METHOD OVERLAODING

In Java, method overloading allows you to define multiple methods with the same name but with different parameters within the same class. This enables you to perform similar operations on different types of data or with different numbers of arguments. The compiler determines which overloaded method to call based on the arguments provided during the method invocation.

Here's an example program that demonstrates method overloading in Java:

```
public class MathUtils {
    public int add(int a, int b) {
        return a + b;
    }

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

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

    public String add(String a, String b) {
        return a + b;
    }

    public static void main(String[] args) {
        MathUtils math = new MathUtils();

        int sum1 = math.add(2, 3);
        double sum2 = math.add(2.5, 3.7);
        int sum3 = math.add(2, 3, 4);
        String concat = math.add("Hello", "World");
```

```
        System.out.println("Sum 1: " + sum1);          // Output: Sum 1: 5
        System.out.println("Sum 2: " + sum2);          // Output: Sum 2: 6.2
        System.out.println("Sum 3: " + sum3);          // Output: Sum 3: 9
        System.out.println("Concatenation: " + concat);// Output: Concatenation: HelloWorld
    }
}
```

In this example, the MathUtils class defines multiple add methods with the same name but different parameters. The methods are overloaded based on the number and type of arguments.

- The add method with two int parameters performs integer addition.
- The add method with two double parameters performs floating-point addition.
- The add method with three int parameters performs integer addition with three numbers.
- The add method with two String parameters concatenates the strings.

When invoking the add method, the compiler determines the appropriate method based on the arguments provided. The return values of each overloaded method are then stored in separate variables and printed to the console.

Note that method overloading is determined at compile-time, based on the arguments' static types.