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.*;
import java.util.*;

class User
{
   String name;

   // Default constructor
   User()
   {
      name = "Default";
   }

   // Parameterized constructor
   User(String name)
   {
      this.name = name;
   }

   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
```

```
        User user = new User();
        System.out.println(user.name);

        System.out.println("Enter name: ");
        String name = sc.nextLine();

        User user1 = new User(name);
        System.out.println(user1.name);
    }
}

// Java program to illustrate
// Constructor Overloading
class Box
{
        double width, height, depth;

        Box(double w, double h, double d)
        {
                width = w;
                height = h;
                depth = d;
        }

        Box()
        {
                width = height = depth = 0;
        }

        Box(double len)
        {
                        width = height = depth = len;
        }

        double volume()
        {
                return width * height * depth;
        }
}

// Driver code
public class Test {
        public static void main(String args[])
        {

                Box mybox1 = new Box(10, 20, 15);
                Box mybox2 = new Box();
                Box mycube = new Box(7);

                double vol;
```

```
        // get volume of first box
        vol = mybox1.volume();
        System.out.println("Volume of mybox1 is " + vol);

        // get volume of second box
        vol = mybox2.volume();
        System.out.println("Volume of mybox2 is " + vol);

        // get volume of cube
        vol = mycube.volume();
        System.out.println("Volume of mycube is " + vol);
    }
}
```

Output -

```
Default
Enter name:
Ram
Ram

=== Code Execution Successful ===
```

```
Volume of mybox1 is 3000.0
Volume of mybox2 is 0.0
Volume of mycube is 343.0

=== Code Execution Successful ===
```

Conclusion:

Function and constructor overloading are powerful features in Java that promote code efficiency and flexibility. By allowing the same method or constructor name to handle different parameter configurations, Java enables developers to write cleaner, more intuitive code.