



**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);
    }
}
```

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:**

### **1) 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(int c, double d)

{

System.out.println("Sum of 1.00 and 2.06 is "+(c+d));

}

void add(int e, int f, int g)

{

System.out.println("Sum of e,f,g is "+(e+f+g));

}

}
```

```
Microsoft Windows [Version 10.0.22000.1936]
(c) Microsoft Corporation. All rights reserved.

C:\Users\parik>cd C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02

C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02>javac OverloadTest.java

C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02>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

C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02>
```

## 2) Constructor Overloading



```
public class ConstructorOverloadingExample {  
  
    private int value;  
  
    // Default constructor  
  
    public ConstructorOverloadingExample() {  
  
        this.value = 0;  
  
    }  
  
    // Constructor with one parameter  
  
    public ConstructorOverloadingExample(int value) {  
  
        this.value = value;  
  
    }  
  
    // Constructor with two parameters  
  
    public ConstructorOverloadingExample(int value1, int value2) {  
  
        this.value = value1 + value2;  
  
    }  
  
    // Getter method  
  
    public int getValue() {  
  
        return value;  
  
    }  
  
    public static void main(String[] args) {  
  
        ConstructorOverloadingExample obj1 = new ConstructorOverloadingExample();
```



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

```
ConstructorOverloadingExample obj2 = new ConstructorOverloadingExample(5);

ConstructorOverloadingExample obj3 = new ConstructorOverloadingExample(3, 7);


System.out.println("Value of obj1: " + obj1.getValue());

System.out.println("Value of obj2: " + obj2.getValue());

System.out.println("Value of obj3: " + obj3.getValue());

}

}
```

```
Microsoft Windows [Version 10.0.22000.1936]
(c) Microsoft Corporation. All rights reserved.

C:\Users\parik>cd C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02

C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02>javac ConstructorOverloadingExample.java

C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02>java ConstructorOverloadingExample.java
Value of obj1: 0
Value of obj2: 5
Value of obj3: 10

C:\Users\parik\OneDrive\Desktop\Priyanka Bhandari 02>_
```

### Conclusion:

Comment on how function and constructor overloading used using java  
Function and constructor overloading in Java involves creating multiple methods or constructors with the same name within a class but with different parameter lists.

### Function Overloading:

Function overloading allows you to define multiple methods in a class with the same name



but different parameter lists (number or types of parameters). This way, you can provide different behavior for the same operation depending on the inputs. The choice of which method to call is made at compile-time based on the provided arguments.

### Constructor Overloading:

Constructor overloading is similar to function overloading but is used to define multiple constructors within a class with different parameter lists. Constructor overloading enables the creation of objects in various ways, depending on the arguments provided during object instantiation. Like function overloading, constructors can have different parameter types, but the number and types of parameters should differ to distinguish between them.