Lesson 3: Arrays and Methods

Marcin Kurzyna

Lecture Goals

This lecture introduces two important Java concepts: **arrays** for storing multiple values, and **methods** for organizing code into reusable blocks. By the end of this lecture, students should be able to:

- Declare and use arrays.
- Write and call methods with parameters and return values.
- Understand how data and behavior are organized in Java programs.

1 Arrays

An array is a collection of elements of the same type stored in a single variable.

1.1 Declaring and Initializing Arrays

You can declare an array like this:

```
int[] numbers;
numbers = new int[5];
```

Or both at once:

```
int[] numbers = new int[5];
```

This creates an array that can store 5 integers (indices 0–4).

1.2 Assigning and Accessing Elements

Use the index (starting at 0) to assign or read values:

```
numbers[0] = 10;
numbers[1] = 20;
System.out.println(numbers[0]); // prints 10
```

1.3 Array Initialization with Values

You can also initialize directly with values:

```
int[] primes = {2, 3, 5, 7, 11};
```

1.4 Looping Through Arrays

Often you'll use a loop to process array elements:

```
for (int i = 0; i < primes.length; i++) {
    System.out.println(primes[i]);
}
```

The property `length` gives the number of elements in the array.

1.5 Enhanced `for` Loop

Java also provides a shorter syntax:

```
for (int p : primes) {
    System.out.println(p);
}
```

This loop automatically iterates over all elements in the array.

2 Example: Average of Numbers

```
class AverageExample {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        int sum = 0;
        for (int n : numbers) {
            sum += n;
        }
        double average = (double) sum / numbers.length;
        System.out.println("Average = " + average);
    }
}
```

3 Methods

A method is a reusable block of code that performs a specific task.

3.1 Defining a Method

General form:

```
returnType methodName(parameters) {
    // method body
}
```

Example:

```
void greet() {
    System.out.println("Hello!");
}
```

3.2 Calling a Method

To execute the method:

```
greet();
```

3.3 Methods with Parameters

Methods can take inputs:

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

Call it like this:

```
greetUser("Alice");
```

3.4 Methods with Return Values

Some methods return results:

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

Usage:

```
int result = add(3, 5);
System.out.println(result);
```

3.5 Combining Methods and Loops

You can use methods to process arrays or perform calculations:

```
class SumArray {
    static int sum(int[] arr) {
        int total = 0;
        for (int n : arr) {
            total += n;
        }
        return total;
    }

    public static void main(String[] args) {
        int[] values = {1, 2, 3, 4, 5};
        int result = sum(values);
        System.out.println("Sum = " + result);
    }
}
```

Summary

This lecture introduced:

- Arrays: structures for storing multiple values of the same type.
- Methods: reusable code blocks that can take parameters and return results.
- Using loops with arrays for processing collections of data.

4 Exercises

1. Create an array of 5 integers and print all elements using a `for` loop.
2. Write a program that finds the largest number in an array.
3. Compute the sum and average of numbers in an array using a method.
4. Write a method `isEven(int n)` that returns `true` if a number is even.
5. Write a program that counts how many even numbers are in an array.
6. Create a method `reverseArray(int[] arr)` that prints the array elements in reverse order.
7. Write a method `max(int a, int b)` that returns the larger of two numbers.
8. Combine methods and arrays: Write a program that fills an array with numbers 1–10 and prints their squares using a separate method.
9. Challenge: Write a method that calculates the factorial of a number using recursion.
10. Challenge: Write a program with a method that checks whether a given word is a palindrome.