PROGRAMMING IN JAVA

Assignment 4

TYPE OF QUESTION: MCQ

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

QUESTION 1:

Which is the keyword used to specify the default access modifier in java?

a. **default**
b. **DEFAULT**
c. **package**
d. **“There is no keyword”**

Correct Answer:

d. **“There is no keyword”**

Detailed Solution:

If you do not specify any access modifier before a variable or method, it is considered to be declared under the `default` access modifier.

QUESTION 2:

What is the output of the Java program with access modifiers?

```
package nptel1;

public class ProgrammingInJava {

  public String week = "FOUR";
}
// ---------------
package nptel2;

import nptel1;

public class Course {

  public static void main(String[] args) {
    ProgrammingInJava java = new ProgrammingInJava();
    System.out.println(java.week);
  }
}
```

a. **FOUR**
b. **Runtime Error**
c. **null**
d. **Compiler Error**

Correct Answer:

d. **Compiler Error**

Detailed Solution:

In order to use instance variables, one should import either a specific Class or all Classes of a Java Package. So, without using `import nptel1.*;` or `import nptel1.ProgrammingInJava;` there will be a compiler error: `ProgrammingInJava` cannot be resolved to a type.

QUESTION 3:

What is the output of the below Java Code Snippet with access modifiers?

```
package nptel1;

public class ProgrammingInJava {

  int weeks = 12; // course duration
}

// ------------------
package nptel1;

public class Course {

  public static void main(String[] args) {
    ProgrammingInJava obj = new ProgrammingInJava();
    System.out.println("Weeks = " + obj.weeks);
  }
}
```

a. **Weeks = 0**
b. **Weeks = 12**
c. **No Error, blank output**
d. **Compiler error**

Correct Answer:

b. **Weeks = 12**

Detailed Solution:

The `default` variables or methods of class can be called inside any class of the same Package.

QUESTION 4:

Which of the following is the correct representation of access modifiers in order of increasing visibility?

a. **private < default < protected < public**
b. **private < protected < default < public**
c. **public < protected < default < private**
d. **protected < default < private < public**

Correct Answer:

a. **private < default < protected < public**

Detailed Solution:

Here's the order of the access modifiers from the least restrictive to the most restrictive:

`public` > `protected` > `default` > `private`

Thus, the correct representation of access modifiers in order of increasing visibility:

`private` < `default` < `protected` < `public`

QUESTION 5:

Which of the following package stores all the standard java classes?

a. **java.util**
b. **java.lang**
c. **java.java**
d. **java.packages**

Correct Answer:

b. **java.lang**

Detailed Solution:

In Java, two packages `java.lang` package and a no-name package (also called default package) are imported by default in all the classes of Java. Default Package doesn't have a name but it is present by default and thus, it is named the default package. Java Virtual Machine imports these packages by default in Java internally. And that is the reason we are able to access all the classes of these packages. We are not required to explicitly import `java.lang` package or any of its classes like we import other packages and their classes such as `java.math` or `java.util`.

QUESTION 6:

Which of the following is/are true about packages in Java?

1. **Every class is part of some package.**
2. **All classes in a file are part of the same package.**
3. **If no package is specified, the classes in the file go into a special unnamed package.**
4. **If no package is specified, a new package is created with folder name of class and the class is put in this package.**

a. **Only 1, 2 and 3**
b. **Only 1, 2 and 4**
c. **Only 4**
d. **Only 1 and 3**

Correct Answer:

a. **Only 1, 2 and 3**

Detailed Solution:

In Java, a package can be considered as equivalent to C++ language's namespace. Thus, every class is part of some package. All classes in a file are part of the same package. If no package is specified, the classes in the file go into a special unnamed package.

QUESTION 7:

What is the output of following Java program?

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

class ProgrammingInJava {

  public static void main(String args[]) {
    out.println("Welcome!");
  }
}
```

a. **Compiler Error**
b. **Runtime Error**
c. **Welcome!**
d. **None of the above**

Correct Answer:

c. **Welcome!**

Detailed Solution:

Static import is a feature introduced in Java programming language (versions 5 and above) that allows members (fields and methods) defined in a class as `public` `static` to be used in Java code without specifying the class in which the field is defined. The only time we need to pay attention to packages is when we have a name conflict . For example both, `java.util` and `java.sql` packages have a class named `Date`. So if we `import java.util.*;` and `import java.sql.*;` then the compiler will not be able to figure out which Date class do we want.

QUESTION 8:

Which of these access specifiers can be used for an interface?

a. **Public**
b. **Protected**
c. **private**
d. **All of the mentioned**

Correct Answer:

a. **Public**

Detailed Solution:

Access specifier of an interface is either public or no specifier. When no access specifier is used then default access specifier is used due to which interface is available only to other members of the package in which it is declared, when declared public it can be used by any code.

QUESTION 9:

Which of the following is the correct way of implementing an interface NPTEL by class Java?

a. **class Java extends NPTEL {}**
b. **class Java implements NPTEL {}**
c. **class Java imports NPTEL {}**
d. **none of the mentioned**

Correct Answer:

b. **class Java implements NPTEL {}**

Detailed Solution:

`interface` is inherited by a `class` using `implements`.

QUESTION 10:

What will be the output of the following Java program?

```
interface calculate {
  void cal(int item);
}

class display implements calculate {
  int x;

  public void cal(int item) {
    x = item * item;
  }
}

class interfaces {
  public static void main(String args[])
        {
            display arr = new display();
            arr.x = 0;
            arr.cal(2);
            System.out.print(arr.x);
        }
}
```

a. **0**
b. **2**
c. **4**
d. **Compiler Error**

Correct Answer:

c. **4**

Detailed Solution:

There is no error in the program and the `cal()` function is called successfully, computes the square of `item` and stores in `x`. Which is successfully printed.