Lecture 07: Arrays

SE115: Introduction to Programming I

Previously on SE 115…

- Previously on SE 115…
    - Previously on SE 115…
        - Previously on SE 115…
- Yes, we have talked about recursion last week.

Data, lots of data

- There comes a time when we need to store a lot of data.
- We can handle variables like n1, n2, n3, but after a while it gets annoying.
- The reason we call them n1 and n2 is because they are related. They are the number 1 and number 2 of *something*.
- In mathematics, we have a way of handling these “numbers” - we call them indices or indexes.

$$f(x) = x^2 + 3x - 10$$

There are two roots for this function; $x_1 = -5$ and $x_2 = 2$.

Data, lots of data

- It is obvious that we cannot create n number of variables for a program.
- For example, I want to find the average midterm grade (next week).
- I also want to find the minimum and maximum grades. So I have to store everyone’s grade in variables.
- I should **NOT** do the following:

```
int g1, g2, g3, g4, g5, g6, g7, g8, g9, g10, g11, g12, g13, g14, g15,
g16, g17, g18, g19, g20, g21, g22, g23, g24, g25, g26, g27, g28, g29,
g30, g31, g32, g33, g34, g35, g36, g37, g38, g39, g40, g41, g42, g43,
g44, g45, g46, g47, g48, g49, g50, g51, g52, g53;
```

Arrays

- So, let me introduce arrays.
- The word “array” means “organized”, and the first time you hear it, it might sound strange.
- In Turkish we call it a “series” (dizi), and it might help you better understand what it does.
- It is a “collection” of values of a **single** type.
- An array provides a means to access a specific element when we provide the “index” or “subscript” value.
- It uses brackets [ ] and you have been using them since day 1:

```
public static void main(String[] args) {...}
// the variable args is an array of String values.
```

Arrays

- When we write…

  ```
  int[] arr = new int[10];
  ```

- …we say that we want to create 10 integer values.
- The name of the array is **`arr`**.
- Its first element is **`arr[0]`**, last element is **`arr[9]`**.
- Notice that the index between the brackets is an **int**eger.
- Arrays are the simplest data structure, you will learn more about data structures next year in CE 221.
- Let's write some code to demonstrate what we can do with arrays.

Arrays

```
public class ArrayInit {
    public static void main(String[] args) {
        int[] arr = new int[10];
        for(int i=0;i<10;i++) {
            arr[i] = i*2 + 1;
        }

        for(int i=0;i<10;i++) {
            System.out.println(arr[i]);
        }

        System.out.println("Element with index 5: " + arr[5]);
    }
}
```

We say: I need space to store 10 integer values.

I assign some value to each element in the array.

Notice that I can access an element in the array
by using an int variable!

Just as I can access them with an integer **value**.

Arrays

- Data structures are an essential part of programming.
- Arrays provide a basic solution to store lots of data.
- We can use them to store, access, search, and display data.
- Here is a scenario: Let there be 50 students in a class. Write a program that finds the minimum, maximum, average, and standard deviation of these grades.
    - We can fill in their grades by generating random values between 0 and 100.
    - To find the standard deviation, we find the average first (called a *mean* in statistics), then subtract the mean from each value and square them to find out how far they are from the mean. We add these values and take their average… Easier to code than to write.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

```java
import java.util.Random;

public class StudentStats {
    public static void main(String[] args) {
        Random r = new Random(System.currentTimeMillis());
        int[] grades = new int[50];
        int sum = 0;
        for(int i=0;i<50;i++) {
            grades[i] = r.nextInt(101);
            sum += grades[i];
        }

        int min = grades[0];
        int max = grades[0];
        for(int i=0;i<50;i++) {
            if(grades[i] > max) max = grades[i];
            if(grades[i] < min) min = grades[i];
        }

        float mean = (float)sum / (float)50;
        float stdsum = 0.0f;
        for(int i=0;i<50;i++) {
            stdsum += Math.pow(grades[i] - mean,2);
        }
        double std = Math.sqrt(stdsum / (double)50);

        System.out.println("Max: " + max + ", Min: " + min + ", Average: " +
        mean + ", Std: " + std);
    }
}
```

This object generates random values…

r.nextInt returns a random value between 0 (inclusive) and 101 (exclusive).

A standard approach to find the minimum and maximum in a list of values.

Math class, in java.lang.Math, provides the power and square root functions so that I could provide some results…

Package java.lang (Ref: https://docs.oracle.com/javase/8/docs/api/java/lang/package-summary.html) is available to use without the need for an explicit import statement. For example, the classes Math, System, String, Object are ready to use...

Arrays

- The one thing that I don’t like is how we have used “50” in this example.
- Such numbers are called “magic numbers” because we don’t know why we have used 50 there… It is probable that it is the number of students, but using literal values can lead to confusion since their meaning can get lost in time.
- So, it is better to use a variable to store this value, and name it accordingly.

```
int numStudents = 50;
int[] grades = new int[numStudents]; // much better!
```

Fibonacci… again!

- Wouldn’t it be easier to use an array for Fibonacci numbers?
- Let’s calculate the first 50 of them.

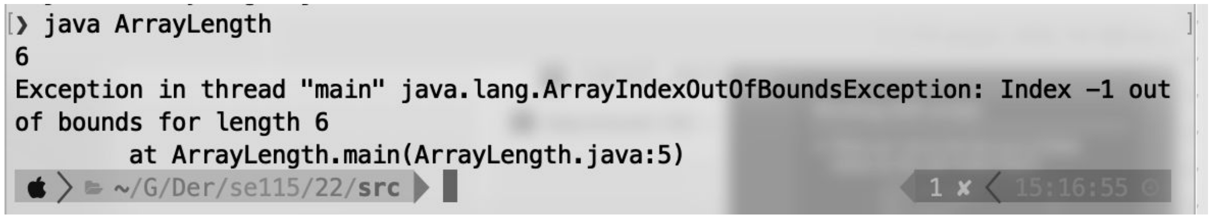
```java
public class FibonacciArray {
    public static void main(String[] args) {
        int numFib = 50; // no more magic
        int[] fib = new int[numFib];
        fib[0] = 0;
        fib[1] = 1;
        for(int i=2;i<numFib;i++) {
            fib[i] = fib[i-1] + fib[i-2];
        }

        for(int i=0;i<numFib;i++) {
            System.out.println("Fib(" + i + ") is " + fib[i]);
        }
    }
}
```

Arrays

- Now that we are familiar with arrays, here is more information.
- You can initialize the elements of an array as you are declaring it.
- Arrays in Java are collection **objects**. Being an object, it has additional methods. For example, we can access the length of an array.

```
public class ArrayLength {
    public static void main(String[] args) {
        int[] arr = { 2, 3, 13, 5, 6, 7 };
        System.out.println(arr.length);
    }
}
```

Arrays

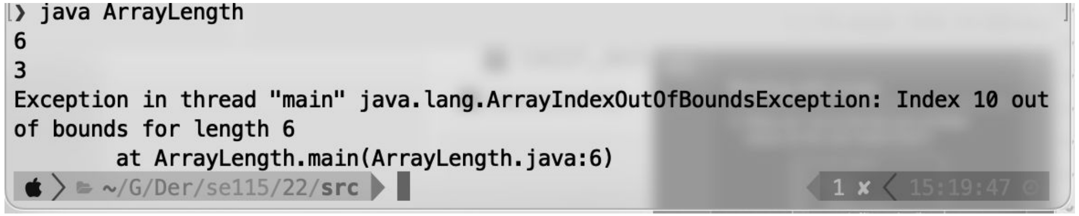


```
❯ java ArrayLength
6
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index -1 out
of bounds for length 6
        at ArrayLength.main(ArrayLength.java:5)
 ~/G/Der/se115/22/src                                      1 ✘  15:16:55
```

- What happens if you want to access an element that is not there?
  - For example, arr[-1], or an out of bounds arr[51] for an array of 50 elements.
- Let me add these lines to the “ArrayLength” class.

```
1.  public class ArrayLength {
2.      public static void main(String[] args) {
3.          int[] arr = { 2, 3, 13, 5, 6, 7 };
4.          System.out.println(arr.length);
5.          System.out.println(arr[-1]);
6.          System.out.println(arr[10]);
7.      }
8.  }
```

I had to change -1 to 1 to make the program get to line 6. Otherwise it stops running once we get the Exception.



```
❯ java ArrayLength
6
3
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 10 out
of bounds for length 6
        at ArrayLength.main(ArrayLength.java:6)
 ~/G/Der/se115/22/src                                      1 ✘  15:19:47
```

Arrays

- Arrays in Java are much more secure than C or C++, where there is no check for accessing “out of bounds” elements.
- This does not come for free, of course.

Arrays

- Arrays have a fixed size.
- If you initialize an array of 10 elements, the compiler asks for a contiguous memory space (that is each element is next to another).
- It is not possible for an array to “grow” - if you have filled in all the space, and you need more space, then you have to create a new array that is larger, and copy everything there.
- We have to copy each element one by one…
- Yes, of course I can show you how!

Arrays

```
public class GrowingArrays {
    public static void main(String[] args) {
        int[] arr = new int[5];
        for(int i=0;i<arr.length;i++) {
            arr[i] = i*i;
        }

        int[] larger = new int[10];
        for(int i=0;i<arr.length;i++) {
            larger[i] = arr[i]; // copying...
        }

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

This array is full, I need more space…

Copying one element at a time…

Arrays

- Well, Java provides a function called **arraycopy** to do this for you in the System class, if you want to. Here is its signature and then the code that shows how it works.

```
public static void arraycopy(Object src, int srcPos, Object dest, int destPos,
int length)
```

```
public class GrowingArrays {
    public static void main(String[] args) {
        int[] arr = new int[5];
        for(int i=0;i<arr.length;i++) {
            arr[i] = i*i;
        }

        int[] larger = new int[10];
        for(int i=0;i<arr.length;i++) {
            larger[i] = arr[i]; // copying...
        }

        for(int i=0;i<arr.length;i++) {
            System.out.println("arr[" + i + "] = " + arr[i] + ", larger[" + i
            + "] = " + larger[i]);
        }

        int[] copying = new int[10];
        System.arraycopy(arr, 0, copying, 0, arr.length);

        for(int i=0;i<arr.length;i++) {
            System.out.println("arr[" + i + "] = " + arr[i] + ", larger[" + i +
            "] = " + copying[i]);
        }
    }
}
```

This line says copy from “arr”, starting from position 0, copy them to “copying”, starting from position 0, and copy as many as arr.length.

Homework

- Write a Java program that stores 5 integers in an array and prints the total sum of the elements.

- Create an array of integers and print the largest number in the array. <u>The size of the array will be given by the user.</u>

- Given an <u>array of characters</u>, count how many times the letter 'a' appears.

  ```
  char[] letters = {'a', 'b', 'c', 'a', 'd', 'a', 'e'};
  ```

Next Week

- Passing arrays as arguments to functions
- Multidimensional Arrays
- **Quiz - 2** on your lab hours.

  Every lecture from Introduction to Arrays will be in the quiz.

  Study well, it will be more difficult than the first one.