Experiment No. 4
Implement a program on method and constructor overloading.
Date of Performance:
Date of Submission:

Aim: Implement a program on method and constructor overloading.

Objective: To use concept of method overloading in a java program to create a class with same function name with different number of parameters.

Theory:

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

Example: This example to show how method overloading is done by having different number of parameters for the same method name.

```
Class DisplayOverloading
{
   public void disp(char c)
   {
       System.out.println(c);
   }
   public void disp(char c, int num)
   {
       System.out.println(c + " "+num);
   }
}
Class Sample
{
  Public static void main(String args[])
  {
```

```
    DisplayOverloading obj = new DisplayOverloading();
    Obj.disp('a');
    Obj.disp('a',10);
  }
}
```

Output:

A

A 10

Java supports Constructor Overloading in addition to overloading methods. In Java, overloaded constructor is called based on the parameters specified when a new is executed.

Sometimes there is a need of initializing an object in different ways. This can be done using constructor overloading.

For example, the Thread class has 8 types of constructors. If we do not want to specify anything about a thread then we can simply use the default constructor of the Thread class, however, if we need to specify the thread name, then we may call the parameterized constructor of the Thread class with a String args like this:

Thread t= new Thread (" MyThread ");

Code:

```
import java.io.*;
class Student {
public void StudentId(String name, int roll_no)
{
System.out.println("Name :" + name + " "
+ "Roll-No :" + roll_no);
}
public void StudentId(int roll_no, String name)
{

System.out.println("Roll-No :" + roll_no + " "
+ "Name :" + name);
}
```

```
}
class Stdetails {
public static void main(String[] args)
{
Student obj = new Student();
obj.StudentId("Mahek", 45);
obj.StudentId(6, "xyz");
}
}
```

```
C:\Mahek Shah CSEDS>javac Stdetails.java

C:\Mahek Shah CSEDS>java Stdetails
Name Mahek Roll-No :45
Roll-No :6 Name :xyz
```

constructor overloading

Code:

```
class Rectangle
{
   int length, width;
   void getData(int x, int y)
   {
      length=x;
      width=y;
   }
   int rectArea()
   {
      int area=length*width;
      return area;
   }   }
class constructoroverloading
{
  public static void main(String args[])
   {
      int area1,area2;
      Rectangle rect1=new Rectangle();
      Rectangle rect2=new Rectangle();
      rect1.length=20;
      rect1.width=30;
      area1=rect1.length*rect1.width;
      rect2.getData(20,10);
      area2=rect2.rectArea();
      System.out.println("Area1="+area1);
```

```
        System.out.println("Area2="+area2);
    }
}
```

```
C:\Mahek Shah CSEDS>javac constructoroverloading.java

C:\Mahek Shah CSEDS>java constructoroverloading
Area1=600
Area2=200
```

Conclusion:

Constructor and function overloading are valuable features in object-oriented programming, particularly when working with Java, as they allow developers to create objects and methods with different initial states, configurations, and behaviors based on the parameters provided during object instantiation or method calls. These overloading techniques simplify object initialization and method usage, improve code reusability, and enhance code readability and maintainability. Constructor overloading, for instance, allows you to provide multiple ways to initialize an object, catering to various scenarios, while function overloading enables you to create methods with the same name but different parameter lists, making your code more versatile and adaptable.