Methods

COP2250: Java Programming

Kevin Pyatt, Ph.D.

State College of Florida
Pyatt Labs

Week 7

Today's Objectives

- Define and call methods
- Understand parameters and return values
- Distinguish `void` methods from value-returning methods
- Use method overloading
- Understand variable scope
- Apply the divide-and-conquer approach to problem solving

Why Methods?

Without methods:

- Code duplication everywhere
- Hard to read, hard to debug
- Changes require editing multiple places

With methods:

- Write once, call many times
- Each method does **one thing**
- Easier to test and debug
- Code reads like a story

Think of methods as tools in a toolbox — you build them once and reach for them when needed.

Anatomy of a Method

```
public static int add(int a, int b) {
    int sum = a + b;
    return sum;
}
```

`public static`	Modifiers (we'll explore these later)
`int`	Return type
`add`	Method name
`(int a, int b)`	Parameters
`return sum;`	Return statement

Rule: Return type must match what you actually return.

Calling a Method

```java
public static void main(String[] args) {
    int result = add(5, 3);
    System.out.println(result); // 8

    // Can also use directly
    System.out.println(add(10, 20)); // 30
}

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

Key points:

- Arguments are **copied** into parameters (pass by value)
- Method can be defined before or after `main`
- Return value can be stored or used directly

Void Methods

```
public static void printGrade(int score) {
    if (score >= 90)
        System.out.println("A");
    else if (score >= 80)
        System.out.println("B");
    else
        System.out.println("Below B");
}
```

- `void` = this method does NOT return a value
- It **does something** (prints, modifies, etc.)
- You **cannot** assign a void method to a variable
- `return;` can be used to exit early (optional)

`printGrade(85);` ✓

`int x = printGrade(85);` × (compiler error)

Parameters vs. Arguments

```java
// "a" and "b" are PARAMETERS (placeholders)
public static int max(int a, int b) {
    return (a > b) ? a : b;
}

// 5 and 3 are ARGUMENTS (actual values)
int biggest = max(5, 3);
```

Parameter	Variable in the method signature
Argument	Value passed when calling the method

Arguments must match parameters in order, number, and compatible type.

Method Overloading

```java
public static int max(int a, int b) {
    return (a > b) ? a : b;
}

public static double max(double a, double b) {
    return (a > b) ? a : b;
}

public static int max(int a, int b, int c) {
    return max(max(a, b), c);
}
```

Same name, different parameter lists.
Java picks the right version based on arguments.

Not overloading: changing only the return type.

Variable Scope

```
public static void main(String[] args) {
    int x = 10;
    doSomething();
    System.out.println(x); // still 10
}

public static void doSomething() {
    int x = 99; // different x!
    System.out.println(x); // 99
}
```

- Variables declared inside a method are **local** to that method
- Each method has its own **scope**
- Same variable name in different methods = different variables
- Parameters are also local variables

Method Design Principles

1. **One job per method** — if you need "and" to describe it, split it
2. **Descriptive names** — `calculateTax()` not `doStuff()`
3. **Keep methods short** — if it scrolls, break it up
4. **Minimize parameters** — 3 or fewer is ideal
5. **Return don't print** — let the caller decide what to do with the result

"A method should do one thing, do it well, and do it only." — *Clean Code*

Common Mistakes

1. Forgetting to return:

```
public static int square(int n) {
    int result = n * n;
    // missing return result;
}
```

2. Ignoring the return value:

```
add(5, 3); // result goes nowhere
```

3. Wrong argument types/order:

```
// Method expects (String, int)
printInfo(25, "Alice"); // WRONG ORDER
```

4. Trying to access local variables from another method

Assignment 6: Sort Three Numbers

Exercise 6.5: Write a method:

- `displaySortedNumbers(double, double, double)`
- Takes three numbers, displays them in increasing order
- `main` prompts user, calls the method

Lab: MethodPractice

- Build utility methods step by step
- Practice `void` and value-returning methods
- Apply method design principles

Next Week: Single-Dimensional Arrays (Chapter 7)