PROGRAMMING IN JAVA

Assignment 03

TYPE OF QUESTION: MCQ

Number of questions: 10 Total marks**: 10 × 1 = 10**

QUESTION 1:

In Java, what is the role of the `public static void main(String[] args)` method?

a. **Initialization method**

b. **Execution entry point**

c. **Constructor**

d. **Destructor**

Correct Answer:

b. **Execution entry point**

Detailed Solution:

The `public static void main(String[] args)` method is the entry point for the execution of a Java program.

QUESTION 2:

Consider the following code.

```
class First {
    static void staticMethod() {
        System.out.println("Static Method");
    }
}

class MainClass {
    public static void main(String[] args) {
        First first = null;
        First.staticMethod();
    }
}
```

What is the output of the above code?

a. **Static Method**

b. **Throws a NullPointerException**

c. **Compile-time error**

d. **Run time error**

Correct Answer:

a. **Static Method**

Detailed Solution:

The provided Java code will compile and execute successfully without any exceptions. When calling a static method, it doesn't require an instance of the class. Therefore, you can call the static method `staticMethod()` from class `First` using the null reference `first.`

QUESTION 3:

Consider the following code.

```
class access {
    public int x;
    private int y;

    void cal(int a, int b) {
        x = a + 1;
        y = b;
    }
    void print() {
        System.out.println(" " + y);
    }
}
public class access_specifier {
    public static void main(String args[]) {
        access obj = new access();
        obj.cal(2, 3);
        System.out.print(obj.x);
        obj.print();
    }
}
```

What is the output of the above code?

a. **2 3**

b. **3 3**

c. **Runtime Error**

d. **Compilation Error**

Correct Answer:

b. **3 3**

Detailed Solution:

The first 3 is printed by System.out.println(obj.x); because x was set to 3 (2 + 1) in the cal method.
The second 3 is printed by obj.print(); because y was set to 3 in the cal method.
Although y is a private variable, it is still accessible within the methods of the same class. Therefore, the print method can access and print its value.

QUESTION 4:

Members which are not intended to be inherited are declared as

a. **Public members**
b. **Protected members**
c. **Private members**
d. **Private or Protected members**

Correct Answer:

c. **Private members**

Detailed Solution:

Private access specifier is the most secure access mode. It doesn't allow members to be inherited. Even Private inheritance can only inherit protected and public members.

QUESTION 5:

Which type of inheritance leads to diamond problem?

a. **Single level**

b. **Multi-level**

c. **Multiple**

d. **Hierarchical**

Correct Answer:

c. **Multiple**

Detailed Solution:

When 2 or more classes inherit the same class using multiple inheritance and then one more class inherits those two base classes, we get a diamond like structure. Here, ambiguity arises when same function gets derived into 2 base classes and finally to 3rd level class because same name functions are being inherited.

QUESTION 6:

Consider the following code.

```
class superDemoClass {
    final void func() {
        int a = 20;
        System.out.println("value of a = " + a);
    }
}

class subDemoClass extends superDemoClass {
    void func() {
        int b = 60;
        System.out.println("value of b = " + b);
    }
}

class demo {
    public static void main(String[] args) {
        subDemoClass subc = new subDemoClass();
        subc.func();
    }
}
```

What is the output of the above code?

a. **error: func() in subDemoClass cannot override func() in superDemoClass**

b. **value of b = 60**

c. **value of a = 20**

d. **None of the above**

Correct Answer:

a. **error: func() in subDemoClass cannot override func() in superDemoClass**

Detailed Solution:

Here in this program, the subclass is trying to override the final method of the superclass, i.e. it is trying to change the behavior of the final method. The behavior of the final method cannot be changed in the subclass. In other words, the final method cannot be overridden in any subclass because the final method is a complete method. Therefore, error: func() in subDemoClass cannot override func() in superDemoClass

QUESTION 7:

Consider the following code.

```
class StaticScopeDemo {
  static int x = 5;

  public static void main(String[] args) {
      int x = 10;
      {
          int x = 15; // Compilation Error
          System.out.println(x);
      }
  }
}
```

What is the output of the above code?

a. **15**

b. **Compilation Error**

c. **5**

d. **10**

Correct Answer:

b. **Compilation Error**

Detailed Solution:

The block within `main()` tries to declare a local variable `x` that has the same name as an already existing variable in the same scope, which causes a compilation error. Variable names must be unique within the same scope

QUESTION 8:

Consider the following code.

```
public class NptelExample {
  public static int fun(int n) {
      if (n == 0) {
          return 1;
      }
      return n * fun(n - 1);
  }

  public static void main(String[] args) {
      System.out.println(fun(5));
  }
}
```

What is the output of the above code?

a. **5**

b. **24**

c. **120**

d. **Runtime Error**

Correct Answer:

c. **120**

Detailed Solution:

The fun method is a recursive function that calculates the factorial of a number. For fun(5), the computation is 5 * 4 * 3 * 2 * 1 = 120.

QUESTION 9:

If a variable of primitive datatype in Java is declared as final, then

a. **It cannot get inherited**

b. **Its value cannot be changed**

c. **It cannot be accessed in the subclass**

d. **All of the above**

Correct Answer:

b. **Its value cannot be changed**

Detailed Solution:

A final variable of a primitive data type cannot change its value once it has been initialize.

QUESTION 10:

Consider the following code.

```
class static_out {
    static int x;
    static int y;

    void add(int a, int b) {
        x = a + b;
        y = x + b;
    }
}

public class static_use {
    public static void main(String args[]) {
        static_out obj1 = new static_out();
        static_out obj2 = new static_out();
        int a = 2;
        obj1.add(a, a + 1);
        obj2.add(5, a);
        System.out.println(obj1.x + " " + obj2.y);
    }
}
```

What is the output of the above code?

a. 7 7.4

b. 6 6.4

c. 7 9

d. 9 7

Correct Answer:

c. 7 9

Detailed Solution:

`x` and `y` are static variables, so they are shared across all instances of the class `static_out`.
When `obj1.add(a, a + 1)` is called, `x` and `y` are updated to 5 and 8, respectively.
When `obj2.add(5, a)` is called, `x` and `y` are updated to 7 and 9, respectively.
The final values of `x` and `y` after all method calls are 7 and 9, respectively, which are printed by `System.out.println(obj1.x + " " + obj2.y);`.