Practical No.2

Aim: Write a Java program to method and constructor

overloading

Resources Required:

- · Java Development Kit (JDK)
- · Text Editor (e.g., Notepad++) or IDE (e.g., Eclipse, IntelliJ IDEA, NetBeans)
- · Command-line terminal or Java compiler

Theory:

Java is a high-level, object-oriented, platform-independent programming language. It provides many powerful features that make it widely used in software development. Encapsulation, Inheritance, Polymorphism and abstraction are main fetures of Java.

Polymorphism means "many forms." It allows the same function name to behave differently based on context (i.e., number or type of arguments, object type, etc.).

Types of Polymorphism in Java

1. **Compile-Time Polymorphism** (Static Binding)

Achieved using:

- **Method Overloading**
- **Constructor Overloading**

2. **Run-Time Polymorphism** (Dynamic Binding)

Achieved using:

- **Method Overriding** (with inheritance)

Method Overloading (Compile-Time Polymorphism) having multiple methods with the same name but different parameter lists in the same class is called method overloading.

Syntax:

```
class ClassName {
   void display() { }
   void display(int a) { }
   void display(String s) { }
}
```

Constructor Overloading

Constructor Overloading (Compile-Time Polymorphism) Having multiple constructors in the same class with different parameter lists is called constructor overloading.

Syntax:

```
class ClassName {

   ClassName() { }

   ClassName(int a) { }

   ClassName(String s) { }

}
```

Source Code:

1. Constructor Overloading:

```
class Student {
    Student() {
        System.out.println("Default Constructor");
    }
    Student(String name) {
        System.out.println("Name: " + name);
    }
    Student(int age, String course) {
        System.out.println("Age: " + age + ", Course: " + course);
    }
    public static void main(String[] args) {
        Student s1 = new Student();
        Student s2 = new Student("Gauri");
        Student s3 = new Student(20, "Java");
    }
}
```

Output:

2. Method Overloading:

```
class Calculator {
    void add(int a, int b) {
        System.out.println("Sum: " + (a + b));
    }
    void add(double a, double b) {
        System.out.println("Sum: " + (a + b));
    }
```

```
    void add(String a, String b) {
        System.out.println("Concatenation: " + (a + b));
    }
    public static void main(String[] args) {
        Calculator c = new Calculator();
        c.add(5, 10);           // Calls int version
        c.add(2.5, 3.5);        // Calls double version
        c.add("Hello", "Java"); // Calls String version
    }
}
```

Conclusion:

This Java program successfully demonstrates the implementation of constructor and method overloading.