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:

METHOD OVERLOADING
code:

```
class MethodOverloading
{
public static void main(String args[])
{
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
}
}

class Adder
{
static int add(int a, int b)
{
return a+b;
}
static double add(double a, double b)
{
```

```
return a+b;
}
}
```

OUTPUT:

```
C:\Users\Sharvari A Bhondekar\OneDrive\Desktop\JAVA PROGRAMS\Exp 4>javac MethodOverloading.java

C:\Users\Sharvari A Bhondekar\OneDrive\Desktop\JAVA PROGRAMS\Exp 4>java MethodOverloading.java
22
24.9
```

CONSTRUCTOR OVERLOADING

code:

```
class ConstructorOverload
{
public static void main(String[] args)
{ System.out.println(Multiplier.mult(2,3));
System.out.println(Multiplier.mult(2,2,2));
}
}
class Multiplier
{
static int mult(int a,int b)
{
return a*b;
}
static int mult(int a,int b,int c)
{
return a*b*c;
}
}
```

OUTPUT:

```
C:\Users\Sharvari A Bhondekar\OneDrive\Desktop\JAVA PROGRAMS\Exp 4>java ConstructorOverload.java
6
8
```

Conclusion:

Comment on how method and constructor overloading used using java

In Java, **method overloading** allows defining multiple methods within the same class that share the same name but differ in their parameter lists (by type, number, or order). This enables a method to handle different types or numbers of arguments, providing flexibility while maintaining the method's core functionality.

Constructor overloading works similarly, where a class can have multiple constructors with the same name (class name) but different parameters. This enables objects to be instantiated in various ways, depending on the arguments provided during object creation. Both are examples of compile-time polymorphism.