Experiment No.4
Aim: Basic programming constructs like branching and looping
Name: Aman Mehtar
Roll no: 32
Date of Performance: 08/8/24
Date of Submission: 22/8/24

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:

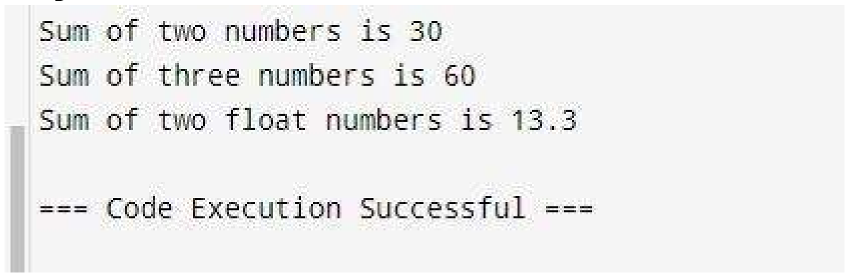
Thread t= new Thread (" MyThread ");

Code for method overloading:

```
import java.io.*;
import java.util.*;
class Calculator {
   int add(int a, int b) {
      return a + b;
   }

   int add(int a, int b, int c) {
      return a + b + c;
   }

   double add(double a, double b) {
      return a + b;
   }
}

public class Main {
   public static void main(String[] args) {
      Calculator calc = new Calculator();
```

```
        System.out.println("Sum of two numbers is "+calc.add(10, 20));
        System.out.println("Sum of three numbers is "+calc.add(10, 20, 30));
        System.out.println("Sum of two float numbers is "+calc.add(5.5, 7.8));
    }
}
```

Output:



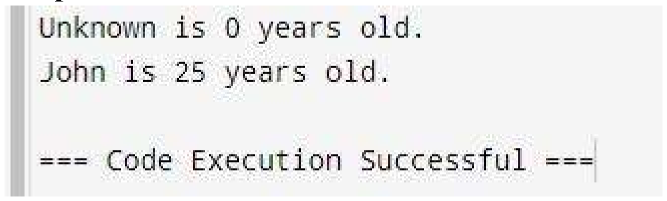
```
Sum of two numbers is 30
Sum of three numbers is 60
Sum of two float numbers is 13.3

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

code for constructor overloading:

```
import java.io.*;
import java.util.*;
class Person {
    String name;
    int age;

    Person() {
        name = "Unknown";
        age = 0;
    }

    Person(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println(name + " is " + age + " years old.");
    }
}

public class Main {
    public static void main(String[] args) {
        Person person1 = new Person();
        Person person2 = new Person("John", 25);

        person1.display();
        person2.display();
    }
```

}

Output:



```
Unknown is 0 years old.
John is 25 years old.

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

Conclusion:

Function and constructor overloading are powerful features in Java that enhance code usability and flexibility. Function overloading allows methods to handle varying input parameters, improving readability and maintainability. Constructor overloading enables multiple ways to initialize objects, providing clarity and flexibility in object creation.

Together, these overloading techniques support the principles of object-oriented programming by promoting code reusability and clarity, making Java a robust language for developing complex applications.