Object Oriented Programing

Name	Muhammad Ishfaq
Student Id	16002
Department	CS2
Instructor	M.Ayub Khan

Iqra National University Peshawar, Pakistan

Q1. a. Why access modifiers are used in java, explain in detail Private and Default Access modifiers?

Answer (a)

The access modifiers are used in Java to specify the scope or accessibility of a class, method, field, or constructor. Using access modifiers in java we can alter the access level of class, method, field, and constructor.

Private Access Modifier Details:

Private Specifiers **accomplish** the accessibility of the lowest level. Private Fields & methods could only be accessed inside the same class in which the fields and methods belong. Private Fields and methods are not visible within child classes and not inherited by child classes. Hence, Private access **Modifier** is opposite to the public access **Modifier**. Using Private Specifier we can **accomplish** encapsulation & data hiding from the outside of class.

Default Access Modifiers Details:

Default: When no access modifier exists for a class, data member, or method.
It is also known as to be having the default access specifier by default.
The class, data members or methods which does not declare using any access specifier. For Example, having default access specifier are accessible only and only within same package.

b. Write a specific program of the above mentioned access modifiers in java.

Answer (b)

Private Access Modifiers Program:

```
class MyClass {
   private String name;
}
public class Main {
   public static void main(String[] main){
      MyClass Obj1 = new MyClass();
      Obj1.name = "Muhammad Ishfaq";
   }
}
```

Output of Private Access Modifier:

```
Main.java:18: error: name has private access in Data
      Obj1.name = "Muhammad Ishfaq";
```

Default Access Modifiers Program:

```
package default_modifier;
public class Default_modifier {
   void Show()
         {
         System.out.println("Hi, My id is 16002, and this program is about Default
Access modifier");
         }

   public static void main(String[] args) {
     Default_modifier Obj1=new Default_modifier();
     Obj1.Show();
   }  }
```

Output of Default Access Modifier:

```
Hi, My id is 16002, and this program is about Default_modifier
BUILD SUCCESSFUL (total time: 0 seconds)
```

Q2. a. Explain in detail Public and Protected access modifiers?

Answer (a)

Public Access Modifier Details:

The public access specifier is specified using a reserve word *public*.
The public access specifier has the huge range of scope among all access modifiers.
Classes, data members or methods which are declared publically are accessible from outside of the program. There isn’t restriction on the scope of public data members.

Protected Access Modifier Details:

Protected data member and method has only accessibility by the classes of the same package & the subclasses present in any package. You can say that protected access specifier is same as default access specifier with 1 exception that it has visibility in subclasses.
Classes can't be declared protected. This access specifier is generally used in parent child relationship.

b. Write a specific program of the above mentioned access modifiers in java.

Answer (b)

Public Access Modifier Program:

```
package package16002;
public class Multiplication{
  public int MultiplyNumbers(int x, int y){
        return x+y;
  }
}
package package2;
import package16002.*;
class MyClass{
  public static void main(String args[]){
    Multiplication Obj1 = new Addition();
    System.out.println("Output is:  "Obj1.MultiplyNumbers (3, 5));
  }
}
```

Output of Public Access Modifier:

```
Output is: 15
```

Protected Access Modifier Program:

```
package package16002;
public class Subtract{
  protected int SubtractNumbers(int m, int n){
        return m-n;
  }
}
```

```
package package2;
import package16002.*;
class MyClass extends Subtract{
  public static void main(String args[]){
        MyClass Obj1 = new MyClass();
        System.out.println("Result is :  ",Obj1.SubtractNumbers(15, 5));
  }
}
```

Output of Protected Access Modifier:

```
Result is:  10
```

Q3. a. What is inheritance and why it is used, discuss in detail?

Answer (a)

The process by which 1 class acquire properties & functionalities of other class is known as inheritance. The purpose of inheritance is to give facility of the reusability of code, hence a class has to write the unique features and rest of the common functionalities & properties may be extended from the other class.
Class that extends features of other class is known as sub class, derived class or child class.
The class which functionalities & properties are inherited by another class is called super class, Base class, or parent class.

Why it is used:
We are using inheritance because it allows us to define new class based on an existing class by extending its common data members and methods.
Using Inheritance we are able to reuse the code and it is used to improve the reusability of our java application.

b. Write a program using Inheritance class on Animal in java.

Answer (b)

```
class Animal {
  public void FIGHT() {
    System.out.println("I am good in fighting");
  }
  public void RUN() {
    System.out.println("I am good in running");
  }  }
class Monkey extends Animal {
  public void CLIMB() {
    System.out.println("I am good in climbing");
  } }
class Main {
  public static void main(String[] args) {
    Monkey Obj1 = new Monkey();
    Obj1.FIGHT();
    Obj1.RUN();
    Obj1.CLIMB();
  }  }
```

Output:

```
I am good in fighting
I am good in running
I am good in climbing
```

Q4. a. What is polymorphism and why it is used, discuss in detail?

Polymorphism considered as one of the crucial feature of Object-Oriented Programming (OOP). Polymorphism allow us to perform action in various form. In other words, it allows us to define 1 interface and have multiple implementations. Word “poly” means many & “morphs” means ways, it means many forms. Polymorphism in Java is broadly divided into 2 types first one is Compile time Polymorphism & the second one is Runtime Polymorphism.

Why it is used:

The reason for why Polymorphism is required in java is because this concept is extensively used in implementing inheritance. It plays important role in allowing objects having various internal structures connected with the same external interface. Polymorphism itself as stated clear, one which mapped for many.

b. Write a program using polymorphism in a class on Employee in java.

```
class Employee{
  public void EmployeeSalery() {
    System.out.println("The Emoployee taking salery from INU
University: ");
  }
}

class Teachers extends Employee{
  public void SALERY() {
    System.out.println("Teachers Taking Saleries between 40k to 50k");
  }
}

class HODs extends Employee{
  public void SALERY() {
    System.out.println("Head of department (HODs) Taking Saleries
between 60k to 70k");
  }
}

class MainClass {
  public static void main(String[] args) {
    Employee Obj1= new Employee();
    Employee ObjTea = new Teachers();
    Employee ObjHod = new HODs();

    Obj1.SALERY();
    ObjTea.SALERY();
    ObjHod.SALERY();
  }
}
```

Output:

```
The Emoployee taking salery from INU University
Teachers Taking Saleries between 40k to 50k
Head of department (HODs) Taking Saleries between 60k to 70k
```

Q5. a. Why abstraction is used in OOP, discuss in detail?

It one of the key concept of object oriented programming languages. The main goal of **abstraction** is to deal with complexity by hiding unnecessary detail from the user. That is very generic **Conception** that not limited to object oriented programming (OOP). You can find it everywhere in real world.
Abstraction is the key elements of better software layout/design. It helpful for encapsulate behavior & also for decouple software elements. When developing with high level of abstraction, you communicate behavior and less implementation.

b. Write a program on abstraction in java.

```
abstract class MyClass {

   public abstract void SHOW();

  public void DISPLAY() {

    System.out.println("I am Display Method");

  } }

class YourClass extends MyClass{

  public void SHOW() {

    System.out.println("I am Show Method ");

  } }

class MyMainClass {

  public static void main(String[] args) {

    YourClass Obj1 = new YourClass();

    Obj1.SHOW();
  Obj1.DISPLAY();
 } }
```

Output:

```
I am Show Method
I am Display Method
```