PROGRAMMING IN JAVA

Assignment 05

TYPE OF QUESTION: MCQ

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

QUESTION 1:

Which of the following statement(s) is/are true about `finally` in Java?

I. **The `finally` block is executed regardless of whether an exception is thrown or not.**
II. **A `finally` block can exist without a `catch` block.**
III. **The `finally` block will not execute if `System.exit()` is called in the `try` block.**
IV. **A `finally` block can have a `return` statement, but it is not recommended to use.**

a. **I and II**

b. **II and III**

c. **I, II and III**

d. **I, II, III and IV**

Correct Answer:

d. **I, II, III and IV**

Detailed Solution:

The `finally` block always executes except when the JVM exits using `System.exit()`. It can exist without a `catch` block and may contain a return statement, though this is not recommended.

QUESTION 2:

Consider the following code.

```
interface A {
    int x = 10;
    void display();
}
class B implements A {
    public void display() {
        System.out.println("Value of x: " + x);
    }
}
public class Main {
    public static void main(String[] args) {
        B obj = new B();
        obj.display();
    }
}
```

What will be the output of the above code?

- **a. Value of x: 10**
- **b. Value of x: 0**
- **c. Compilation Error**
- **d. Runtime Error**

Correct Answer:

- **a. Value of x: 10**

Detailed Solution:

Variables in interfaces are `public`, `static`, and `final` by default. Hence, the value of `x` is accessible in the `display` method.

QUESTION 3:

Consider the following code.

```
class NPTEL {
    public static void main(String[] args) {
        try {
            int a = 5;
            int b = 0;
            System.out.println(a / b);
        } catch (ArithmeticException e) {
            System.out.print("Error ");
        } finally {
            System.out.print("Complete");
        }
    }
}
```

What will be the output of the above code?

a. 5 Complete

b. Error Complete

c. Runtime Error

d. Compilation Error

Correct Answer:

b. Error Complete

Detailed Solution:

An `ArithmeticException` is caught in the `catch` block, which prints “Error”. The `finally` block executes afterward, printing “Complete”.

QUESTION 4:

Which of the following is TRUE regarding abstract class and an interface in Java?

I. **Abstract classes can contain constructors, but interfaces cannot.**
II. **Interfaces support multiple inheritance, but abstract classes do not.**
III. **Abstract classes can have both abstract and concrete methods, whereas interfaces only had abstract methods before Java 8.**

a. **I, II and III**

b. **II only**

c. **I and II only**

d. **II and III only**

Correct Answer:

a. **I, II and III**

Detailed Solution:

Abstract classes can have constructors and concrete methods. Interfaces support multiple inheritance. Before Java 8, interfaces could only contain abstract methods, but now they can include default and static methods.

QUESTION 5:

Which of the following is a *checked exception* in Java?

a. **NullPointerException**

b. **ArrayIndexOutOfBoundsException**

c. **IOException**

d. **ArithmeticException**

Correct Answer:

c. **IOException**

Detailed Solution:

IOException is a checked exception, meaning it must be either caught or declared in the `throws` clause of a method. The others are unchecked exceptions, which do not require explicit handling.

QUESTION 6:

Consider the following code.

```
interface Demo {
    void display();
}

class Test implements Demo {
    public void display() {
        System.out.println("Hello, NPTEL!");
    }
}

public class Main {
    public static void main(String[] args) {
        Test obj = new Test();
        obj.display();
    }
}
```

What will be the output of the above code?

a. **Hello, NPTEL!**

b. **Compilation Error**

c. **Runtime Error**

d. **No Output**

Correct Answer:

a. **Hello, NPTEL!**

Detailed Solution:

The `Test` class implements the `Demo` interface and provides a definition for the `display` method. When `display()` is called, it prints “Hello, NPTEL!”.

QUESTION 7:

Consider the following code.

```
interface Calculator {
    void calculate(int value);
}

class Square implements Calculator {
    int result;
    public void calculate(int value) {
        result = value * value;
        System.out.print("Square: " + result + " ");
    }
}

class Cube extends Square {
    public void calculate(int value) {
        result = value * value * value;
        super.calculate(value);
        System.out.print("Cube: " + result + " ");
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator obj = new Cube();
        obj.calculate(3);
    }
}
```

What will be the output of the above code?

a. **Square: 9 Cube: 9**

b. **Cube: 27 Square: 9**

c. **Square: 9 Square: 27 Cube: 27**

d. **Square: 9 Cube: 27 Square: 27**

Correct Answer:

a. **Square: 9 Cube: 9**

Detailed Solution:

The `Cube` class overrides the `calculate` method of the `Square` class. In the `Cube` class's `calculate` method, `super.calculate(value)` is called, which executes the `calculate` method of the `Square` class. First, "Square: 9" is printed by the superclass method. Then, the overridden method in `Cube` prints "Cube: 9".

QUESTION 8:

Which keyword is NOT used by Java during exception handling?

- **a. try**
- **b. catch**
- **c. final**
- **d. finally**

Correct Answer:

- **c. final**

Detailed Solution:

In Java, exceptions are handled using the `try`, `catch`, and `finally` blocks. The `try` block contains code that might throw an exception, the `catch` block handles specific exceptions, and the `finally` block executes regardless of whether an exception occurs.

QUESTION 9:

A method that potentially generates a checked exception must include this keyword in its method signature:

a. **throw**

b. **extend**

c. **throws**

d. **extends**

Correct Answer:

c. **throws**

Detailed Solution:

Any Java class that generates a checked exception and does not handle it internally must use the throws keyword to alert other methods of its instability.

QUESTION 10:

Consider the following code.

```
class Test extends Exception {}

class Main {

  public static void main(String args[]) {
    try {
      throw new Test();
    } catch (Test t) {
      System.out.println("Got the Test Exception");
    } finally {
      System.out.println("Inside finally block ");
    }
  }
}
```

What will be the output of the above code?

a. **Got the Test Exception**
   Inside finally block

b. **Got the Test Exception**

c. **Inside finally block**

d. **Compiler Error**

Correct Answer:

a. Got the Test Exception
Inside finally block

Detailed Solution:

In Java, the finally is always executed after the try-catch block. This block can be used to do the common cleanup work.