PROGRAMMING IN JAVA

Assignment 5

TYPE OF QUESTION: MCQ

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

QUESTION 1:

Which of the following is an incorrect statement about interfaces?

a. **Interfaces specifies what class must do but not how it does.**
b. **Interfaces are specified public if they are to be accessed by any code in the program.**
c. **All variables in interface are implicitly final and static.**
d. **All variables are static and methods are public if interface is defined public.**

Correct Answer:

d. **All variables are static and methods are public if interface is defined public.**

Detailed Solution:

All methods and variables are implicitly public if interface is declared public.

QUESTION 2:

How do you access a static method of an interface?

a. **Using the interface name**
b. **Using the method name directly**
c. **Through an object of the interface**
d. **Through an implementation class**

Correct Answer:

a. **Using the interface name**

Detailed Solution:

Static methods in an interface are accessed using the interface name, similar to static methods in classes.

QUESTION 3:

What is the output of the below Java program with an Interface?

```
interface Car {
  int basePrice = 1000;
}

public class InterfaceTest2 implements Car {
  void changePrice() {
    basePrice = 2000;
    System.out.print(basePrice);
  }

  public static void main(String[] args) {
    new InterfaceTest2().changePrice();
  }
}
```

a. **1000**
b. **2000**
c. **Compiler error**
d. **None of the above**

Correct Answer:

c. **Compiler error**

Detailed Solution:

Java Interface treats its variables like constants. So, the classes implementing Interfaces, can not reassign values to the variables.

QUESTION 4:

What happens when we access the same variable defined in two interfaces implemented by the same class?

a. **Compilation failure**
b. **Runtime Exception**
c. **The JVM is not able to identify the correct variable**
d. **The interfaceName. variableName needs to be defined**

Correct Answer:

d. **The interfaceName.variableName needs to be defined**

Detailed Solution:

Explanation: The JVM needs to distinctly know which value of variable it needs to use. To avoid confusion to the JVM interfaceName.variableName is mandatory.

QUESTION 5:

Predict the output of following Java program

```
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 ");
    }
  }
}
```

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.

QUESTION 6:

What happens if an exception is not caught in the catch block?

a. **The finally block handles it**
b. **The exception is ignored**
c. **The exception is thrown to the caller method**
d. **The program terminates immediately**

Correct Answer:

c. **The exception is thrown to the caller method**

Detailed Solution:

If an exception is not caught in the catch block, it is propagated back to the caller method.

QUESTION 7:

What will be the output of the following Java program?

```
class exception_handling {

  public static void main(String args[]) {
    try {
      System.out.print("Hello" + " " + 1 / 0);
    } catch (ArithmeticException e) {
      System.out.print("World");
    }
  }
}
```

a. **Hello**
b. **World**
c. **HelloWorld**
d. **Hello World**

Correct Answer:

b. **World**

Detailed Solution:

System.out.print() function first converts the whole parameters into a string and then prints, before “Hello” goes to output stream 1 / 0 error is encountered which is cached by catch block printing just “World”.

QUESTION 8:

What is the output of the below Java code with Exceptions?

```
public class ExceptionTest2 {

  public static void main(String[] args) {
    try {
      int ary[] = { 10, 20, 30 };
      int tempt = ary[4];
    } catch (ArrayIndexOutOfBoundsException e1) {
      System.out.println(e1.getMessage());
    } catch (Exception e2) {
      System.out.println("Some exception");
    }
  }
}
```

a. **Index 4 out of bounds for length 3**
b. **Index 4 out of bounds for length 3**
   Some exception
c. **Some exception**
d. **No exception occurs**

Correct Answer:

a. **Index 4 out of bounds for length 3**

Detailed Solution:

IndexOutOfBoundsException is raised by TRY block. Observe the order of catching the exceptions. Always catch a Subclass exception before a Superclass exception.

QUESTION 9:

What is the output of the Java code with FINALLY block and RETURN statement?

```
public class ExceptionTest6 {

  static void show() {
    try {
      System.out.println("inside TRY");
      return;
    } finally {
      System.out.println("inside FINALLY");
    }
  }

  public static void main(String[] args) {
    show();
  }
}
```

a. **inside TRY**
b. **inside TRY**
   inside FINALLY
c. **inside FINALLY**
d. **Compiler error**

Correct Answer:

b. **inside TRY**
   inside FINALLY

Detailed Solution:

Even if a RETURN statement is present at the last line of TRY block, the control is not returned to the calling method. The JVM searches for the suitable FINALLY block and executes it before returning. So, the FINALLY block has higher priority than a RETURN statement.

QUESTION 10:

What is the purpose of the finally block in Java exception handling?

a. **To handle an exception**
b. **To catch an exception**
c. **To clean up resources after a try block**
d. **None of the above**

Correct Answer:

c. **To clean up resources after a try block**

Detailed Solution:

The purpose of the finally block in Java exception handling is to clean up resources after a try block. The finally block is executed whether or not an exception is thrown in the try block. It is typically used to release resources such as open files, database connections, or network sockets that were acquired in the try block. The finally block ensures that these resources are released even if an exception is thrown, which helps prevent resource leaks and other issues. The finally block is optional, but it is good practice to use it when dealing with resources that need to be released.