class OverloadingDemo {

// Constructor overloading

public OverloadingDemo() {

System.out.println("Default constructor called");

}

public OverloadingDemo(int num) {

System.out.println("Parameterized constructor with int called: " + num);

}

public OverloadingDemo(double num) {

System.out.println("Parameterized constructor with double called: " + num);

}

// Method overloading

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

public String add(String str1, String str2) {

return str1 + str2;

}

}

public class OverloadingExample {

public static void main(String[] args) {

OverloadingDemo constructorDemo1 = new OverloadingDemo();

OverloadingDemo constructorDemo2 = new OverloadingDemo(42);

OverloadingDemo constructorDemo3 = new OverloadingDemo(3.14);

System.out.println("Sum of integers: " + constructorDemo1.add(5, 7));

System.out.println("Sum of doubles: " + constructorDemo1.add(3.2, 4.8));

System.out.println("Concatenated strings: " + constructorDemo1.add("Hello, ", "World!"));

}

}

In this program, we have a **OverloadingDemo** class that demonstrates both constructor overloading and method overloading.

Constructor Overloading:

* We have three constructors in the **OverloadingDemo** class, with different parameter types: default, int, and double.
* When objects are created with these constructors, the appropriate constructor is called.

Method Overloading:

* We have overloaded the **add** method with different parameter types: int, double, and String.
* Depending on the argument types, the appropriate version of the **add** method is called.

The program demonstrates the use of constructor and method overloading by creating objects and calling the methods with different arguments.