06-Method Overloading

Method Overloading in Java

Definition:

Method Overloading in Java allows multiple methods to share the same name but differ in their signatures. A method's signature can vary by:

- The number of input parameters
- The type of input parameters
- A combination of both the number and type of input parameters

This concept is also known as **Compile-time Polymorphism**, **Static Polymorphism**, or **Early Binding**. When a method is overloaded, Java gives priority to the method that matches the most specific signature during compilation.

Example:

Let’s consider a class Calculator with a method that adds two numbers and returns the result as an integer:

```java
class Calculator {
    public int add(int n1, int n2) {
        return n1 + n2;
    }

    public static void main(String[] args) {
        Calculator obj = new Calculator();
        int result = obj.add(2, 5);
        System.out.println(result); // Outputs: 7
    }
}
```

Suppose the requirements change, and you now need to add three numbers instead of two. Without method overloading, you would need to modify the existing method or create a new one with a different name. However, if requirements frequently change, this approach becomes cumbersome.

With method overloading, you can create multiple methods with the same name but different parameters:

```java
class Calculator {
    public int add(int n1, int n2) {
        return n1 + n2;
    }

    public int add(int n1, int n2, int n3) {
        return n1 + n2 + n3;
```

TELUSKO

```
    }

    public double add(double n1, int n2) {
        return n1 + n2;
    }
}
```

Different Ways of Method Overloading in Java:

1. **Changing the Number of Parameters:**
   - Method overloading can be achieved by altering the number of parameters passed to different methods.
   - Example:

```
public int add(int n1, int n2) {
    return n1 + n2;
}

public int add(int n1, int n2, int n3) {
    return n1 + n2 + n3;
}
```

2. **Changing the Data Types of the Arguments:**
   - Methods can be overloaded if they have the same name but different parameter types.
   - Example:

```
public int add(int n1, int n2) {
    return n1 + n2;
}

public double add(double n1, double n2) {
    return n1 + n2;
}
```

3. **Changing the Order of the Parameters:**
   - Method overloading can also be implemented by rearranging the order of parameters in two or more methods with the same name.
   - Example:

```
public void display(String name, int rollNo) {
    System.out.println("Name: " + name + ", Roll No: " +
rollNo);
}

public void display(int rollNo, String name) {
    System.out.println("Roll No: " + rollNo + ", Name: " +
name);
}
```

Advantages of Method Overloading:

- **Improved Readability:** Overloaded methods allow the use of the same method name for similar tasks, making the code easier to understand.
- **Code Reusability:** The same method name can be used for different operations, reducing redundancy in code.
- **Reduced Complexity:** Overloading allows related methods to be grouped together, simplifying the program structure.
- **Efficiency:** Programmers can perform tasks more effectively by using method overloading to handle different types or numbers of arguments with minimal code changes.
- **Flexible Object Initialization:** Objects of a class can be initialized in different ways using overloaded constructors.