary to have catch clause for a try block every time, we can have both catch and finally or any one of them, for any try block. So, option A is incorrect.

Option B is incorrect since it is perfectly legal to have nested try/catch.

Option C is correct , we can't add any other code between the try, catch, or finally blocks.

QUESTION 55/77:

Which of the following is true?

A. We must have a catch clause for a try block.

B. We can't nest few try/catch blocks.

C. It is illegal to add any code in between the try, catch, or finally blocks

D. None of above.

EXPLANATION:

If method is declared to throw a checked exception then calling method should catch or specify that exception.

Option A declared to throw a parseException which is a checked exception, but at main method, we haven't catch or throw the parseException So, option A is correct, Hence, it will not compile.

Other options are incorrect as all of other throwing exceptions are unchecked exception.

QUESTION 56/77:

Given

```
1.          public class Program{
2.                  static Double x;
3.                  public static void main(String args[]){
4.                          new Program().test();
5.                   }
6.
7.                  //insert here
8.          }
```

Which insert at line 7, will result compile time error?

A. public void test()throws ParseException{ /* Codes */}

B. public void test()throws RuntimeException{ /* Codes */}

C. public void test()throws NullPointerException{ /* Codes */}

D. public void test()throws StackOverflowError{ /* Codes */}

E. public void test()throws ClassCastException { /* Codes */}

EXPLANATION:

Option D is correct since static inner classes can never access instance members of outer class. So, trying to access instance variable "x" at line 11, causes a compile time error.

Option A is incorrect since code fails to compile.

Option B is incorrect as static inner class elements can be access with class name from outer class.

Option C is incorrect as at line 10 there is no issue in creating static classes.

QUESTION 57/77:

Given

```
1.    class Program{
2.
3.     int x = 10;
4.     static int y = 15;
5.
6.          public static void main(String args[]){
7.               System.out.print(Program.In.j);
8.          }
9.
10.         private static class In{
11.              static int j = y + x;
12.         }
13.   }
```

Which is the output?

A. 25

B. Compilation fails due to an error on line 7.

C. Compilation fails due to an error on line 10.

D. Compilation fails due to an error on line 11.

E. Compilation fails due to multiple errors.

EXPLANATION:

Statement I is incorrect as this code is completely disagree with encapsulation principals.

Statement I and III are incorrect as both methods and variables in this class doesn't implement the encapsulation principals. Here all variables are declared as public which allows anyone to modify variables without using methods and also, methods have declared to be private which should have declared as public.

Option E is correct since all statements are incorrect.

QUESTION 58/77:

Given

```
1.    class Program{
2.        public int i;
3.        public char c;
4.        public static void main(String [] args){
5.        //codes
6.        }
7.        private int getI(){
8.            return i;
9.        }
10.   }
```

Which, of the following can be considered as true about above code?

I. This code has correctly implemented the encapsulation principals.

II. This "getI" method at line7, has correctly implemented the encapsulation principals but not the variables.

III. Variables in this code have correctly implemented the encapsulation principals but not the "getI" method.

A. I only

B. II only

C. I and II only

D. I and III only

E. None

EXPLANATION:

Option D is correct since when passing primitives to a method, only the value is passed So, ant changes applied inside the method which takes the values as the argument, won't affect on its value. So, in this code the value of the variable x remain 5 after calling the "change" method. Therefore, 5 5 will be get printed.

Other options are incorrect as explained above.

Option E is incorrect as the code compiles fine.

QUESTION 59/77:

Given

```
1.    public class Program{
2.
3.        public static void main(String[] args) {
```

```
4.
5.                                  int x = 5;
6.                                  System.out.print(x + " ");
7.                                  apply (x);
8.                                  System.out.print(x);
9.                          }
10.
11.                 public static void apply(int x) {
12.
13.                                 if(++x > 10)
14.                                         x= 10;
15.                                 else
16.                                         x++;
17.                         }
18.         }
```

Which is the output?

A. 5 10

B. 5 11

C. 5 12

D. 5 5

E. Compilation fails.

EXPLANATION:

Not like primitives, when an object is passed to a method, actual object is never passed instead the reference to the object is passed. So, any changes on passed object in the method which takes object as argument will affect the original object.

Option B is correct as explained above the “updateAge” method changes the value of the variable “age” of object p. So, the output will be 11 12.

Options A and C are incorrect as explained above.

Option D is incorrect as the code compiles fine.

QUESTION 60/77:

Given

```
1.          public class Program{
2.
3.                  public static void main(String[] args) {
4.                          Person p = new Person("Will" , 11);
5.                          System.out.print(p.age +" ");
6.                          updateAge(p, 12);
```

```
7.                                  System.out.print(p.age);
8.                          }
9.
10.                         public static void updateAge(Person ps, int a) {
11.                                 ps.age = a;
12.                         }
13.         }
14.
15.         class Person{
16.                         Person(String s, int i){
17.                                 name = s;
18.                                 age = i;
19.                         }
20.                 String name;
21.                 int age;
22.         }
```

Which is the output?

A. 11 11

B. 11 12

C. 12 12

D. Compilation fails.

EXPLANATION:

Option E is correct as trying to access instance variable “j” from static method “calc” causes a compile time error. So, line 10 will cause a compile time error.

Options A, B and C are incorrect since the code fails to compile.

Option D is incorrect as line 6 is completely legal.

QUESTION 61/77:

Given

```
1.          public class Program{
2.                  final int j = 32;
3.
4.                  public static void main(String args[]){
5.                          char c = 'A'; //ASCII value of 'A' is 65 and 'a' is 97
6.                          System.out.print((char)calc(c));
7.                  }
8.
9.                  static int calc(int i){
```

```
10.                         return (i+j);
11.                 }
12.         }
```

What is the output?

A. 97

B. a

C. A

D. Compilation fails due to error on line 6.

E. Compilation fails due to error on line 10.

EXPLANATION:

Option A is correct as instance methods can access instance variables and instance methods directly.

Option B is incorrect as instance methods can access class variables and class methods directly.

Option C is incorrect as class methods can't access instance variables and instance methods directly.

Option D is incorrect as static content can never use this since static content is not belongs to any instance, it belongs to class.

Option E is incorrect as static variables belong to class So, all instance created using that class will share same static variable.

QUESTION 62/77:

Which is true?

A. Instance methods can access instance variables and instance methods directly.

B. Instance methods can't access class variables and class methods directly.

C. Class methods can access Instance variables and Instance methods directly.

D. Both class methods and instance methods can use the keyword "this".

E. Using the "static" keyword, we can create variables that are common to all objects of any class.

EXPLANATION:

According to the method call on line 5, we could assume following things about the method we are trying to invoke;

- method name is "val'
- method should be static since it doesn't use any object reference.
- method should return something, because if method returns nothing then a compile time error occur.
- method should be able to char value.
- method can have any access level since they are in the same class.

Option A is correct since it is the only method which satisfies all above requirements.

Options B, D and E are incorrect as those methods are not static.

Option C is incorrect since it doesn't return anything.

QUESTION 63/77:

Given

```
1.      class Program{
2.
3.          public static void main(String args[]){
4.              char c = 'A';
5.              System.out.print( val(c) );
6.          }
7.          //insert here
8.              int x = c;
9.              return x;
10.         }
11.     }
```

Choose the correct method signature for compilation to succeed?

A. private static int val(char c){

B. public int val(char c){

C. public static void val(char c){

D. public char val(char c){

E. public char val(){

EXPLANATION:

When we overload a method, we must change the argument list, also, not like overridden methods, we can return any different type in overloaded version. So, option C is correct.

Options A and B are incorrect since we haven't change argument list.

Option D is invalid method.

QUESTION 64/77:

Given

```
public void overLoadMe(){
        System.out.println("Over load me");
}
```

Choose the correct method signature for compilation to succeed?

A. public String overLoadMe(){ return "Over load me"; }

B. public void overLoadMe(){ System.out.println("Overloaded"); }

C. double overLoadMe(double x){ return x/2; }

D. public double overLoadMe(){ System.out.println("Overloaded"); }

E. None of above

EXPLANATION:

Option A is correct as the compiler provides a default constructor when we fail to specify a constructor.

Option C is correct as when we specify a constructor then there will be no default constructor created.

Option B is incorrect as default constructor has only “super()” call.

Option D is incorrect as any constructor, no matter user defined or default, all must have same name as class name.

Option E is incorrect as it is illegal to have two constructors with same argument list.

QUESTION 65/77: Select Multiple Answers

Which is/are correct? (Choose 2)

A. When we failed to provide a constructor to a class, a default constructor is created.

B. The default constructor has only the statement “this()”.

C. If we created a constructor, then the default constructor will be vanished.

D. The default constructors’ name is same as the class’s’ name but user defined constructors may have different names.

E. We can write two constructors that have the same number and type of arguments for the same class.

EXPLANATION:

Option E is correct since there are two errors in this code. Fist error is at line 5. Because we have defined a constructor, the default constructor is not created So, now there is only one constructor which can take int, So, trying to invoke the constructor without any argument causes a compile time error. Second error is at line 13. When we define a constructor a super or this call come before any statement. Therefore, there are multiple errors in this code at line 5 and 13.

Options A and B are incorrect as code fails to compile.

Options C and D are incorrect since code has multiple errors as explained above.

QUESTION 66/77:

Given

```
1.  class Program{
2.      public static void main(String args[]){
3.          System.out.print(new Pro(5).y);
4.          System.out.print(new Pro().x);
5.      }
6.  }
7.  class Pro{
8.      Pro(int i){
9.          y = i;
10.         System.out.print("Pro");
11.         super();
12.     }
13.     int y;
14.     int x = 10;
15. }
```

Which is the output?

A. Pro510

B. Pro5 followed by an exception.

C. Compilation fails due to an error on line 5.

D. Compilation fails due to an error on line 13.

E. Compilation fails due to multiple errors.

EXPLANATION:

You cannot use default methods to override any of the non-final methods in the java.lang.Object class. For example, you cannot override the equals method, it will cause a compile time error. So, in above code, trying to override equals method at line 5, causes a compile time error. So, option C is correct.

QUESTION 67/77:

Given

```
1.  interface A{
2.
3.      int groupID = 10;
4.
5.      default boolean equals(Object obj){
6.          return this.groupID == ((A)obj).groupID;
7.      }
8.
9.      static void print(){
10.         System.out.println("A");
11.     }
12. }
```

Which is true about above code?

A. Above code compiles fine.

B. Code fails to compile since interface can't has any non abstract method.

C. Compilation fails due to error at line 5.

D. Compilation fails due to error at line 6.

EXPLANATION:

Option D is correct. The interface variable price is correctly declared, with public and static being assumed and automatically inserted by the compiler, So, option B is incorrect. The method declaration for pass() on line 3 is correct because the method has been declared without body but with public and abstract being assumed and automatically inserted by the compiler. The method declaration for price() on line 4 is incorrect, since an interface method that provides a body must be marked as default or static explicitly. Therefore, option FD is the correct answer since this code contains only one error at line 5.

QUESTION 68/77:

Given

```
1.  public interface Exam {
2.      int price = 10;
3.      public void pass();
```

```
4.
5.          public int price() {
6.                  return 13;
7.          }
8.      }
```

Which is true about above code?

A. It compiles and runs without issue.

B. The code will not compile due to error at line 2.

C. The code will not compile due to error at line 3.

D. The code will not compile due to error at line 5.

E. The code will not compile due to multiple errors.

EXPLANATION:

Option A is correct since Interface variables are assumed to be public static final; therefore, other options are correct.

QUESTION 69/77:

What modifiers are assumed for all interface variables?

A. public,static and final

B. protected, final

C. public,static

D. final,static

E. public,abstract,static

EXPLANATION:

The compiler performs the following checks when you override a non private method:

1. The method in the child class must have the same signature as the method in the parent class.

2. The method in the child class must be at least as accessible or more accessible than the method in the parent class.

3. The method in the child class may not throw a checked exception that is new or broader than the class of any exception thrown in the parent class method.

4. If the method returns a value, it must be the same or a subclass of the method in the parent class, known as covariant return types.

So, as explained in second rule, we have use either protected or public. Hence, option C is correct.

QUESTION 70/77:

Given

```
1.      class Animal{
2.          protected void makeSound() {
3.                  System.out.println("sound");
4.          }
5.      }
```

```
6.
7.      class Dog extends Animal {
8.          _________ void makeSound() {
9.              System.out.println("Dog Barking");
10.     }
```

Which of the following can be used to fill the blank So, code compiles fine?

A. private

B. final

C. public

D. abstract

E. We can leave it blank

EXPLANATION:

Option D is correct. The compiler performs the following checks when you override a non private method:

1. The method in the child class must have the same signature as the method in the parent class.

2. The method in the child class must be at least as accessible or more accessible than the method in the parent class.

3. The method in the child class may not throw a checked exception that is new or broader than the class of any exception thrown in the parent class method.

4. If the method returns a value, it must be the same or a subclass of the method in the parent class, known as covariant return types.

QUESTION 71/77:

Which of the following is true?

A. When method overriding we can change argument list.

B. We can change throw any new exception in overriding method.

C. We can return anything in overriding method.

D. We must use same or less restrictive access modifier for overriding method.

E. None of above.

EXPLANATION:

Java predicate is a function interface and its' functional method is test(Object). It is method signature is

```
Boolean test(Object c);
```

Here we have created two predicates, one checks employer name start with A and other checks employer salary greater than 10000. at line 18 we have combined both predicates using default and method of predicate interface. And method returns a composed predicate that represents a short-circuiting logical AND of this predicate and another. So, objects passed only if both predicate match.

In given list only the Object with name Alice satisfies both conditions, So, Option B is correct.

QUESTION 72/77:

Given

```
1.      import java.util.ArrayList;
```

```
2.  import java.util.List;
3.  import java.util.function.*;
4.
5.  public class Labs {
6.
7.      public static void main(String[] args) {
8.
9.          List<Employer> list = new ArrayList<Employer>();
10.         list.add(new Employer("John", 12000));
11.         list.add(new Employer("Alice", 11000));
12.         list.add(new Employer("Dean", 16700));
13.
14.         Predicate<Employer> nFilter = (e) -> e.name.startsWith("Al");
15.         Predicate<Employer> sFilter = (e) -> e.salary > 10000;
16.         System.out.print("[");
17.         for(Employer e : list){
18.             if(nFilter.and(sFilter).test(e)){
19.                 System.out.println(e + " ");
20.             }
21.         }
22.         System.out.print("]");
23.     }
24. }
25.
26. class Employer{
27.     String name;
28.     double salary;
29.
30.     Employer(String s,double sal){
31.         name = s;
32.         salary = sal;
33.     }
34.     public String toString(){
35.         return name + " - " + salary;
36.     }
37. }
```

What is the output?

A. [John - 12000.0 Alice - 11000.0 Dean - 16700.0]

B. [Alice - 11000.0]

C. An exception is thrown.

D. Compilation fails due to error at line 14.

E. Compilation fails due to error at line 18.

EXPLANATION:

Java SE 8 added a new class for joining strings – StringJoiner which allows the ability to construct a sequence of characters separated by a delimiter and optionally starting with a supplied prefix and ending with a supplied suffix. Following example shows how it can be used,

The String "[George:Sally:Fred]" may be constructed as follows:

```
StringJoiner sj = new StringJoiner(":", "[", "]");
sj.add("George").add("Sally").add("Fred");
String desiredString = sj.toString();
```

In this code, we have created a list and then while iterating trough that list we add each string to the joiner. So, as shown in above example, result will be option C.

QUESTION 73/77:

Given

```
1.  import java.util.ArrayList;
2.  import java.util.List;
3.  import java.util.StringJoiner;
4.
5.  public class Program{
6.      public static void main(String[] args){
7.
8.          List<String> strings = new ArrayList<>();
9.
10.         strings.add("A"); strings.add("B"); strings.add("C");
11.         strings.add("D"); strings.add("E"); strings.add("F");
12.
13.         StringJoiner joiner = new StringJoiner(":", "[", "]");
14.         for (String str : strings) {
15.             joiner.add(str);
16.         }
17.
18.         System.out.print(joiner);
19.     }
20. }
```

What is the output?

A. [A]:[B]:[C]:[D]:[E]:[F]

B. [A:][B:][C:][D:][E:][F:]

C. [A:B:C:D:E:F]

D. An exception is thrown.

E. Compilation fails due to error at line 13.

EXPLANATION:

Java 5 introduced generics, which allow you to specify the type of class that the ArrayList will contain.

ArrayList<String> list = new ArrayList<String>();

Java 5 allows you to tell the compiler what the type would be by specifying it between <and >.

Starting in Java 7, you can even omit that type from the right side. The < and > are still required, though. This is called the diamond operator because <> looks like a diamond.

ArrayList<String> list = new ArrayList<>();

As explained above option D is correct.

QUESTION 74/77:

Which of the following creates an arrayList that can hold only Strings?

A. List list = new ArrayList(String);

B. List<> list = new ArrayList<String>();

C. List<String> list = new ArrayList ()<String>;

D. List<String> list = new ArrayList<String>();

E. None of above

EXPLANATION:

Timestamp class have method called from it obtains an instance of Timestamp from an Instant object. It has following method signature

public static Timestamp from(Instant instant)

We can get instant of current time, using Instant interface now method, So, option B is correct because we pass instant object to get timestamp.

Option A is incorrect, Timestamp class doesn't have such a method called now.

Option C is incorrect as Timestamp class doesn't have empty constructor.

Option C is incorrect, Timestamp class doesn't have such a method called valueOf.

QUESTION 75/77:

Which of the following will create Timestamp successfully?

A. Timestamp stamp = Timestamp.now();

B. Timestamp stamp = Timestamp.from(Instant.now());

C. Timestamp stamp = new Timestamp();

D. Timestamp stamp = Timestamp.valueOf(Intstant.now());

E. None of above.

EXPLANATION:

LocalDate class has parse method that can use to pass valid date string. In this example we pass string for 30th of December of 2014. At line 6, we have invoke plusDays by passing 2, So, it will add 2 days to current date, which makes the current date to 2015-01-1. But at line 7, we try to invoke method call plusHours but there is no such a method defined in LocalDate class. So, option E is correct.

QUESTION 76/77:

Given

```
1.  import java.time.LocalDate;
2.
3.  public class Program{
4.      public static void main(String[] args) {
5.          LocalDate date = LocalDate.parse("2014-12-30");
6.          date = date.plusDays(2);
7.          date.plusHours(24);
8.          System.out.println(date.getYear() + " " + date.getMonth() + " " + date.getDayOfMonth());
9.      }
10. }
```

What is the output?

A. 2015 JANUARY 1

B. 2015 JANUARY 2

C. 2014 JANUARY 2

D. An exception is thrown.

E. Compilation fails.

EXPLANATION:

The StringBuilder class has trimToSize() method which attempts to reduce storage used for the character sequence. If the buffer is larger than necessary to hold its current sequence of characters, then it may be resized to become more space efficient. Calling this method may, but is not required to, affect the value returned by a subsequent call to the capacity() method. This method takes no argument, So, at line 6, trying to invoke trimToSize method by passing int causes a compile time error. Hence, option E is correct.

QUESTION 77/77:

Given

```
1.  public class Program{
2.      public static void main(String args[]){
3.          StringBuilder sb = new StringBuilder();
4.          sb.append("OCPAJP");
5.          sb.deleteCharAt(4);
6.          sb.trimToSize(3);
```

```
7.                          System.out.println(sb);
8.                  }
9.          }
```

Which of the following could be the output?

A. OCPAJP

B. OCAJP

C. OCA

D. An Exception will be thrown.

E. Compilation fails.

THIRD TEST

EXPLANATION:

When using non static imports we need to provide class name or wild card to import all classes. Hence, option A is incorrect and options C and E are incorrect as it use invalid while card.

Option D is correct as it is valid import statement to import List.

QUESTION 1/77:

Which of the following is correct import statement?

A. import java;

B. import java.util;

C. import java.util.*.*;

D. import java.util.List;

E. import java.*.*;

EXPLANATION:

In given code we have used the static max method of Math class directly, So, we have to import statically that method or all static members of the class. Correct syntax to import max method statically is;

```
import static java.lang.Math.max;
```

So, option A is correct.

QUESTION 2/77:

Given

```
1.          // insert here
2.
3.          public class Program{
4.                  public static void main(String args[]){
5.                          System.out.println(max(2,3));
6.                  }
7.          }
```

Which of the following import statement we have to insert at line 1?

A. import static java.lang.Math.max;

B. import java.lang.Math.*;

C. import static java.Math.*;

D. import java.Math.max;

E. No import statement needs.

EXPLANATION:

All classes in lang package imported default So, import statements in line 1 and 2 are not necessary. We don't need to import List to work with ArrayList So, import statement in line 4 is not necessary. So, there are 3 statements we can remove, Hence, option C is correct.

QUESTION 3/77:

Given

```
1.      import java.lang.Math;
2.      import java.lang.*;
3.      import java.util.ArrayList;
4.      import java.util.List;
5.
6.      public class Program{
7.          public static void main(String args[]){
8.              ArrayList<Integer> list = new ArrayList<Integer>();
9.              list.add(2);list.add(3);
10.             System.out.println(Math.max(list.get(0),list.get(1)));
11.         }
12.     }
```

How many unnecessary import statement/s in above code?

A. 1

B. 2

C. 3

D. 4

E. None

EXPLANATION:

The scopes of the local variable x at line 3 and 4 limited to enclosing blocks. And line 6 will change value of x at line 5, So, 12 will be printed.

Hence, option B is correct.

QUESTION 4/77:

Given

```
1.      public class Program{
```

```
2.      public static void main(String args[]){
3.          {int x = 15;}
4.          {int x = 20;}
5.          int x  = 10;
6.          {x = 12;}
7.          System.out.println(x);
8.      }
9.  }
```

What is the output?

A. 10

B. 12

C. 20

D. Compilation fails due to line 6.

E. Compilation fails due to multiple errors.

EXPLANATION:

Code fails to compile because we have declared two variables with same name, “x” at line 2 and line 3, since they are in same scope, compilation fails and option D is correct.

QUESTION 5/77:

Given

```
1.  public class Program{
2.      public static void main(String x[]){
3.          int x = 10;
4.          x++;
5.          System.out.println(x++);
6.      }
7.  }
```

What is the output?

A. 10

B. 11

C. 12

D. Compilation fails due to line 3.

E. Compilation fails due to multiple errors.

EXPLANATION:

In program class we have instance variable and in the method at line 10, we have declared the argument variable as x too, in this case argument variable shadow the instance variable. So, value of argument will be printed instated instance one. Option B is correct.

QUESTION 6/77:

Given

```
1.  public class Program{
2.
3.      int x = 10;
4.
5.      public static void main(String args[]){
6.          Program pr = new Program();
7.          pr.method(3);
8.      }
9.
10.     public void method(int x){
11.         System.out.println(x);
12.     }
13. }
```

What is the output?

A. 10

B. 3

C. Compilation fails due to line 10.

D. Compilation fails due to multiple errors.

EXPLANATION:

Inside a class method, when a local variable has the same name as one of the instance variables, the local variable shadows the instance variable inside the method block. So, the value of "x" (10) in the method print() can be seen as the output. Therefore, the answer is B.

QUESTION 7/77:

Given

```
1.  class Program{
2.      int x = 11;
3.      public static void main(String[] args) {
4.          new Program().print();
5.      }
6.      public void print(){
7.          int x  = 10;
8.          System.out.print(x + " ");
9.      }
10. }
```

Which is the output?

A. 11

B. 10

C. 21

D. Compilation fails.

EXPLANATION:

Both variables “a” and “b” are local variables, So, they can’t be accessed from other methods. Trying to use variable b in method “print()” will Therefore, cause the compilation to fail. Answer E is correct because we tried to use variable b in line 12, which will cause the compilation to fail.

QUESTION 8/77:

Given

```
1.    class Program{
2.        public static void main(String[] args) {
3.            new Program().doit();
4.        }
5.
6.        public void doit(){
7.            int a =3 , b=9;
8.            print(a) ;
9.        }
10.       public void print(int a){
11.           int c = b%a;
12.           System.out.print(c);
13.       }
14.   }
```

Which is the output?

A. 3

B. 0

C. 1

D. Compilation fails due to an error on line 7.

E. Compilation fails due to an error on line 12.

EXPLANATION:

The life of for loop variable x ends with the end of the “for loop” as the scope of variable x is limited to within the “for loop.”Therefore, the value of the static variable will print, because we have used the pre increment operator the output will be 9 and option A is correct.

QUESTION 9/77:

Given

```
1.    class Program{
```

```
2.      static int x = 8;
3.      public static void main(String[] args) {
4.          for(int x=0;x<3;x++);
5.          System.out.print(++x);
6.      }
7.  }
```

Which is the output?

A. 9

B. 8

C. 4

D. 3

E. Compilation fails.

EXPLANATION:

Java source can have any number of non public classes, but only one public class. And when there is a public class its name should be same as the file name.

Option C is correct since we can't have more than one package statement per source code file.

QUESTION 10/77:

A java source code file can't have

A. Ten non public classes.

B. Only one public class.

C. Two package statements.

D. One import statement.

E. Two methods with same name.

EXPLANATION:

Java source can have any number of non public classes, but only one public class. And when there is a public class its name should be same as the file name. So, this source file should name as "A.java" and option A is correct.

QUESTION 11/77:

Following is the content of a java source code.

```
1.  class Program{ }
2.
3.  public class A{
4.      public static void main(String args[]){    }
5.  }
6.
7.  interface I{}
```

Which is the following should be source file name?

A. A.java

B. I.java

C. Program.java

D. Any name can be used.

E. This is invalid source file.

EXPLANATION:

To compile java program we use javac command and we need to use full file name for compiling for example;

javac Program.java

To run a java program we can use java command with just only class name, without any extension part, like follows

java Program

So, option D is correct.

QUESTION 12/77:

Which of the following can be used to run following class file?

Program.class

A. java Program.class

B. javac Program.class

C. javac Program

D. java Program

E. None of above.

EXPLANATION:

The length of array “a” is 6, So, the value of the variable “i" is 5. Execution of while loop will print the array elements in reverse order as the variable “i” has initial value 5.So, printing of the elements start with 6 (a[5]) and runs till the value of variable “i” equal to 0. Therefore, the first element of the array “a” (a[0]) prints last.

A is incorrect as the value of “i” start with 0. D is incorrect as there is no way of throwing exceptions. E is incorrect as the code has no compile errors.

So, the answer will be C.

QUESTION 13/77:

Consider following code segment

```
1.  public class Program{
2.      public static void main(String args[]){
3.        int []a = {1,2,3,4,5,6};
4.          int i = a.length - 1;
5.
6.          while(i > 0){
7.              System.out.print(a[i]);
8.              i--;
9.          }
```

```
10.                     }
11.             }
```

What would be the output, if it is executed as a program?

A. 123456

B. 654321

C. 65432

D. An Exception

E. Compilation fails.

EXPLANATION:

The condition in while loop is (i==3 && j<5) fails in the first attempt, So, there is no output. The whole expression fails because i==3 fails, the initial condition is (1==3) is false. Then the interpreter won't check the second condition. Option C is correct.

QUESTION 14/77:

Given

```
1.          class Program {
2.          public static void main(String[] args){
3.                  byte i=1,j=1;
4.                          while (i==3 && j<5) {
5.                                  System.out.print (i +" "+j+" ");
6.                                  i++;
7.                                  j+=2;
8.                          }
9.                  }
10.         }
```

What is the output?

A. 0 0 1 2 3 4

B. 0 0 1 1 2 2 3 3

C. No output

D. Compilation fails

E. An Exception

EXPLANATION:

The condition in while loop is (i==3 || j<5)won't fail even though the condition i==3 fails in the first attempt, If the conjunction '||' evaluates the whole expression to true, if any one of the condition is true in the expression. Option A is correct.

QUESTION 15/77:

Consider following code segment

```
1.          class Program{
```

```
2.      public static void main(String[] args){
3.          byte i=1,j=1;
4.          while (i==3 || j<5) {
5.              System.out.print (i +" "+j+" ");
6.              i++;
7.              j+=2;
8.          }
9.      }
10.  }
```

What is the output?

A. 1 1 2 3 3 5

B. 1 1 2 2 3 3

C. No output

D. Compilation fails

E. An Exception

EXPLANATION:

The compiler complaints “error: unreachable statement” at line. This happens because the condition of while is always false and the statement k=30 never executes. So, the answer will be E.

QUESTION 16/77:

Given

```
1.  class Program{
2.      public static void main(String[] args){
3.          int k = 20;
4.          while (false){
5.              k = 30;
6.              System.out.println(k);
7.          }
8.      }
9.  }
```

What would be the output, if it is executed as a program?

A. Prints 30

B. Prints 20

C. No output

D. An Exception

E. Compilation fails

EXPLANATION:

The length of array “a” is 6, So, the value of the variable “i" is 5. Execution of loop will print the array elements in reverse order as the variable “i” has initial value 5. So, printing of the elements start with 6 (a[5]) and runs till the value of variable “i” equal to 0. Therefore, the first element of the array “a” (a[0]) prints last.

A is incorrect as the value of “i” start with 5. D is incorrect as there is no way of throwing exceptions. E is incorrect as the code has no compile erros.

So, the answer will be B.

QUESTION 17/77:

Consider following code segment

```
1.  public class Program{
2.      public static void main(String args[]){
3.          char []a = {'a','b','c','d','e','f'};
4.          int i = a.length - 1;
5.
6.          do{
7.              System.out.print(a[i]);
8.          } while((i--) > 0);
9.      }
10. }
```

What is the output?

A. abcdef

B. fedcba

C. fedcb

D. An Exception

E. Compilation fails.

EXPLANATION:

for Loop Syntax: for (<initialization>; <termination>; <updation>)

The for loop construct for(; ;) is allowed in java. The initialization, condition and update are null statements are allowed. Since the elements are null, the loop runs infinitely. Option A is correct.

QUESTION 18/77:

Consider following code segment

```
1.  class  Program{
2.      public static void main(String[] args){
3.          int i=1;
4.          for(; ;){
5.          }
6.      }
7.  }
```

What is the output?

A. Repeats infinite loop

B. Compile Error

C. Runtime Error

D. Program terminates immediately

EXPLANATION:

The loop starts at 0 and increments up to 127 then to -128 and increments up to 127 and repeats. The loop runs forever.

So, the option A is Correct.

QUESTION 19/77:

Given

```
1.    class  Program {
2.        public static void main(String[] args){
3.            byte i=0;
4.            for(; ;){
5.                System.out.print(i++  +"  “);
6.            }
7.        }
8.    }
```

What is the output?

A. Infinite loop prints 0 to 127 and -128 to 0

B. Infinite loop prints nothing

C. Prints 0 to 127 and -128 to 0 once

D. An Exception

E. Compilation fails.

EXPLANATION:

The for loop in the above snippet is called for each loop. The loop works by iterating (automatically forwarding to next) the elements of the array. Here ele is float type and the answer is B.

QUESTION 20/77:

Given

```
1.    class Program{
2.      public static void main(String args[]){
3.        int MyArr[]={15,33,76,44};
4.
5.        for(float ele:MyArr){
6.            System.out.print(ele+" “);
7.        }
8.      }
```

```
9.          }
```

What is the output?

A. 15 33 76 44

B. 15.0 33.0 76.0 44.0

C. An Exception.

D. Compilation fails.

EXPLANATION:

Initially k value is 3, prints the value, and before going for the next iteration it checks condition, The loop terminates at k=0. Option C is correct.

QUESTION 21/77:

Given

```
1.          public class Program {
2.              public static void main(String[] args){
3.                  int k = 3;
4.                  do{
5.                      System.out.print(k+" ");
6.                  }while(--k >0);
7.              }
8.          }
```

What is the output?

A. 2 1 0

B. 1 2 3

C. 3 2 1

D. 0 1 2

E. 2 1 0 -1

EXPLANATION:

The loop terminates when a<6 fails in while condition or a++>2 is true. So, here a++>2 matches first and print 1 only. Option D is correct.

QUESTION 22/77:

Given

```
1.          class Program{
2.              public static void main(String args[]){
3.                  int a=1;
4.                  do {
5.                      System.out.println( a+"  ");
6.                      if(a++>2) break;
```

```
7.                          }while(a<6);
8.                  }
9.          }
```

What is the output?

A. 1 2 3 4 5

B. 1 2 3 4 5 6

C. 1

D. 1 2 3

E. Compilation fails.

EXPLANATION:

At line 7 there is a break instruction So, compiler will give an error al line 8 of type unreachable code. Option E is correct.

QUESTION 23/77:

Given

```
1.          class Program{
2.                  public static void main(String args[]){
3.                          int x=10;
4.                          do {
5.
6.                                  if(x-->2){
7.                                          break;
8.                                          System.out.print( x+"  ");
9.                                  }
10.                         }while(x>5);
11.                 }
12.         }
```

What is the output?

A. 9 8 7 6

B. 9 8 7 6 5

C. 9 8

D. 1 2 3

E. Compilation fails.

EXPLANATION:

We have added statements after the break statement in if block, So, compiler sees that there is no way to reach printing statement at line 10 and produce a compile time error. Option E is correct.

QUESTION 24/77:

Given

```
1.  class Program{
2.      public static void main(String args[]){
3.      out:for(int i=1; i<=3; i++){
4.      System.out.println();
5.      for(int j=1; j<=3; j++){
6.          if(i == j){
7.              break out;
8.              System.out.print(j+" ");
9.          }
10.     }
11. }
```

What is the output?

A. 1

B. 1 2

C. 1 1 2

D. 1 2 3

E. Compilation fails.

EXPLANATION:

When the condition i==j is true then the outer loop continues without executing the statements after continue. The loop continues to the outer loop (continue out;). So, option A is correct.

QUESTION 25/77:

Given

```
1.  public class Program{
2.      public static void main(String args[]){
3.          out:for(int i=1; i<=3; i++){
4.              System.out.println();
5.              for(int j=1; j<=3; j++){
6.                  if(i == j) continue out;
7.                  System.out.print(j+" ");
8.              }
9.          }
10.     }
11. }
```

What is the output?

A.

1

1 2

B. 1 1 2

C. 1 2 3

D.

1 2 3

1 2 3

E. Compilation fails.

EXPLANATION:

This code expects input from java console, So, if we couldn't provide argument, then at line 3, it will throw ArrayIndexOutOfBoundsException because when program tries to access second element of the array it throws ArrayIndexOutOfBounds Exception.

Even if we pass value if it is not convertible to a number then it will result a NumberFormatException. Option C is correct.

QUESTION 26/77:

Given

```
1.    class Program{
2.        public static void main(String args[]){
3.            int y = Integer.parseInt(args[1]);
4.            System.out.print(y);
5.        }
6.    }
```

Which of the following invokes will result a arrayindexoutofboundsexception?

A. java Program 1 2 3

B. java Program A c 1 2

C. java Program 1

D. java Program 10 A

E. None of above

EXPLANATION:

At line 2, we have declared wrapper, since we haven't given any value to the variable i, it will take its' default value which is null. So, at line 5 trying to invoke the intValue method on null object result a NullPointerException. So, catch block will print Exception, and then "Done" will be printed since finally block executes every time, no matter there is exception or not. Hence, option C is correct.

QUESTION 27/77:

Given

```
1.    public class Program{
2.        static Integer i;
3.        public static void main(String args[]){
```

```
4.                          try{
5.                              System.out.println(i.intValue());
6.                          }catch(NullPointerException e){
7.                              System.out.println(“Error!”);
8.                          }finally{
9.                              System.out.print("Done");
10.                         }
11.                 }
12.         }
```

What is the output?

A. 0 Done

B. Done

C. Error! Done

D. null Done

E. Compilation fails.

EXPLANATION:

It is not necessary to have catch clause for a try block every time, we can have both catch and finally or any one of them, for any try block. So, option A is incorrect.

Option B is incorrect since it is perfectly legal to have nested try/catch.

Option D is correct.

QUESTION 28/77:

Which of the following is true?

A. We must have a finally clause for a try block.

B. If there are mutilple catch blocks then finally block should be there.

C. We can’t nest few try/catch blocks.

D. None of above.

EXPLANATION:

You can’t enter code between the try and catch clause. Here line 7 causes the failure. So, the answer is C. If you remove line 7, then the code will compile fine and provide “error” as the output, in which case the answer would be B.

A and E are incorrect as the code produces compile time errors and won’t run So, there won’t be any output. In the catch clause we have used “Exception e” which will catch almost every exception So, D is incorrect.

QUESTION 29/77:

Given

```
1.          class Program{
2.              public static void main(String args[]){
3.                  int x = 0, y=10;
4.                  try{
```

```
5.                                  y /=x;
6.                          }
7.                          System.out.print("Divide by 0");
8.                          catch(Exception e){
9.                                  System.out.print(“error”);
10.                         }
11.                 }
12.         }
```

What is the output?

A. 0

B. error

C. An uncaught exception is thrown at runtime.

D. Compilation fails.

EXPLANATION:

If method is declared to throw a checked exception then calling method should catch or specify that exception.

Option E declared to throw a FileNotFoundException which is a checked exception, but at main method, we haven’t catch or throw the FileNotFoundException Hence, it will not compile. So, option D is correct.

QUESTION 30/77:

Given

```
1.          public class Program{
2.
3.                  public static void main(String args[]){
4.                          readFile();
5.                   }
6.
7.                  //insert here
8.          }
```

Which insert at line 7, will result compile time error?

A. public static void readFile() throws RuntimeException{ /* Codes */}

B. public static void readFile() throws NullPointerException{ /* Codes */}

C. public static void readFile() throws StackOverflowError{ /* Codes */}

D. public static void readFile() throws FileNotFoundException{ /* Codes */}

E. public static void readFile() throws ClassCastException { /* Codes */}

EXPLANATION:

The NegativeArraySizeException is thrown by the JVM when code uses an illegal index to access an array. Hence, option D is correct.

QUESTION 31/77:

Which of the following exception thrown by the JVM when code uses an negative array size for creating an array?

A. NullPointerException

B. NumberFormatException

C. IllegalArgumentException

D. NegativeArraySizeException

E. ArrayIndexOutOfBoundsException

EXPLANATION:

The derived most class should be caught first. The order of classes at the same hierarchy level is not important.

So, correct order should be C, B, A and Exception and the answer is D.

QUESTION 32/77:

Given the following set of classes:

```
class ExceptionA extends Exception {}
class ExceptionB extends A {}
class ExceptionC extends B {}
```

What is the correct sequence of catch blocks for the following try block?

```
try {
 // codes
}
```

A. Catch Exception, A, B,C

B. Catch Exception, C, B, and A

C. Catch A, B, C, and Exception

D. Catch C,B,A and Exception

E. Any order.

EXPLANATION:

At line 4, first evaluate the 1 *2 and 3 * 4, which reduces the expression to the following:

```
int x = 2 + 12 - 5;
```

Then, evaluate the remaining terms in left-to-right order, resulting in a value of x of 9.

At line 5, we have values in the same order, but with set of parentheses:

```
int y = 1 * 2  +  3 * (4 -5);
```

This time you would evaluate the reduce operator (4-5), which reduces the expression to the following:

```
int y = 1 *2 + 3 * -1
```

So, the value of y would be -1. So, finally 8 will be printed and the answer is D.

QUESTION 33/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          int x = 1 * 2  +  3 * 4 -5;
5.          int y = 1 * 2  +  3 * (4 -5);
6.          System.out.println(y+x);
7.      }
8.  }
```

What is the result?

A. 18

B. -2

C. 9

D. 8

E. Compilation fails.

EXPLANATION:

The instaceof operator is a comparison operator. It compares an object to a specified type. You can use it to test if an object is an instance of a class, an instance of a subclass, or an instance of a class that implements a particular interface. Hence, option B is correct.

QUESTION 34/77:

Which of the following is a comparison operator?

A. +

B. intanceof

C. +=

D. --

E. >

EXPLANATION:

The unary operators require only one operand; they perform various operations such as incrementing/decrementing a value by one, negating an expression, or inverting the value of a boolean. So, here only ! is unary operator and the answer is B.

Option A, and D are incorrect as they are assignment operators.

QUESTION 35/77:

Which of the following can consider as unary operator?

A. =

B. !

C. ==

D. +=

E. +-

EXPLANATION:

Input to switch statement doesn't need to be final So, A is incorrect. Default block is not necessary then B is incorrect. Enum can be used in input of statement and then C is incorrect.

D is the answer since only the byte, short, char, and int primitive data types are allowed.

QUESTION 36/77:

Which of the following is true about switch?

A. Input to switch statement should be final.

B. There should be a default block in switch.

C. Enum can't be used as case or switch expression.

D. We can't use long for case or switch expression.

E. None of above

EXPLANATION:

Option B is correct.

QUESTION 37/77:

Given

```
1.    public class Program{
2.        public static void main(String[] args){
3.
4.            String a = "A";
5.            final String A = "a";
6.
7.            switch(a){
8.                default : System.out.println("default");
9.                case A : System.out.println("1");
10.               case "A" : System.out.println("2");break;
11.           }
12.       }
13.   }
```

What is the output?

A. 1

B. 2

C. 12

D. Compilation fails due to error at line 9.

E. Compilation fails due to multiple errors.

EXPLANATION:

When using == to check equality of primitives, values of the primitives are checked, primitive type not considered, So, here test will return true. Hence, option B is correct.

Oracle Reference : http://docs.oracle.com/javase/tutorial/java/IandI/objectclass.html

QUESTION 38/77:

Given

```
1.  public class Program{
2.      public static void main(String[] args){
3.
4.          float f  = 100.2f;
5.          double d = 100.2f;
6.
7.          System.out.print((f == d));
8.
9.      }
10. }
```

What is the output?

A. false

B. true

C. Compilation fails due to line 4.

D. Compilation fails due to line 5.

E. Compilation fails due to multiple errors.

EXPLANATION:

Option A is correct.

QUESTION 39/77:

Given

```
1.  public class Program{
2.
3.      public static void main(String[] args) {
4.          int x = 3, y = 5;
5.          String print = "";
6.
7.          if(x < 5)
8.          if(y > 0)
9.          if(x < y)
10.         print += "1";
11.         else print += "2";
12.         else print += "3";
```

```
13.                     else print += "4";
14.                     System.out.println(print);
15.             }
16.     }
```

What is the output?

A. 1

B. 12

C. 13

D. 124

E. Compile error.

EXPLANATION:

Option E is correct as the code fails to compile due to line 10. When using a primitive or reference type variables as case constants, we should keep in mind that they should be compile time constant. Here the variable i is final but it is not compile time as the value is assigned later. So, trying to use the variable i at line 10, causes a compile time error

QUESTION 40/77:

Given

```
1.      public class Program{
2.
3.              public static void main(String[] args) {
4.                      final int i;
5.                      i = 1;
6.                      final int y = 2;
7.
8.                      switch(2){
9.                              case 0   : {System.out.print("A");}
10.                             case i   : System.out.print("B");
11.                             default  : System.out.print("default"); break;
12.                             case y   : System.out.print("C");
13.                     }
14.             }
15.     }
```

What is the output?

A. ABdefault

B. Bdefault

C. default

D. C

E. Compilation fails

EXPLANATION:

The compiler complains " error: read(int) is already defined in Overload (Compile time binding)". This happens because the compiler unable distinguishes the methods because they have same signature (Signature of a method can be defined as name of the method along with number and type and order of arguments). Option E is correct.

QUESTION 41/77:

Given

```
1.    class Program{
2.
3.        int a,b;
4.
5.        void read(int a){
6.            System.out.println(a);
7.        }
8.
9.        int read(int a){
10.           System.out.println(a+"   "+b);
11.           return 0;
12.       }
13.
14.       public static void main(String[] args){
15.           new Program().read(2);
16.       }
17.   }
```

What is the output?

A. 0 2

B. 2

C. 0 0

D. An Exception

E. Compilation fails.

EXPLANATION:

Passing of an array to the method is pass by reference in java. So, that whatever the updates applied on the reference variable will be reflected to the parameter in the calling method. Option B is correct.

QUESTION 42/77:

Given

```
1.    class Program{
2.        void incrBy(int a[]){
```

```
3.                      for(int i=0;i<a.length;i++)
4.                              a[i]++;
5.              }
6.
7.              public static void main(String args[]){
8.                      int b[]={7,3,6,8,9};
9.                      Program r = new Program();
10.                     r.incrBy(b);
11.                     for(int i=0;i<b.length;i++)
12.                             System.out.print(b[i]);
13.             }
14.     }
```

What is the output?

A. 73689

B. 847910

C. 84791

D. An Exception

E. Compilation fails.

EXPLANATION:

Option A is correct.

QUESTION 43/77:

Given

```
1.      class Program{
2.              void incrBy(int a){
3.                      a=a+10;
4.                      System.out.println(a+" ");
5.              }
6.
7.              public static void main(String args[]){
8.                      int a=10;
9.                      System.out.println(a+" ");
10.                     Program r = new Program();
11.                     r.incrBy(a);
12.                     System.out.println(a+" ");
13.             }
14.     }
```

What is the output?

A. 10 20 10

B. 10 20 20

C. 20 20 20

D. 10 10 10

E. Compilation fails.

EXPLANATION:

h.aboutme() calls aboutme() method in the Honda class and the super.aboutme() calls the Super class(Car) mehod aboutme() and prints the output which is as in option B. The benefit here is that the reuse of the functionality what we did in the super class.

QUESTION 44/77:

Given

```
1.      class Car{
2.          void aboutme(){
3.              System.out.println("I am a Car");
4.          }
5.      }
6.
7.      class  Honda extends Car{
8.          void aboutme(){
9.              super.aboutme();
10.             System.out.println("I am Honda");
11.         }
12.         public static void main(String args[]){
13.             Honda h = new Honda();
14.             h.aboutme();
15.         }
16.     }
```

What is the output?

A.

I am Honda

I am Car

B.

I am Car

I am Honda

C. I am Honda

D. I am Car

E. Compilation fails.

EXPLANATION:

If we won't define any method in the sub class then the method of the super class is called. The benefit is that when the same functionality is required in the sub class, then you can extend and use the super class methods. Option A is correct.

QUESTION 45/77:

Given

```
1.      class A{
2.          void method(){
3.              System.out.println("Method of A");
4.          }
5.      }
6.
7.      class B extends A{
8.
9.          public static void main(String args[]){
10.             B bobj=new B();
11.             bobj.method();
12.         }
13.     }
```

What is the output?

A. Method of A

B. An Exception

C. Compile Error at line 9

D. Compile Error at line 13

E. Compile Error at line 12

EXPLANATION:

We can use any access modifier Constructor, So, C is incorrect and D is incorrect. E is correct as a constructor should not have a return type.

A is incorrect as when there is a user defined constructor, the default constructor vanishes. B is incorrect as the default constructor has "super();".

QUESTION 46/77:

Which are true?

A. Default constructor will be there, even if we declare our own constructor.

B. Default constructor has only "this();" .

C. When we create our own constructor we can't use private access modifier.

D. We should not use "private" access modifier with constructors

E. A constructor should not have a return type even "void".

EXPLANATION:

We can use the keyword “extends” to extend either two classes or two interfaces. In the case of a class, it doesn’t matter whether it is abstract or not. We can use the keyword “implements” to implement an interface to a class. Multiple interfaces can be implemented by class or interface but not multiple classes. So, B, D and E are correct.

A is incorrect as we tried to implement an abstract class, but the abstract class should be extended. C is incorrect as we can’t extend more than one class in java.

QUESTION 47/77:

Given

- A and E are Classes
- B and D are interfaces
- C is an abstract class

Which is invalid?

A. class F implements B ,C{ }

B. class F implements B{ }

C. class F extends E{ }

D. class F extends E implements B{ }

E. class F implements B,D{ }

EXPLANATION:

Option B is the correct answer as the overloaded methods can change the return type.

Option A and C are incorrect as the overloaded method must change the argument list. They can throw new or broader exceptions. So, A, C, D, E and F are incorrect.

QUESTION 48/77:

Which is true?

A. Overloaded method may change the argument list.

B. Overloaded method may change the return type.

C. Overloaded method should not declare new or broader checked exception.

D. All of the above

EXPLANATION:

Reference type can be an interface or abstract class but the object must not So, this code compiles fine. As Sub extends an abstract class Ab, and implements an interface I , it can be treated as Ab or I. D is correct and C is not as both “instanceof” tests return true. So, the output will be true true.

QUESTION 49/77:

Given

```
1.  interface I{
2.      //codes
3.      }
```

```
4.
5.      abstract class Ab{
6.          //codes
7.          }
8.
9.      class Sub extends Ab implements I{
10.         //codes
11.         }
12.
13.     class Program{
14.         public static void main(String [] args){
15.             I i = new Sub();
16.             Ab ab = new Sub();
17.
18.             boolean t1 = i instanceof I ,t2 =  ab instanceof Sub;
19.
20.             System.out.print(t1 +" " + t2);
21.         }
22.     }
```

Which are true?

A. Compilation fails due to multiple errors.

B. Compilation fails due to error at line 15 as I is an interface.

C. The output will be true false

D. The output will be true true

E. Compilation fails due to error at line 16 as Ab is an abstract class.

EXPLANATION:

Interface methods are by default public. So, the meth1() method in the interface is a public method. We are trying to override with a more restrictive access level (default) at line8, So, the compilation fails due to line 8. So, option C is correct.

QUESTION 50/77:

Given

```
1.      interface I{
2.      void meth1();
3.      }
4.
5.
6.      class Sub implements I{
```

```
7.
8.      void meth1(){
9.      System.out.print("class Sub");
10.     }
11.
12.     }
13.
14.     class program{
15.         public static void main(String [] args){
16.             I i = new Sub();
17.             i.meth1();
18.
19.         }
20.     }
```

The output will be?

A. No output

B. Class sub

C. Compilation fails due to error on line 8

D. Compilation fails due to error on line 17

E. Compilation fails due to multiple errors

EXPLANATION:

Statement II and III are correct as the overloaded method must change the argument list and can change the return type and also, the access modifier. They can throw new or broader exceptions. So, the answer C is correct.

Statement I is incorrect as it hasn't change the argument list, So, A, B, D and E are incorrect.

QUESTION 51/77:

Given

```
1.      class A{
2.          protected void meth1(){
3.              System.out.print("A");
4.              }
5.          }
6.          class B extends A{
7.              //overload meth1() here
8.          }
```

Consider

I. public void meth1(){ }

II. private final int meth1(int i){ return i;}

III. private final void meth1(String s)throws Exception{ }

Which is true?

A. I only

B. II only

C. II and III only

D. I and II only

E. All

EXPLANATION:

Option D is correct as we give int array as the argument.

Option C and E are incorrect because this method is marked with "void" So, it doesn't return anything.

A is incorrect as this method is marked with "static"So, it can be invoked using a class name. B is incorrect as it marked with "public" which is the least restrictive access modifier.

QUESTION 52/77:

Consider following method

```
public static void method(int[] a){
                                    }
```

Which is true?

A. This method can only invoke through an instance of containing class.

B. This method has marked with highest restrictive access modifier.

C. This method can return null.

D. This method takes int array as argument.

E. This method returns int array.

EXPLANATION:

Since method doesn't take anything as the argument as the argument list is empty So, option C is incorrect.

A is incorrect as this method has "default" as the access modifier but adding it is optional. As it adds default and it is not the lowest restrictive access modifier, B is incorrect. So, option D is correct.

QUESTION 53/77:

Consider following method

```
int method(){   //codes    }
```

Which is true?

A. This method is incorrect as it doesn't have access modifier.

B. This method has marked with lowest restrictive access modifier.

C. This method takes int array as argument.

D. This method returns int.

EXPLANATION:

A is correct as a constructor must have the same name of the class.

They can't have return type So, E is correct. B is valid as they can be overloaded but not override. D is valid as interfaces can't have constructors but abstract classes do have them. Every class including abstract must have at least one constructor, So, C is correct. So, the answer will be F.

QUESTION 54/77:

Choose invalid statement.

A. A constructor may have same name as the class.

B. Constructors can be overloaded

C. A class must have at least one constructor.

D. Interfaces do not have constructors.

E. Constructors should not have a return type.

F. none

EXPLANATION:

A is correct as a constructor can have any access modifier. E is like a default constructor as it only has "super();" inside. So, E is also, correct.

B is incorrect as we can't use final with Constructors. D is incorrect as we should use call super() or this() before any declaration inside a constructor. C is incorrect as there must be no return type (then it would be a method), even void.

QUESTION 55/77: Select Multiple Answers

Given

```
1.      class Town{
2.          //What would be correct constructor for this class
3.          }
```

What would be correct constructor for Town class? (Choose 2)

A. private Town(int x){ }

B. final Town(){ }

C. void Town(Strings){ }

D. Town(){ System.out.print("Town");
super(); }

E. Town(){ }

EXPLANATION:

Code will compile fine So, B and D are incorrect Arrays are objects so, So, the value "true" will be returned, Therefore, answer C is correct and A is incorrect.

E is incorrect as the length (number of elements) of this array is 4.

QUESTION 56/77:

Given

```
1.      class Program{
2.          public static void main(String[] args) {
```

```
3.                  int arr[] = {1,2,3,4};
4.                  System.out.print(arr instanceof Object);
5.          }
6.      }
```

Which is correct?

A. Will produce output as false.

B. Compilation fails due to error at line 3.

C. Will produce output as true.

D. Compilation fails due to error at line 4.

E. Length of this array is 3.

EXPLANATION:

Indexing of array elements begins with zero. So, [1] refers to the second element of an array. So, here, a[3] refers to the fourth element of array a. Its value is 4 and we have assigned 4 to b[0][2]. We have assigned an octal value to a[2] So, the value of the element is 43 in decimal. And we have assigned 43 in decimal to b[2][1]. Therefore, both will print true and option C is correct.

QUESTION 57/77:

Given

```
1.      class Program{
2.          public static void main(String[] args) {
3.              int arr[] = { 1,2,053,4};
4.              int arr1[][] = { {1,2,4} , {2,2,1},{0,43,2}};
5.              System.out.print(arr[3]==arr1[0][2] );
6.              System.out.print(" " + (arr[2]==arr1[2][1]));
7.          }
8.      }
```

Which is the output?

A. false false

B. false true

C. true true

D. An Exception.

E. Compilation fails

EXPLANATION:

We can use several steps to complete the creation of a multi-dimensional array by using anonymous arrays. We have assigned two one dimensional int arrays to the two dimensional array a. So, the code compiles fine and produces the output as 2. Thus,answer D is correct, and A and B are incorrect.

QUESTION 58/77:

Given

```
1.  class Program{
2.      public static void main(String[] args) {
3.          int a[][] = new int[3][];
4.          a[1] = new int[]{1,2,3};
5.          a[2] = new int[]{4,5};
6.          System.out.print(a[1][1]);
7.      }
8.  }
```

Which is the output?

A. 2

B. 3

C. 5

D. Exception will be thrown in runtime.

E. Compilation fails due to line 4.

EXPLANATION:

Size or length of the array is the number of elements in an array. Index positions of an array starts from zero, So, final. So, the last index is 1 less than the length. Here array has 5 elements, So, the size of array is 5 and last index is 4. Hence, option E is correct.

QUESTION 59/77:

The size and the last index of following array respectively.

char[] chs = {'1', '2', '3', '4', '5'};

A. 5,5

B. 5,6

C. 6,4

D. 4,4

E. 5,4

EXPLANATION:

Following is the correct syntax for creating the anonymous array.

type[]{ elements separated by comma.}

Remember that you do not specify a size when using anonymous array creation syntax. The size is derived from the number of items (comma-separated) between the curly braces. So, option B is correct.

As explained above options A and C incorrect.

Option D is incorrect as [] is missing.

QUESTION 60/77:

Which of the following is a correct anonymous array?

A. new int[3]{1,2,3,4};

B. new int[]{1,2,3,4};

C. new int[4]{1,2,3,4};

D. new int{1,2,3,4};

EXPLANATION:

Option A is correct as when we create an array, elements of the array are initialized to default values. So, here all elements will be 0 since the default value of int is zero. At line 4, when initializing array we have passed binary literal which is perfectly legal. Passed binary value is 5 in decimal So, the array will store 5 elements So, output will contain 5 zeros.

QUESTION 61/77:

Given

```
1.    public class Program{
2.        public static void main(String[] args) {
3.
4.            int []ints = new int[0b101];
5.            int len = ints.length;
6.
7.            for(int i : ints)
8.            System.out.print(i);
9.        }
10.   }
```

Which is the output?

A. 00000

B. 0000

C. nullnullnullnull

D. A NullPointerException is thrown.

E. Compilation fails due to error at line 4.

EXPLANATION:

Option D is correct since all are primitive literals, they are double, long and char.

Option A is incorrect as “a” is a String literal.

Option B is incorrect as True is incorrect literal is should be true.

Option C is incorrect as ‘AB’ is illegal; char literal can only has one letter.

QUESTION 62/77:

Which of the following set contains only primitive literals?

A. 1, ‘c’, “a”

B. 1, 1.5f, True

C. ‘AB’, 10, 3L

D. 1.5D, 2l, ‘c’

E. None of above.

EXPLANATION:

Option B and D correct since 0.0 are the default values for floating point types, doubles and floats

Option A is incorrect since 0 is the default value for long variable.

Option C is incorrect since false is the default value for boolean variable.

Option E is incorrect since 0 is the default value for a byte variable.

QUESTION 63/77: Select Multiple Answers

0.0 is the default value of? (Choose 2)

A. long

B. double

C. boolean

D. float

E. byte

EXPLANATION:

Option E is correct as the code fails to compile. There are three types of variables class, instance and local. Local variables (variables which are declared inside a method) must be initialized before using them. But at line 4 the statement "int x, y = 100", only initialized the variable y. Therefore, at line 5, trying to use the variable "x" before initialize, causes a compile time error.

Other options are incorrect as the code fails to compile. However, if we could initialize the variable "x" then the code would compile and output will be the value of the variable "x". This concept is called as variable shadowing. Option E is correct.

QUESTION 64/77:

Given

```
1.  public class Program{
2.      static int x = 50;
3.      public final static void main(String[] args){
4.          int x, y = 100;
5.          System.out.print(x);
6.      }
7.  }
```

What is the result?

A. 50

B. 100

C. 0

D. An Error is thrown at the runtime, stating that, Main method not found in class Pro.

E. Compilation fails.

EXPLANATION:

We can use various literal types for numeric type variables, ie

binary literal – 0b11

Octal literal – 012

Decimal literal – 11

Hexadecimal literal – 0xff

At line 5, we have used octal literal, for an octal literals we can used only 0-7 digits, So, using 8 will result compile time error. Hence, option D is correct.

QUESTION 65/77:

Given

```
1.  public class Program{
2.
3.      static int x = 0b01;
4.      static int y = 0xF;
5.      static int z = 028;
6.
7.      public static void main(String args[]){
8.
9.          System.out.println(x+z+y);
10.     }
11.
12. }
```

What is the output?

A. 28

B. 31

C. 36

D. Compilation fails due to an error on line 5.

E. Compilation fails due to multiple errors.

EXPLANATION:

Option B is correct since an object is eligible for GC when there is no reference to an object in a currently live thread.

Like other any program, java applications can run out of memory. So, option A is incorrect.

Option C is incorrect as the purpose of the GC is to remove the objects which have no reference in a currently live thread.

Option D is incorrect as the JVM decide when to run GC whether we request or not.

QUESTION 66/77:

Which is true?

A. Java applications never run out of memory as Garbage Collector manages the memory.

B. An object is eligible for GC when there is no live thread can reach it.

C. The purpose of GC is to delete objects that there is no use at the moment.

D. When you request GC to run, it will start to run immediately.

E. None of above.

EXPLANATION:

Option A is correct as only the statement I is correct.

Statement I is correct since the variables inside a method are called as local variables.

Statement II is incorrect as the local variables are not initialized to its default value.

Statement III is incorrect as we can't say anything about the variable "x" as it could be a local variable or instance variable because it doesn't use any modifier. If It has used a modifier except "final" then it should be an instance variable.

QUESTION 67/77:

Which is/are true?

I. The variables declared inside a method are called as local variables.

II. The local variables are initialized to its default value.

III. The variable "x" in the declaration "int x = 10" should be an instance variable.

A. Only I.

B. Only II.

C. Only III.

D. Only I and II.

E. Only I and III.

EXPLANATION:

At line 3, we have created a string object and its value is "Java". Because no new assignment was made, the new String object created with the"concat()" method and the "replace()" method was abandoned instantly. At the end of the day, value of the "s" is only "java". Therefore, the answer is D.

So, A, B, and, C are incorrect as explained above.

QUESTION 68/77:

Given

```
1.  class Program{
2.      public static void main(String [] args){
3.              String s = "Java";
4.              s.concat(" SE 7");
5.              s.replace('7','8');
6.              System.out.print(s);
7.      }
8.  }
```

What is the result?

A. Java SE 7

B. Java SE 8

C. Java SE

D. Java.

E. Compilation fails.

EXPLANATION:

Line 3 creates a new String object and gives it the value “Java”. At line 4, String objects are created (“SE 8” and “Java SE 8”). At line 5, a new String object is created (“java”). So, four objects are created.But “s” only refers to “Java,” So, answer D is correct and others are incorrect.

QUESTION 69/77:

Given

```
1.      class Program{
2.          public static void main(String [] args){
3.              String s = "Java";
4.              s.concat(" SE 8");
5.              s.toLowerCase();
6.              System.out.print(s);
7.              //how many?
8.          }
9.      }
```

How many String objects are created when code reaches to line 7?

A. 1

B. 2

C. 3

D. 4

E. 5

EXPLANATION:

StringBuilder class has append, insert and delete methods (and more). StringBuilder objects are also, changeable like StringBuffer objects.(StringBuilder and StringBuffer classes are different as StringBuffer classes methods are synchronized.)We can see “chained methods” in line 4and it is perfectly legal.

The Option B is correct as inserting “The latest” to the beginning of “java” and appending adds “1.7” to the end of it. Then deleting removes 7 and then appending again adds 8 to end. So, A and C are incorrect.

QUESTION 70/77:

Given

```
1.      class Program{
2.          public static void main(String [] args){
3.              StringBuilder sb =new StringBuilder("Java");
```

```
4.  sb.insert(0 , "The latest ").append("Version is").append("
1.7").delete(28,29).append("8");
5.  System.out.print(sb);
6.  }
7.  }
```

What is the result?

A. The latest Java version is 1.7

B. The latest Java version is 1.8

C. The latest Java version is 8

D. Compilation fails due to an error at line 4.

E. Compilation fails due to multiple errors.

EXPLANATION:

B is correct as we pass “ABC” as args[0] So, the equals test will produce true and equal will print.

D is correct as we don’t provide any string to main method So, trying to access args[0] at line 5 will throw aArrayIndexOutOfBoundsException. So, A is incorrect as above.

E is incorrect as we provide “abc” as args[0] and it will evaluate as false in the equals test, So, nothing will print.

This code compiles fine So, C is incorrect.

QUESTION 71/77: Select Multiple Answers

Given

```
1.  public class Program{
2.
3.  public static void main(String args[]){
4.  String s1 = new String("ABC");
5.  String s2 = new String(args[0]);
6.
7.  if(s1.equals(s2)){
8.  System.out.print("Equal");
9.  }
10.  }
11.  }
```

What is true? (Choose 2)

A. If we use command line invocation, java Program ABC abc, There will be no output

B. If we use command line invocation, java Program ABC, The output will be Equal.

C. Compilation fails.

D. If we use command line invocation, java Program, an ArrayIndexOutOfBoundsException will be thrown.

E. If we use command line invocation, java Program abc, The output will be Equal.

EXPLANATION:

First here we have LocalDate instance and then invoking the atTime will combine passed time with LocalDate value, and result will be LocalDateTime object, So, we have to have LocalDateTime reference, Hence, option D is correct.

QUESTION 72/77:

Given

LocalDate loc = LocalDate.of(2015, 1, 21);

_____ ldt = loc.atTime(1,1);

Which of the following can be used to fill the blank?

A. Instant

B. LocalDate

C. LocalTime

D. LocalDateTime

E. Date

EXPLANATION:

Option A is correct, the systemDefaultZone() obtains a clock that returns the current instant using the best available system clock, converting to date and time using the default time-zone.

Clock class constructor in protected So, we can't use new keyword and invoke the constructor to create Clock instant, Hence, option B is incorrect.

The instant() method return instant from the clock and it is not static, So, option D is incorrect.

Option C is incorrect as there is no such a method, System method expect ZoneId as the argument So, Option C is incorrect.

QUESTION 73/77:

Which of the following is correct way of creating Clock instance?

A. Clock c = Clock.systemDefaultZone();

B. Clock c = new Clock();

C. Clock c = Clock.system();

D. Clock c = Clock.instant();

E. Clock c = Clock.getClock());

EXPLANATION:

The ArrayList class subList method,

public List<E> subList(int fromIndex,int toIndex)

Above method Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive. (If fromIndex and toIndex are equal, the returned list is empty.)

So, subList method won't change the list, but at line 16, passing ArrayList to removeAll method will remove all equal elements of the list, So, 0 and 2 will be removed. Hence, the final result will be [2, 1, 5, 4]. So, option A is correct.

QUESTION 74/77:

Given

1. import java.util.*;

```
2.
3.          public class Program{
4.                  public static void main(String[] args){
5.                          List<Integer> list = new ArrayList<Integer>();
6.                          List<Integer> rem = new ArrayList<>();
7.                          rem.add(0); rem.add(3);
8.                          list.add(2);
9.                          list.add(3);
10.                         list.add(1);
11.                         list.add(5);
12.                         list.add(4);
13.                         list.add(0);
14.
15.                         list.subList(0, 2);
16.                         list.removeAll(rem);
17.
18.                         System.out.println(list);
19.                 }
20.         }
```

What is the output?

A. [2, 1, 5, 4]

B. [1, 5, 4]

C. [5, 4]

D. An Exception is thrown.

E. Compilation fails.

EXPLANATION:

A functional interface should have only one abstract method So, option C is correct and B is incorrect.

It is optional to include default or static method in functional interface, So, option A is incorrect.

It is optional to have @FunctionalInterface notation, for a functional interface. So, option E is incorrect.

QUESTION 75/77:

A Functional interface should

A. Contain a static method.

B. Contain one or more abstract methods.

C. Contain only one abstract method.

D. Not contain abstract methods.

E. Contain the @FunctionalInterface notation.

EXPLANATION:

Functional interfaces provide target types for lambda expressions and method references. Each functional interface has a single abstract method (SAM), called the functional method for that functional interface, to which the lambda expression's parameter and return types are matched or adapted. It can also, include other methods such as static and default methods.

Also, we should remember that interfaces can have default and/or static methods, but they should be non abstract, this is not just for functional interfaces, this applies for all java interfaces.

Given interface satisfies all above requirements So, it can be considered as a functional interface, and it is optional to use @FuncationalInterface annotation.

So, option D is correct.

QUESTION 76/77:

Given

```
1.    public interface FunInterface{
2.       void function();
3.    }
```

Which of the following is true about above code fragment?

A. We can't consider above as functional interface.

B. We need to remove line 2 to make this code a functional interface.

C. If we add @FunctionalInterface anannotation, then above can consider above is functional interface.

D. None of above.

EXPLANATION:

From the given output we can decide that, period is for 3 months and 2 days.

Option A is correct since there we have used overloaded version of the "of" method of Period class.

```
public static Period of(int years, int months, int days)
```

This could use to obtain a Period representing a number of years, months and days. Here we have passed 3 for months and 2 for days which will satisfy expected result.

QUESTION 77/77:

Which of the following will create period which will result following output when invoking toString on that period?

```
P3M2D
```

A. Period period = Period.of(0,3,2);

B. Period period = Period.of(3,2);

C. Period period = Period.of (2).plusMonths(3);

D. Period period = Period.of(2,3);

E. Period period = Period.plus(3,2);