Arrays

Week 12: Storing Multiple Values

The Problem: Too Many Variables

What if we need to store grades for 40 students?

int grade1, grade2, grade3, ... grade40;

What if we need to calculate the average?

average = (grade1 + grade2 + ... + grade40) / 40;

This is tedious, error-prone, and doesn't scale!

The Solution: Arrays

An array stores multiple values of the SAME type

Think of it like a row of mailboxes:

- Each mailbox holds one value
- Each mailbox has a number (index)
- All mailboxes hold the same type of mail

```
int[] grades = new int[40];

// Much easier to work with!
for (int i = 0; i < 40; i++) {
    sum += grades[i];
}
average = sum / 40;
```

Array Vocabulary

ELEMENT: one value in the array

INDEX: the position/location (starts at 0!)

LENGTH: how many elements total

DECLARATION: creating the array

Example: int[] scores = new int[5];

Creating Arrays: Two Ways

Method 1: Create empty, fill later

- Good when you don't know values yet
- Good when loading from user/file

Method 2: Create with values

- Good when you know all values
- Shorter, cleaner code

```
// Method 1: Declare size, fill later
int[] temps = new int[7];
temps[0] = 72;
temps[1] = 75;
// etc.

// Method 2: Declare AND fill
int[] temps = {72, 75, 68, 71};
```

Accessing Array Elements

Use brackets [ ] with the index

Reading: arrayName[index]

Writing: arrayName[index] = value;

Remember: indices start at 0!

- First element: [0]
- Last element: [length-1]

```
int[] scores = {85, 90, 78, 92, 88};

System.out.println(scores[0]); // 85
System.out.println(scores[2]); // 78
System.out.println(scores[4]); // 88

scores[1] = 95;  // Change 90 to 95
System.out.println(scores[1]); // 95
```

Examples #1

1. Create an int array called ages with 5 values:

   {18, 21, 19, 22, 20}

2. Print out the FIRST age

3. Print out the LAST age

4. Change the third age to 25

5. Print out the third age to verify

Array Length Property

Every array knows its own size!

Use .length (NO parentheses)

Why is this useful?

- Loops: for (int i = 0; i < arr.length; i++)
- Finding last element: arr[arr.length - 1]
- Avoiding errors

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

// .length tells you the size
System.out.println(numbers.length); // 5

// Last index is always length - 1
int lastIndex = numbers.length - 1;
System.out.println(numbers[lastIndex]); //
```

Arrays + Loops = Power

Loops make arrays useful!

Pattern for processing arrays:

```
for (int i = 0; i < array.length; i++)
{
   // do something with array[i]
}
```

The variable i is your INDEX

```
int[] scores = {85, 90, 78, 92, 88};

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

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

Examples #2

1. Create an int array:
   - At least 5 elements
   - Fill with any numbers
2. Use a loop to print ALL elements
3. Use a loop to calculate the SUM
4. Print the sum

Arrays as Parameters

You can pass arrays to methods!

Syntax: `(datatype[] parameterName, …)`

Example: `public static int arithmeticMethod(int[] scores) {`

Example: `public static void myExcellentMethod(String[] names) {`

```
public static void printArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        System.out.println(arr[i]);
    }
}

public static void main(String[] args) {
    int[] nums = {10, 20, 30};
    printArray(nums);  // Pass entire array
}
```

Methods Can Return Arrays

Return type: datatype[]

Example: public static int[] methodName()

Useful for:

- Creating new arrays
- Loading data from files
- Transforming arrays

```
public static int[] doubleValues(int[] arr) {
    int[] result = new int[arr.length];
    for (int i = 0; i < arr.length; i++) {
        result[i] = arr[i] * 2;
    }
    return result;
}

public static void main(String[] args) {
    int[] original = {1, 2, 3, 4, 5};
    int[] doubled = doubleValues(original);
}
```

Examples #3

Write TWO methods:

1. public static int sum(int[] numbers)
   - Takes an int array
   - Returns the sum of all elements

2. public static double average(int[] numbers)
   - Takes an int array
   - Calls sum() to get the total
   - Returns the average

Test both methods in main!

Common Array Mistakes

1. ArrayIndexOutOfBoundsException
   - Trying to access index that doesn't exist
   - arr[5] when arr.length is 5

2. Off-by-one errors
   - for (int i = 0; i <= arr.length; i++) ❌
   - for (int i = 0; i < arr.length; i++) ✓

3. Forgetting arrays start at 0
   - First element is [0], not [1]

Key Takeaways

✓ Arrays store multiple values of ONE type

✓ Index starts at 0, ends at length-1

✓ Use .length to get the size

✓ Loops are your friend for processing arrays

✓ Arrays can be parameters and return values

✓ Practice with Peer Learning and Lab activities!