Introduction to Arrays

Objectives

- Recap previous session
- Why we need arrays
- Introduction to arrays
- Exercises
- Homework Exercises
- Guidelines

Recap previous session

- What is iteration/looping?
- What looping instructions do you know?
- What is the difference between a `for` and a `while`
- What is JVM?

Why we need arrays

- You will often need to store many data values of a particular kind in your programs
    - For example we might want to keep track of the heights of our colleagues and then compute an average
- We can easily solve this kind of problem by creating as many variables as needed. For example, if we only have 5 colleagues we can create 5 variables as follows:

```
int heightAndrei = 175;
int heightBogdan = 172;
int heightRoxana = 165;
int heightMirela = 178;
int heightIoana = 199;
```

- But as you might observe, this can become troublesome if we have more data that we want to store.
    - What will happen if we want to save the height of our 40 colleagues? Should we define 40 variables? No way!
- In this lesson we will see how can we solve this problem using Arrays.

Introduction to arrays

- An array is a fixed number of data items that are all of the same type.
- The data items in an array are referred to as elements
- The folowing array declaration is similar to a declaration for a normal variable that contains a single value, except that you've typed brackets `[]` and then the name of the variable:

```
int[] numbers;
```

    - The previous statement is only a declaration of the variable of type `array of int`, but it has not also been defined or initialized. In order to do this, we type:

```
int[] numbers = new int[10]
```

        - Here we can see that we have used the `new` operator which might look familiar to you.
        - What are we doing here, simply says, create a new variable of type `int array` and then make it of size 10, meaning we can store at most 10 elements inside it.
    - Each of the data items stored in an array is accessed by the same name
        - In the previous statement the array name is `numbers`
    - You select a particular element by using an `index value` between square brackets following the array name
    - Index values are sequential integers that start from zero, and `0` is the index value for the first array element
        - In our example, the index values for the elements in the `numbers` array run from 0 to 9, so the index value 0 refers to the first element and the index value 9 refers to the last element
        - Therefore you access the elements in the `numbers` array as `numbers[0], numbers[1], numbers[2], etc...`
    - Always remember that index values start from `0` and not `1`. In our case, the third element can be accessed using `numbers[2]`, due to this rule of starting from `0`.
    - You can think of the index value for an array element as the offset from the first element:
        - The first element, is the first elements thus the offset is 0
        - The second element is offset by 1 from the first element, thus we access it via `numbers[1]`
        - etc...

- We can also specify an index for an array element by an expression in the square brackets following the array name. The expression must result in an integer value that correspond to one of the possible index values. - For example you could write numbers `[i-2]`. If i is 3 this accesses `numbers[1]`, the second element in an array.

- The only constraint for using an expression when accessing an element is that it must produce an `integer` result, and the result must be a legal index value for the array - If you use an expression for an index value that's outside the legal range for the array, the program will crush with an exception:

    - For example, let's suppose we want to access `numbers[10]` and we need to remember thar our values have indexes from 0 to 9, because our first value has index `0` and not `1`, thus, if we want to access the number with index 10, this means that we are outside the legal range for this array and JAVA will throw the following exception: `Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10`

- For a better understanding of arrays, we should start with a example. In the following snippet we want to compute the average grade score for the students in a class, this will be very familiar for you as there

is nothing new:

```
import java.util.Scanner;

public class Application {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        double sum = 0;
        double average = 0.0;
        int numberOfStudents = 7;
        double grade;
        for (int i = 0; i < numberOfStudents; i++) {
            System.out.print("Enter the grade for student: #"+(i+1)+": ");
            grade = in.nextDouble();
            sum  = sum + grade;
        }

        average = sum / numberOfStudents;
        System.out.println("The average of the grades from these 7
students is: " + average);

    }

}
```

- Now, keep in mind that we were only interested to find the average of the grades, thus, there was no need for storing the grades (in an array for example).
- Let's add another feature to our exercise, namely, suppose that we also want to save the grades for each student and then display the grade and the average next to it. In this scenario, we will need to store the grades, because we want to access the values again, later:

```
    import java.util.Scanner;

    public class Application {

        public static void main(String[] args) {
            Scanner in = new Scanner(System.in);
            double sum = 0;
            double average = 0.0;
            int numberOfStudents = 7;
            double[] grades = new double[numberOfStudents];

            for (int i = 0; i < numberOfStudents; i++) {
                System.out.print("Enter the grade for student: #"+
(i+1)+": ");
                double gradeForCurrentStudent = in.nextDouble();
                grades[i] = gradeForCurrentStudent;
```

```
                sum  = sum + grades[i];
            }

            average = sum / numberOfStudents;

            System.out.println("Student\tGrade\tAverage");
            for(int i = 0; i < grades.length; i++) {
                System.out.println((i+1)+"\t" + grades[i] + "\t" +
average);
            }

        }

    }
```

- Now let's talk about each line from the previous snippet:
  - We start by declaring the variable for holding the sum, average and how many students we have in our class (you can think that we can also read this number of students from the keyboard but for simplicity we decided to go for an hardcoded value)
  - Next we declare the array in which we want to store the grades for the 7 students. Due to the fact that each grade is of type double, our array will be an array of double values. This array will hold at most 7 elements. I say at most because we can store between 0 and 7 elements inside it, we are not forced to always fill the array.
  - After we declare ou array, we are iterating through the data entered by the user and then, at each iteration, we read the number entered by our user and store it in the grades array.
  - The number is stored at the position equal to `i`. And because each iteration will generate a new value for `i` (will be previous `i` + 1), each grade will be stored at its unique position in the array.
    - As a small simulation, when `i` = 0, the first grade will be stored at index 0, when `i` = 1, the second grade will be stored at index 1, etc.
  - After we store our recently introduced grade, we add it to the sum. And as you can see, we are accessing it via its index, the position where it has been previously saved (1 line above).
  - The next line is easy to understand as it simply computes the average but what is new follows right after.
  - You see a print statement which simply prints like a header for what we want to display, remember that our goal is to display something like a table:

```
    Student  Grade   Average
        1    6.7     8.9
        2    8.2     8.9
        3    9.6     8.9
        n    5.0     8.9
```

and this println statement simply prints these three words separated by a tab character (that's why we say `\t`)
    - And then we have a for loop in which we iterate through the elements stored inside an array. This will be the most common operation that you will do when using arrays: iterate through them.
    - You see that the condition says `i < grades.length`, that's because grades is an object of type Array, and this one has a method called `length` which simply says how many elements can be inside this array
        - I say how many elements, because if you have, say an array of 7 elements and only store 4 elements, the length will still be 7 and the positions which were not occupied, will default to having value 0.
    - And in the last print statement, we simply display each grade and the average next to it such that it is easier to see how each student compares to the average of the class

Memory Management

- when we declare an array, the JVM will reserve for us a contiguous block of memory which will be equal to the following formula: `How many bytes occupies the type for the array x How many elements can be in the array`
    - Let's suppose the following example and also consider that double type occupies 8 bytes of memory:

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

        - This means that JVM will reserve 8 x 10 bytes for us.
    - Why do we need to know this? Because we should allocate arrays close to our needs. For example, if we want to store the average temperature for each month, it is useless to have an array of size bigger than 12 as we do not have more than 12 months and if we choose a bigger number, we are simply wasting the memory of the system where our application is running.
    - Always keep in mind to allocate an array with a size close to the maximum that you will need for that variable

Common errors

- It is very common to make the mistake of accessing the element which is at an index past the valid range.
- For example if we have the following array variable declaration:

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

  the length of the array will be 5 but the index of our last element will be `4` because we start from 0 and if we want to access the element of an index outside the range (greater than 4 in our case), an IndexOutOfBounds exception will be thrown.
    - We can extract a rule from what we talked before, something like: `The elements of an array can be accessed by using an index between 0 and length-1, inclusive!`

Initializing an array

- There are multiple ways to initialize an array:

1. Initialize manually each element:

```
int[] arr = new int[3];
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
```

2. Initialize the elements using a loop:

```
Scanner in = new Scanner(System.in);
int[] arr = new int[3];
for(int i = 0; i < arr.length; i++) {
    System.out.println("Enter a value: ");
    arr[i] = keyboard.nextInt();
}
```

3. Initialize it when you declare it:

```
int[] arr = new int[]{1,2,3};
```

    - Note that initializing an array using the last method, we should not specify how many elements are inside the array! JAVA is smart enough to compute it herself.

Exercises

1. Create a JAVA program which initializes an array with values read from the keyboard. The program should display at the end, the elements of the array, each on a separate line. For this functionality, of displaying the array, create a method which receives an array of int as parameter and displays the array.

    - Solution:

```
import java.util.Scanner;

public class Application {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int[] arr = new int[5];
        for(int i = 0; i < arr.length; i++) {
            System.out.print("Enter a value: ");
            arr[i] = in.nextInt();
        }
```

```
            display(arr);
        }

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

        }

    }
```

2. Create a JAVA method which counts how many even numbers are inside an array. The method should return the result.
    - Sample Input: 1 2 3 4 6 8 120 12 14
    - Sample Output: 7
    - Sample Input2: 1 3 5 7 9
    - Sample Output2: 0
    - Solution:

```
    public class Application {

        public static void main(String[] args) {
            int[] arr = new int[]{1,2,3,4,6,8,120,12,14};
            int evenNumbersCount = count(arr);
            System.out.println(evenNumbersCount);
        }

        public static int count(int[] arr) {
            int count = 0;
            for(int i = 0; i < arr.length; i++) {
                if(arr[i] % 2 == 0) {
                    count++;
                }
            }
            return count;
        }
    }
```

3. Create a JAVA method which displays the elements from an array, which are on odd indexes (1,3,5, etc). The elements should be displayed on the same line
    - Sample Input: 1 2 3 4 5 6 7 8

- Sample Output: 2 4 6 8
- Solution:

```
import java.util.Scanner;

public class Application {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int[] arr = new int[] {1,2,3,4,5,6,7,8};
        display(arr);
    }

    public static void display(int[] arr) {
        for(int i = 0; i < arr.length; i++) {
            if( i % 2 != 0) {
                System.out.print(arr[i] + " ");
            }
        }
    }
}
```

4. Create a JAVA method which receives an array of double as parameter and returns the maximum value from the array:
   - Sample Input:
     - 1.2 2.3 2.6 2.8 2.9 2.68
   - Sample Output: 2.9
   - Solution:

```
public class Application {

    public static void main(String[] args) {
        double[] arr = new double[] {1.2, 2.3, 2.6, 2.8, 2.9};
        double max = findMax(arr);
        System.out.println(max);
    }

    public static double findMax(double[] arr) {
        double max = arr[0];
        for(int i = 0; i < arr.length; i++) {
            if( arr[i]> max) {
                max = arr[i];
            }
        }
        return max;
```

```
        }
    }
```

5. Create a JAVA method which returns the index of the element which has the minimum value from an array. The method should receive the array as a parameter
    - Sample Input:
        - 2.2 1.2 2.3 2.6 2.8 2.9 2.68
    - Sample Output:
        - 1
    - Solution:

```
    public class Application {

    public static void main(String[] args) {
        double[] arr = new double[] {2.2, 1.2, 2.3, 2.6, 2.8, 2.9};
        int minPos = findMinPos(arr);
        System.out.println(minPos);
    }

    public static int findMinPos(double[] arr) {
        int minPos = 0;
        for(int i = 0; i < arr.length; i++) {
            if( arr[i]< arr[minPos]) {
                minPos = i;
            }
        }
        return minPos;
    }
}
```

6. Create a JAVA program which reads an array with integer numbers. The program should display how many elements from the array are equal with the difference between the maximum and minimum value of the array.
    - Sample Input:
        - 7 7 9 2 4
    - Sample Output: 2
    - Solution:

```
    public class Application {

    public static void main(String[] args) {
        int[] arr = new int[] {7,7,9,2,4};
        int min = findMin(arr);
        int max = findMax(arr);
        int difference = max - min;
        int elementsEqualWithDifference =
countElementsEqualToValue(difference, arr);
        System.out.println(elementsEqualWithDifference);
    }

    public static int findMin(int[] arr) {
        int min = arr[0];
        for(int i = 0; i < arr.length; i++) {
            if( arr[i]< min) {
                min = arr[i];
            }
        }
        return min;
    }

    public static int findMax(int[] arr) {
        int max = arr[0];
        for(int i = 0; i < arr.length; i++) {
            if( arr[i] > max) {
                max = arr[i];
            }
        }
        return max;
    }

    public static int countElementsEqualToValue(int value, int [] arr)
{
        int count = 0;
        for(int i = 0; i < arr.length; i++) {
            if(arr[i] == value) {
                count++;
            }
        }
        return count;
    }
}
```

Homework exercises

1. Create a JAVA program which initializes an array of 10 doubles with values read from the keyboard. The program should display the elements of the array on a single line, separated by a space. For the logic capable of displaying the elements, create a method which receives an array as parameter and then displays the elements according to the requirement

2. Create a JAVA method which counts how many odd numbers are inside an array with int values. The method should return the result.
    - Sample Input: 1 2 3 4 6 8 120 12 14
    - Sample Output: 2
    - Sample Input2: 1 3 5 7 9
    - Sample Output2: 5
3. Create a JAVA method which displays the elements from an array, which are on even indexes (0,2,4,6, etc). The elements should be displayed on the same line
    - Sample Input: 1 2 3 4 5 6 7 8
    - Sample Output: 1 3 5 7
4. Create a JAVA method which receives an array of double as parameter and returns the minimum value from the array:
    - Sample Input:
        - 2.4 1.2 2.3 2.9 2.8 2.6
    - Sample Output: 1.2
5. Create a JAVA method which returns the index of the element which has the maximum value from an array. The method should receive the array as a parameter
    - Sample Input:
        - 2.2 1.2 2.3 2.6 2.8 2.9 2.68
    - Sample Output:
        - 5
6. Create a JAVA program which reads an array with integer numbers. The program should display how many elements from the array are equal with double the difference between the maximum and minimum value of the array.
    - Sample Input:
        - 7 7 6 2 14 8 9 12 13
    - Sample Output: 1
    - Explanation: The difference between the maximum value (14) and minimum value (2) is 12 and half of this value is 6. In our array we have only one element which is equal to 6.

Guidelines

- As this is a brand new concept, try to revisit the material as many times as needed
- Arrays are a fundamental piece in programming, thus is crucial to have a good understanding of them