



# METHODS IN JAVA

## 1. Introduction

In Java, a **method** is a **block of code** that performs a specific task and can be reused multiple times in a program.

It helps in **modular programming**, **code reusability**, and **better organization** of code

Methods are used to perform certain actions, and they are also known as **functions**.





### Why use methods?

To reuse code: define the code once, and use it many times.

## 2. Definition

A method in Java is a collection of statements that are grouped together to perform an operation.

## 3. Importance of Methods

-  Increases **code reusability**
-  Makes the program **modular and readable**
-  Helps in **debugging and maintenance**
-  Supports **object-oriented design**

## 4. Syntax of Method

```
returnType methodName(parameter1, parameter2, ...) {  
    // method body  
    // statements  
}
```

```
    return value; // optional
}
```

**Example:**

```
int addNumbers(int a, int b) {
    int sum = a + b;
    return sum;
}
```

## 5. Types of Methods

Java methods can be categorized as:

### (A) Built-in Methods

→ Already defined in Java classes (like `System.out.println()` , `Math.sqrt()` , etc.)

### (B) User-defined Methods

→ Defined by programmers for specific tasks.

Further divided into:

- **Without Parameters and Without Return Type**
- **With Parameters but Without Return Type**
- **Without Parameters but With Return Type**
- **With Parameters and With Return Type**

## 6. Method Declaration and Calling

A Method can be called by creating an object of the class in which the method exists followed by the method call

**Example:**

```
public class Demo {
    // Method declaration
    void greet() {
        System.out.println("Hello, Java!");
    }

    public static void main(String[] args) {
        Demo obj = new Demo(); // create object
        obj.greet();           // method call
    }
}
```

### Output:

```
Hello, Java!
```

## 7. Method Parameters , Void, Return Type & Static Keywords

- **Parameters** → Data passed to a method.
- **Return Type** → Type of value returned by a method.
- Void return types → When we don't want our method to return anything we use void as the return types.
- Static Keyword → Static keyword is used to associate a methods of a given class with the class rather than the object . Static method in a class is shared by all the objects.

### Example:

```
int multiply(int x, int y) {  
    return x * y;  
}
```

## 8. Method Overloading

When two or more methods have the same name but different parameters, it's called Method Overloading.

### Example:

```
class Calculator {  
    int add(int a, int b) {  
        return a + b;  
    }  
    double add(double a, double b) {  
        return a + b;  
    }  
}
```

Methods Overloading cannot be performed by changing the return type of methods.

### Key Points:

- Same method name
- Different number/type/order of parameters
- Done within the **same class**

## 9. Static vs Non-static Methods

Feature	Static Method	Non-static Method
Belongs to	Class	Object
Called using	Class name	Object
Memory	Shared among all objects	Separate for each object
Example	<code>Math.sqrt(9)</code>	<code>obj.display()</code>

## 10. Variable Arguments (Varargs)

A functions with varargs can be created in java using the following syntax.

```
public static void fun(int ... a)
{

    // method body
}
```

Example -

```
public class VarargsExample {
public static void printNumbers(int... numbers) {
    System.out.println("Received " + numbers.length + " numbers:");
    for (int number : numbers) {
        System.out.println(number);
    }
}

public static void main(String[] args) {
    printNumbers(1); // Calling with single arguments
    printNumbers(1,2); // Calling with a double argument
    printNumbers(1,2,3); // Calling with multiple arguments
}
}
```

Output -

```
1
1 2
1 2 3
```

## 11. Recursion

A Functions in a Java can call itself . Such calling of function by itself is called recursion.

**Example:**

```
int factorial(int n) {  
    if (n == 1)  
        return 1;  
    else  
        return n * factorial(n - 1);  
}
```

## 12. Advantages of Using Methods

- Code reusability
- Easy to modify
- Reduces redundancy
- Simplifies complex problems
- Better testing and maintenance

## 13. Real-Life Example

```
class Bank {  
    void deposit(int amount) {  
        System.out.println(amount + " deposited successfully!");  
    }  
    void withdraw(int amount) {  
        System.out.println(amount + " withdrawn successfully!");  
    }  
  
    public static void main(String[] args) {  
        Bank user = new Bank();  
        user.deposit(1000);  
        user.withdraw(500);  
    }  
}
```