



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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);  
    }  
}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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 :

```
class OverloadTest
{
    public static void main(String args[])
    {
        OverloadTest o=new OverloadTest();
        o.add(1,4);
        o.add(1.00,2.06);
        o.add(10,12,14);
    }

    void add(int a, int b)
    {
        System.out.println("Sum of a and b is "+(a+b));
    }

    void add(double c, double d)
    {
        System.out.println("Sum of 1.00 and 2.06 is "+(c+d));
    }
}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
}  
  
void add(int e, int f, int g)  
{  
    System.out.println("Sum of e,f,g is "+(e+f+g));  
}  
}
```

Output:

```
C:\Users\User.DESKTOP-VK0H6B7\Documents\Java Projects>javac OverloadTest.java  
  
C:\Users\User.DESKTOP-VK0H6B7\Documents\Java Projects>java OverloadTest.java  
Sum of a and b is 5  
Sum of 1.00 and 2.06 is 3.06  
Sum of e,f,g is 36
```

Constructor Overloading:

```
public class Student  
{  
    //instance variables of the class  
    int id;  
    String name;  
  
    Student(){  
        System.out.println("this a default constructor");  
    }  
  
    Student(int i, String n){  
        id = i;  
        name = n;  
    }  
  
    public static void main(String[] args) {  
        //object creation  
        Student s = new Student();  
        System.out.println("\nDefault Constructor values: \n");  
        System.out.println("Student Id : "+s.id + "\nStudent Name : "+s.name);  
        System.out.println("\nParameterized Constructor values: \n");  
        Student student = new Student(10, "David");  
        System.out.println("Student Id : "+student.id + "\nStudent Name :  
        "+student.name);  
    }  
}
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Output:

```
C:\Users\User.DESKTOP-VK0H6B7\Documents\Java Projects>javac Student.java
```

```
C:\Users\User.DESKTOP-VK0H6B7\Documents\Java Projects>java Student.java  
this a default constructor
```

Default Constructor values:

Student Id : 0

Student Name : null

Parameterized Constructor values:

Student Id : 10

Student Name : David

Conclusion:

Comment on how function and constructor overloading used using java

- Function and constructor overloading are two techniques that allow us to define multiple methods or constructors with the same name but different parameters. They are useful for creating more flexible and reusable code in Java.
- Function overloading is the ability to create multiple methods with the same name but different parameter types, numbers, or orders. For example, we can have a method called add that can take two integers, two doubles, or three integers as arguments and perform the appropriate addition operation. Function overloading allows us to use the same method name for different types of operations and avoid creating multiple methods with different names
- Constructor overloading is the ability to create multiple constructors with the same name but different parameter lists. For example, we can have a class called Student that can have different constructors for initializing the student's id, name, contact number, college name, etc. Constructor overloading allows us to create objects of the same class with different initial values and avoid creating multiple classes with different constructors