Lecture 07

Single-Dimensional Arrays
Single-Dimensional Arrays

- Fixed-size, sequential collection of elements (of the same type)
- Can be used to store *anything* - of the same type
- way better than declaring num1, num2, ..., num500, ...
Declaring arrays

- Syntax:

```java
elementType[] arrayRefVar;
```

- Example:

```java
int[] xPositions;
```
Using Arrays

- **Note:** Simply declaring an array does not actually create any space for it. It’s a placeholder in memory, but value is still *null*.

- In order to use it, you still need to create the object.
Creating an array object

- Syntax:

  ```java
  arrayRefVar = new elementType[arraySize];
  ```

- Or combine with the declaration:

  ```java
  elementType[] arrayRefVar = new elementType[arraySize];
  ```

  ```java
  int[] xPositions = new int[3000];
  ```
Assigning values to elements in array

- **Syntax:**

  \[
  \text{arrayRefVar[index]} = \text{value};
  \]

- **Example:**

  ```
  xPositions[0] = 0;
  xPositions[1] = 10;
  xPositions[2] = 20;
  xPositions[3] = 30;
  xPositions[4] = 40;
  xPositions[5] = 50;
  ```
Array length

- Length is fixed and must be declared when created.
- Can’t get bigger or smaller
- length can be accessed by `arrayVar.length`
- **NOTE:** strings use `myString.length()` and arrays use `myArray.length`
- Default values are given when created. For numbers, it’s `0`, for booleans, its `false`, Strings are `null`
Accessing array items

• Index based, just like the strings we’ve seen

• Syntax:

xPositions[37]
Practice

• Declare an array of 10 integers and assign each element in the array the numbers 1 - 10 in order
public class ArrayPractice {

    public static void main(String[] args) {
        int[] myNumbers = new int[10];

        for (int i = 0; i < myNumbers.length; i++) {
            myNumbers[i] = i + 1;
        }

        for (int i = 0; i < myNumbers.length; i++) {
            System.out.println(myNumbers[i]);
        }
    }
}
Array initializer shortcut

If you know from the start what the values will be, you can initialize it all on one line with this shortcut.

```java
double[] myNumbers = {1.9, 2.9, 3.4, 3.5};
```

Note: There’s no ‘new’ keyword. Can’t break up this line.
Practice: Initializing from user input

- Declare an array and populate it with integers that users input from the console and print the numbers as follows:
  - Number 1 is 40
  - Number 2 is 2
  ...

import java.util.*;
public class ArrayPractice_prep {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int[] numbers = new int[10];

        System.out.print("Enter 10 ints, with spaces in between: ");
        for (int i = 0; i < numbers.length; i++){
            numbers[i] = input.nextInt();
            System.out.println("Number "+ (i+1) + " is " + numbers[i]);
        }
    }
}
Practice

- Write a method that takes an array of doubles as a parameter and returns the highest value from the array.
public static double highestValue(double[] myList) {
    double highest = myList[0];

    for (int i = 0; i < myList.length; i++) {
        if (myList[i] > highest) {
            highest = myList[i];
        }
    }

    return highest;
}
Foreach loop

- Useful when you just want to loop through the whole array

- Syntax:

```java
for (elementType element: arrayRefVar) {
    // Process the element
}
```

- Example:

```java
for (double e: myList) {
    System.out.println(e);
}
```
Indices

• Remember that arrays are zero-indexed, like characters in a String, so the first element is element 0

• So don’t try to access an element at length, only until length - 1
Shuffling an array

- We often need to shuffle arrays randomly, how can we do this?
Command-line arguments

• Now we know what that cryptic String[] args parameter of our main methods is!

• It’s an array of Strings

• We can pass Strings into our program from the command line when we run them.
public class CLArgs {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            System.out.println("Argument "+ i + ": "+ args[i]);
        }
    }
}
Command-line arguments

- What might these be useful for?
  - Working with files
  - ‘help’ option to know how to use the program
  - Running short, repeated commands without needing any other prompt
  - Etc...
To be continued...