B.Sc(H) Computer Science Practical Assignment

Programming in Java

Semester II

Submitted by – Harsh Meena (88028)

Q1. What is the difference between Method overloading and constructor overloading in Java?

Write a suitable program to illustrate the following:

a) Default constructor

b) Parameterized constructor

c) Method overloading with different number of parameters

d) Method overloading with different type of parameters

e) Method overloading with different sequence of parameters

A1.

Constructor Overloading	Method Overloading
A constructor is used to initialize the state of an object.	A method is used to expose the behaviour of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor's name must be same as the class name.	The method name may or may not be same as the class name.
If we want to have different ways of initializing an object using different number of parameters, then we must do constructor overloading.	We do method overloading when we want different definitions of a method based on different parameters.

(a)- Default constructor

```
//Java Program to create and call a default constructor
class Uber{
//creating a default constructor
Uber(){System.out.println("Uber is ready");}
//main method
public static void main(String args[]){
//calling a default constructor
Uber b=new Uber();
}
}
```

(b)- Parameterized constructor

```
//Java Program to demonstrate the use of the parameterized constructor.
class Student{
   int id;
   String name;
   //creating a parameterized constructor
   Student(int i,String n){
   id = i;
   name = n;
   }
   //method to display the values
   void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
   //creating objects and passing values
   Student s1 = new Student(111,"Harsh");
   Student s2 = new Student(222,"Nikku");
   //calling method to display the values of object
   s1.display();
   s2.display();
  }
}
```

(c)- Method overloading with different number of parameters

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

(d)- Method overloading with different type of parameters

```
class Adder{
static int add(int a, int b){return a+b;}
static double add(double a, double b){return a+b;}
}
class TestOverloading2{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
}}
```

(e)- Method overloading with different sequence of parameters

```
class DisplayOverloading
{
  public void disp(char c, int num)
  {
     System.out.println("I’m the first definition of method disp");
  }
  public void disp(int num, char c)
  {
     System.out.println("I’m the second definition of method disp" );
  }
}
class Sample
{
  public static void main(String args[])
  {
     DisplayOverloading obj = new DisplayOverloading();
     obj.disp('x', 51 );
     obj.disp(52, 'y');
  }
}
```

Q2. Write the advantages of using packages in java. Write a suitable program that illustrates different levels of protection in classes/subclasses belonging to same package or different packages.

A2. Advantages of using packages in java -

- Java package is used to categorize by the classes and interfaces.
- It is easy to maintained.
- Java package is provide as access protection.
- It may removes naming collision.
- This packages can be provide reusability of code.
- We can create our own package or extend already available package.

Program Code –

```
class ParentClass{
    int a = 10;
    public int b = 20;
    protected int c = 30;
    private int d = 40;

    void showData() {
        System.out.println("Inside ParentClass");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
    }
}

class ChildClass extends ParentClass{

    void accessData() {
        System.out.println("Inside ChildClass");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        //System.out.println("d = " + d);         // private member can't be accessed
    }

}
public class AccessModifiersExample {

    public static void main(String[] args) {
        ChildClass obj = new ChildClass();

        obj.showData();

        obj.accessData();


    }


}
```