Single-Dimensional Arrays

COP2250: Java Programming

Kevin Pyatt, Ph.D.

State College of Florida
Pyatt Labs

Week 8

Today's Objectives

- Declare and create arrays
- Access elements with index notation
- Process arrays with loops
- Find max, min, sum, and average of array elements
- Understand array bounds and `ArrayIndexOutOfBoundsException`
- Use `length` property

The Problem: Too Many Variables

Store 100 student scores without arrays:

```
double score1 = 85.0;
double score2 = 92.0;
double score3 = 78.0;
// ... 97 more ...
```

With an array:

```
double[] scores = new double[100];
```

One variable, 100 values. Arrays let you store collections of the same type.

Declaring and Creating Arrays

Two steps:

```
// Declare and create (empty, default values)
double[] scores = new double[5];

// Declare and initialize with values
int[] nums = {10, 20, 30, 40, 50};
```

`double[]`	Type is “array of doubles”
`new double[5]`	Creates 5 slots, all initialized to 0.0
`{10, 20, 30}`	Shorthand — size determined by values

Default values: `int` = 0, `double` = 0.0, `boolean` = false, `String` = null

Accessing Elements

```
int[] nums = {10, 20, 30, 40, 50};

System.out.println(nums[0]); // 10 (first)
System.out.println(nums[4]); // 50 (last)
System.out.println(nums[5]); // CRASH!

nums[2] = 99; // Change third element
```

- Index starts at **0**, ends at **length - 1**
- `nums[5]` on a size-5 array → `ArrayIndexOutOfBoundsException`
- `nums.length` returns 5 (not an index!)

Mental model: 5 boxes numbered 0 through 4. No box 5.

Processing Arrays with Loops

Print all elements:

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

Sum all elements:

```
double total = 0;
for (int i = 0; i < scores.length; i++) {
    total += scores[i];
}
double average = total / scores.length;
```

Pattern: `for (int i = 0; i < array.length; i++)`
You will write this loop hundreds of times. Memorize it.

Finding the Maximum

```java
double max = scores[0]; // Start with first

for (int i = 1; i < scores.length; i++) {
    if (scores[i] > max) {
        max = scores[i];
    }
}

System.out.println("Best score: " + max);
```

Key idea:

- Assume the first element is the max
- Walk through the rest — if you find bigger, update
- Loop starts at `i = 1` (already checked index 0)

Same pattern works for min — just flip the comparison.

Enhanced For Loop (for-each)

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

// Enhanced for loop (same result)
for (double s : scores) {
    System.out.println(s);
}
```

Use enhanced for when:

- You need every element
- You do NOT need the index

Use standard for when:

- You need the index
- You need to modify elements
- You need to go backwards or skip elements

Reading Input into an Array

```
Scanner input = new Scanner(System.in);

System.out.print("How many scores? ");
int n = input.nextInt();

double[] scores = new double[n];

for (int i = 0; i < n; i++) {
    System.out.print("Score " + (i + 1) + ": ");
    scores[i] = input.nextDouble();
}
```

Pattern:

1. Ask how many
2. Create array of that size
3. Loop to fill it

Common Mistakes

1. Off-by-one:

```
for (int i = 0; i <= scores.length; i++) // CRASH
for (int i = 0; i < scores.length; i++) // CORRECT
```

2. Uninitialized array:

```
double[] scores;     // declared, not created
scores[0] = 85;      // NullPointerException
```

3. Confusing length with last index:

```
// Array of 5: indices 0-4, length is 5
scores[scores.length]   // CRASH (index 5)
scores[scores.length - 1] // last element
```

4. Wrong initial max:

```
double max = 0;        // WRONG if all scores negative
double max = scores[0];  // CORRECT
```

Assignment 7: Grade by Curve

Exercise 7.1: Write a program that:

- Reads student scores into an array
- Finds the best score
- Assigns grades on a curve:

Grade A	score $\geq$ best $-$ 10
Grade B	score $\geq$ best $-$ 20
Grade C	score $\geq$ best $-$ 30
Grade D	score $\geq$ best $-$ 40
Grade F	otherwise

Lab: ArrayPractice

- Build array utility methods step by step
- Sum, average, max, min, count, search

Next Week: Multi-Dimensional Arrays (Chapter 8)