PROGRAMMING IN JAVA

Assignment 4

TYPE OF QUESTION: MCQ

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

QUESTION 1:

Which of these access specifiers must be used for `main()` method?

a. **private**

b. **public**

c. **protected**

d. **default**

Correct Answer:

b. **public**

Detailed Solution:

`main()` method must be specified `public` as it called by Java run time system, outside of the program. If no access specifier is used then by default member is `public` within its own package & cannot be accessed by Java run time system.

QUESTION 2:

What is the output of the below Java Code Snippet with `protected` access modifier?

```
// Teacher.java ------------------
package nptel1;
public class Teacher {
  protected void showMarks() {
    System.out.println("100 Marks");
  }
}
```

```
// Student.java ------------------
package nptel2;
import nptel1.*;
public class Student extends Teacher {
  void show() {
    showMarks();
  }
  public static void main(String[] args) {
    Student st1 = new Student();
    st1.show();
  }
}
```

a. **100 marks**

b. **No output**

c. **Compiler error**

d. **None of the above**

If a method does not specify an access modifier, it defaults to package-private. This means the method would only be accessible within the same package. However, since showMarks in Teacher is explicitly marked protected, it is accessible to subclasses in other packages.

Correct Answer:

a. **100 marks**

Detailed Solution:

Through inheritance, one can access a `protected` variable or method of a class even from outside the package. Here, we accessed `Teacher` class of `nptel1` from `Student` class of `nptel2`.

QUESTION 3:

What is the process by which we can control what parts of a program can access the members of a class?

a. **Polymorphism**

b. **Augmentation**

c. **Encapsulation**

d. **Recursion**

Correct Answer:

c. **Encapsulation**

Detailed Solution:

Encapsulation in Java is the process by which data (variables) and the code that acts upon them (methods) are integrated as a single unit. By encapsulating a class's variables, other classes cannot access them, and only the methods of the class can access them.

QUESTION 4:

Consider the 2 programs:

```
// Main1.java ------------------
public class Main1{
    public static void main(String args[]){
        int number = 10;
        System.out.println(number++ + ++number);
    }
}
```

10 12 12 11

```
// Main2.java ------------------
public class Main2{
    public static void main(String args[]){
        int number = 10;
        System.out.println(++number + number++);
    }
}
```

11 11 11 12

a. **Both pre-increment and post-increment operators becomes pre-increment during print.**

b. **Both pre-increment and post-increment operators becomes post-increment during print.**

c. **Both Main1 and Main2 classes give the same output.**

d. **Pre-increment and post-increment operators don't work during print.**

Correct Answer:

c. **Both Main1 and Main2 classes give the same output.**

Detailed Solution:

The output of both the program are 22. Therefore, option **c** is correct and we can eliminate option **d** that the operators don't work. Further, the operators are doing exactly what they are supposed to do i.e. pre-increment first increases the values and post-increment increases the value during the next operation. The print statement is the next operation; hence it received the post incremented value as well making option **a** and **b** invalid.

QUESTION 5:

Which is the least restrictive access modifier in Java?

a. **public**

b. **private**

c. **protected**

d. **default**

Public>protected>defualt>private

Correct Answer:

a. **public**

Detailed Solution:

A variable or a method marked `public` is available to all outside classes irrespective of package that a class lives in. So, `public` is the least restrictive access modifier in Java.

QUESTION 6:

Choose the correct syntax of a Java Package below.

a. **package PACKAGE_NAME;**

b. **package PACKAGE_NAME.*;**

c. **pkg PACKAGE_NAME;**

d. **pkg PACKAGE_NAME.*;**

Correct Answer:

a. **package PACKAGE_NAME;**

Detailed Solution:

A package declaration statement should end with a package name but not with *.

QUESTION 7:

In Java, the default package refers to a package that does not have a specified name. Classes in this unnamed package can be accessed by other classes in the same unnamed package but cannot be accessed from classes in named packages. It's generally used for small, simple applications and is not recommended for larger projects.

What is the default package in Java?

a. **It is a package that contains all built-in classes.**

b. **It is a package that needs to be defined as default.**

c. **It is a package that does not have a name.**

d. **It is a package used for importing external libraries.**

Correct Answer:

c. **It is a package that does not have a name.**

Detailed Solution:

If you create two classes, ClassA and ClassB, without specifying a package declaration at the top of the file, they are considered to be in the default package.

The default package is a package without a specified name. It's not recommended to use the default package for your classes.

Since both ClassA and ClassB are in the default package (no package name is defined at the top), ClassB can access ClassA directly. However, if ClassB were in a named package (e.g., package mypackage;), it would not be able to access ClassA without moving ClassA into the same package or giving ClassA a package name and importing it accordingly.

```
public class ClassA {
    public void display() {
        System.out.println("Hello from ClassA!");
    }
}
```

```
public class ClassB {
    public static void main(String[] args) {
        ClassA a = new ClassA();
        a.display();
    }
}
```

QUESTION 8:

A package is a collection of:

a. **classes**

b. **interfaces**

c. **editing tools**

d. **classes and interfaces**

Correct Answer:

d. **classes and interfaces**

Detailed Solution:

A package is a collection of classes, methods and interfaces.

QUESTION 9:

In Java, can a subclass in a different package access a superclass's protected method?

a. **Yes, without any restrictions.**

b. **Yes, but only if they are in the same package.**

c. **No, protected methods are not accessible by subclasses.**

d. **No, protected methods are only accessible within the same class.**

Correct Answer:

b. **Yes, but only if they are in the same package.**

Detailed Solution:

Yes, a subclass in a different package can access a superclass's protected method, as long as they are in a subclass relationship.

Math.cos(2*Math.PI));

QUESTION 10:

Consider the program given below. What will be the output if the program is executed?

```
public class Main{
    public static void main(String args[]){
        System.out.println(cos(2*PI));
    }
}
```

a. **It will give compile-time error**

b. **It will give run-time error**

c. **1.0**

d. **3.14**

In Java, the java.lang package is imported by default. This package includes many core classes that are frequently used, such as:

Object: The superclass of all classes.
String: For working with text.
System: Provides access to system resources like input/output.
Math: Contains mathematical functions and constants, like Math.PI and Math.cos().
Integer, Double, Boolean, etc.: Wrapper classes for primitive types.

Correct Answer:

a. **It will give compile-time error**

Detailed Solution:

The program gives a compile time error as the Math class is missing.
The static import statement needs to be used to import the static members (e.g., PI) of java.lang.Math.

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

This will allow the program to use PI directly.