Java Learning Journey

Chapter 6 – Methods in Java

🖒 Prepared by: Ahmed Elsifi

Introduction to Methods

Methods (also called functions in some languages) are used to:

- Avoid code repetition
- Modularize code (break into smaller, manageable parts)
- Improve readability, debugging, and maintenance
- Promote reusability across classes and projects

Functions vs. Loops

- Loops repeat a block of code within the same method.
- Methods allow you to reuse a block of code from anywhere in the program.
- Methods are better because they support:
 - o Code reuse
 - Better organization
 - Easier testing and debugging
 - Team collaboration

Method Structure

Method Signature:

```
modifier returnType methodName(parameterType parameterName, ...)
```

Example:

```
public static int max(int num1, int num2)
```

Parts of a Method:

- Modifier: e.g., public, static
- Return type: e.g., int, double, void
- Method name: e.g., max
- Parameters: e.g., (int num1, int num2)

Parameters vs. Arguments

- Parameter: Variable defined in the method header.
- **Argument**: Actual value passed to the method when called.

Example:

```
// Definition
public static void greet(String name) { ... } // 'name' is a parameter

// Invocation
greet("Alice"); // "Alice" is an argument
```

Return Types

- void: No return value.
- Primitive types: int, double, char, boolean, etc.
- Reference types: String, arrays, objects.

Use void when the method performs an action without returning a value.

Use a return type when the method computes and returns a result.

Reusing Methods from Other Classes

Use ClassName.methodName() to call a static method from another class.

Example:

```
// In AnotherClass.java
public static void printHello() {
    System.out.println("Hello");
}

// In MyClass.java
AnotherClass.printHello();
```

Typical Java Project Structure

- .java files: Source code
- .class files: Compiled bytecode
- Package: Folder organizing related classes

Static vs. Instance Methods

- Static methods: Belong to the class. Called via ClassName.method().
- Instance methods: Belong to an object. Called via objectName.method().

Use static for utility methods that don't depend on instance variables.

The Function Declaration Order

In Java, you can define methods both before and after main.

The Java compiler processes the entire file before linking, unlike C/C++ which relies on order or headers.

Variable-Length Arguments (Varargs)

Java supports varargs using ellipsis (...):

```
public static void printAll(String... values) {
    for (String s : values) {
        System.out.println(s);
    }
}

// Call with any number of arguments
printAll("A", "B", "C");
```


- Java is always pass-by-value.
- For primitives: the value is copied.
- For objects: the reference (address) is copied, so changes to the object's state are visible, but reassigning the reference is not.

Example:

Refactor > Rename Method

- Available in **NetBeans**, **IntelliJ**, and **VS Code** (with Java extensions).
- Renaming a method automatically updates all its references.
- Helps maintain consistency and avoid errors.

Modularizing Code

- Why: Easier to read, test, reuse, and maintain.
- When: When a block of code is used multiple times or performs a distinct task.

Method Overloading

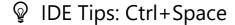
Define multiple methods with the same name but different parameter lists.

Example:

```
public static int max(int a, int b) { ... }
public static double max(double a, double b) { ... }
```

✓ Good Use:

- Same operation for different types
- Different number of parameters



- Press Ctrl+Space in NetBeans, IntelliJ, or VS Code to:
 - See method suggestions
 - Auto-complete code
 - View method signatures

Method Abstraction

- Hide implementation details from the user.
- User only needs to know what the method does, not how.

Example: You use System.out.println() without knowing how it works.

Stepwise Refinement (Divide and Conquer)

- Break a large problem into smaller subproblems.
- Implement each subproblem as a method.
- Use **top-down** (stubs) or **bottom-up** (drivers) approach.

Example: A calendar printing program can be broken into:

- printMonth
- getStartDay
- isLeapYear
- etc.

Design Diagrams & Flowcharts

Useful Tools:

- Lucidchart (free tier)
- Draw.io (free)
- Miro (free for basic use)
- Visual Paradigm (free community edition)

Use these to create:

- Structure charts
- Flowcharts
- UML diagrams

✓ Key Takeaways

- Methods improve reusability, readability, and maintainability.
- Use **method overloading** for similar operations with different inputs.
- Always use **method abstraction** to hide complexity.
- Stepwise refinement helps manage large projects.
- Java uses **pass-by-value** for all parameters.
- Use **static** for utility methods; **instance methods** for object-specific behavior.