Methods

COP2250: Java Programming

Kevin Pyatt, Ph.D.

State College of Florida
Pyatt Labs

Week 7

Today's Objectives

- Understand what methods are and why we use them
- Define methods with parameters and return values
- Distinguish `void` methods from value-returning methods
- Call methods and pass arguments
- Understand method signatures and overloading
- Apply the **divide and conquer** principle

The Problem: Copy-Paste Code

What if you need the same logic in 5 places?

```
// Find max of two numbers... again
int max1;
if (a > b) max1 = a; else max1 = b;

// Same logic, different variables
int max2;
if (x > y) max2 = x; else max2 = y;
```

Problems:

- Duplicate code everywhere
- Fix a bug? Fix it in 5 places
- Hard to read, hard to maintain

Methods let you write it once, call it anywhere.

Anatomy of a Method

```
public static int max(int num1, int num2) {
    if (num1 > num2) {
        return num1;
    } else {
        return num2;
    }
}
```

Part	Meaning
`public static`	Modifiers (we'll use these for now)
`int`	Return type
`max`	Method name
`(int num1, int num2)`	Parameters
`return num1;`	Return statement

`void` Methods

A method that does something but **returns nothing**.

```
public static void printGreeting(String name) {
    System.out.println("Hello, " + name + "!");
}
```

Calling it:

```
printGreeting("Kevin"); // prints: Hello, Kevin!
```

- Return type is `void` — no return value
- You **call** it as a statement, not in an expression
- Used for: printing, display, side effects

Value-Returning Methods

A method that computes and **sends back** a result.

```
public static double celsiusToFahrenheit(double c) {
    return (9.0 / 5) * c + 32;
}
```

Calling it:

```
double temp = celsiusToFahrenheit(100);
System.out.println(temp); // 212.0
```

- Return type matches what `return` sends back
- You **use** the result — assign it, print it, pass it
- Every path through the method must hit a `return`

Arguments vs Parameters

```java
// DEFINITION: num1 and num2 are PARAMETERS
public static int max(int num1, int num2) {
    return (num1 > num2) ? num1 : num2;
}

// CALL: 5 and 3 are ARGUMENTS
int result = max(5, 3);
```

- **Parameters**: variables in the method header (placeholders)
- **Arguments**: actual values you pass when calling
- Arguments must match parameters in **order, number, and compatible type**

Think of it like a function in math: $f(x) = x^2$
x is the parameter. $f(3) = 9$ — 3 is the argument.

Method Overloading

Same name, **different parameters**.

```
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);
}
```

- Java picks the right version based on arguments
- Overloading = same name, different **signature**
- Signature = method name + parameter types
- Return type alone does NOT distinguish methods

Variable Scope

Variables declared inside a method are **local** to that method.

```
public static void methodA() {
    int x = 10; // x lives here only
}

public static void methodB() {
    System.out.println(x); // ERROR: x not found
}
```

- Local variables are created when the method is called
- They are destroyed when the method returns
- Two methods can have variables with the same name — they are **different variables**
- Pass data between methods via **parameters and return values**

Common Mistakes

1. Forgetting `return`:

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

2. Calling void method in expression:

```
int x = printGreeting("Kevin"); // ERROR
```

3. Wrong argument types:

```
max("hello", "world"); // expects int, not String
```

4. Defining a method inside another method:

```
public static void main(String[] args) {
    public static void broken() { } // ERROR
}
```

Today's Lab + Walkthrough

Method Practice Lab

- Write a `void` method that prints a grade from a score
- Write a value-returning method that finds max of three numbers
- Write a method that checks if a number is even
- Call each method from `main` with user input

Assignment 6: Write `displaySortedNumbers(double, double, double)` — sort and display three numbers in increasing order.

Last 15 minutes: Code Walkthrough

- Screen-share your code
- Walk me through it line by line
- Explain *why*, not just *what*
- No working code = no credit