Roll No.: B050
Date :

Experiment 3

Aim:
To define a class having overloaded constructors and instantiating objects of the same class.

Theory:
In Java, a constructor is a block of codes that is similar to the method. It can be used to set initial values for object attributes. It is called when an instance of the class is created. At the time of calling the constructor, memory for the object is allocated in the memory. Every time an object is created using the new() keyword, at least one constructor is called. Constructors must have the same name as the class within which it is defined and they are called only once at the time of Object Creation. Access modifiers can be used in constructor declaration to control its access. There are two types of constructors in Java:
(1) Default Constructor - A constructor having no parameters.
(2) Parameterized constructors - A constructor having parameters.

Types of constructors:-

(1) Default Constructor-

A constructor having no parameters and is invisible. And if we write a constructor with no arguments, the compiler does not create a default constructor. If a class has no constructors, the Java compiler automatically provides a default constructor. This is known as implicit default constructor.

You can also define a default constructor explicitly. The automatic provision of a default constructor simplifies the creation of objects, especially in simple classes or in classes where no specific initialization logic is required.

(2) Parameterized constructor-

It is a constructor that takes one or more parameters, allowing the initialization of an object with specific values. By defining a parameterized constructor, you can ensure that the objects of your class are created in a consistent and predictable state. The syntax involves specifying the parameters within the parenthesis following the constructor name, and then using these parameters within the constructor body to assign values to the instance variables. Additionally, they can be overloaded, means a class can have multiple parameterized constructors.

Syntax: <class_name>() { }

```
public class nameoffile {
 public static void main (String[] args) {
   constructor_name obj1 = new constructor_name
                          (parameter1, parameter2);
   obj1.DisplayInfo();
   .
   .
   .
   .
   }
}
class constructor_name {
    private datatype parameter1;
    private datatype parameter2;
    public constructor_name(datatype parameter1,
                            datatype parameter2)
     {
    this.parameter1 = parameter1;
    this.parameter2 = parameter2;
    }
    :
    .
    public void DisplayInfo() {
    System.out.println(parameter1);
    System.out.println(parameter2);
    }
}
```

Values

Function name.

Example of constructor:

```
public class constructor {
public static void main (String [] args) {
    Geeks geek = new Geeks ();
  }
}

class Geeks {
   System.out.println (" constructor called ");
}
```

Constructor overloading-
Constructor overloading in Java allows developers to define multiple constructors within a class, each within a distinct set of parameters. While a class can have multiple constructors, each with different parameters, the concept of constructor overloading is used. This feature provides flexibility in object initialization, enabling objects to be created in various states depending on the arguments passed during instantiation. When a class has multiple constructors, each constructor must have unique ~~m~~ parameter list. The java compiler differentiates these constructors based on their signatures, which includes the number and type of parameters in the parameter list.

```
Car model: Camry
Car brand: Toyota
Car year: 2022
Car model: Camry
Car brand: Toyota
Car year: Car year unknown
Car model: unknown
Car brand: unknown
Car year: Car year unknown
```

Experiment 3.1

Write a program in Java to create a class car with 3 constructors overloaded in it. Also show how the objects are created using each constructor.

```
public class constructor {
  public static void main (String [] args) {
    car car1 = new car ("camry", "toyota", 2022);
    car1. DisplayInfo();
    car car2 = new car ("camry", "toyota");
    Car2. DisplayInfo();
    car car3 = new Car();
    Car3. DisplayInfo();
    }
}

class car {
    private String model;
    private String brand;
    private int year;
  public car (String model, String brand, int
                year) {
     this. model = model;
     this. brand = brand;
     this. year = year;
    }
```

```
public car (String model, String brand) {
    this.model = model;
    this.brand = brand;
    this.year = -1;
}
public car() {
    this.model = "unknown";
    this.brand = "unknown";
    this.year = -1;
}

public void DisplayInfo() {
    System.out.println("Car model: " + model);
    System.out.println("Car brand: " + brand);
    if (year != -1) {
        system.out.println("Car year: " + year);
    }
    else {
        system.out.println("Car year unknown");
    }
    }
}
```

Conclusion:

We have demonstrated constructor overloadi in Java by defining a class with multiple constructors and instantiating objects with different initial states.