Carleton University
Department of Systems and Computer Engineering
SYSC 2004 – Object Oriented Software Development – Winter 2017

Lab 4 –Arrays and Grouping of Objects

- **Objectives:**

The objective of this lab is to give you some practice in arrays and grouping of objects. At the end of this lab, you should be comfortable with performing any operation on arrays.

- **Attendance/Demo:**

To receive credit for this lab, you must demonstrate your lab work. When you have finished all the exercises, call a TA, who will grade the code you wrote. For those who do not finish during the lab time, a submission link will be provided on cuLearn to give you the chance to submit your solution. You will have a week to submit unfinished labs.

Note: You are required to go to your lab session. In case of your absence, your submission on cuLearn will not be considered. The TAs will collect attendance at the beginning of each lab.

- **Getting Started:**

Step 1: Open Eclipse.
Step 2: Choose a workspace where you want to store your file (*Preferably on your USB*).
Step 3: Create a new Java project named "**Sysc2004Lab4**".

- **Exercise 1: Performing basic operations on an array:**

Array is one of the most important topics in Java. If you would be attending an interview for a software developer position, you should definitely be prepared for questions on array.

An array is a **data structure** which stores a fixed collection of variables of the **same type.** Below is how an array is defined in Java.

```
datatype [] arrayName = new datatype [array Size];
```

For example if you would like to declare an array of 5 integers. Below is how it is done.

```
Int [] arrayOfFiveIntegers = new int[5];
```

Now that we know what an array is, we will perform basic operations like adding, removing and searching for an element in array.

Step 1: Create a class called **BasicArray** in the *src* folder of your created project.

Step 2: Create a method named **sumArray** that accepts an integer array and returns the sum of the elements in the array. Below is the method signature.

```
public int sumArray (int [] array);
```

Step 3: loop through the elements of the array, sum the elements and return the sum. An example of looping through an array is shown below.

```java
/**
 * This method is used to take an array of words and
 * form a sentence with it.
 * @param array
 * @return
 */
public String joinWords(String[] array) {
    String words ="";

    //array.length gives the number of elements in the array
    for (int i=0; i<array.length; i++) {

        //array[i] is used to access individual elements in the array.
        words += array[i] + " ";
    }

    return words;
}
```

Step 4: Test your code in the main method using the example below as reference.

```java
public static void main(String[] args) {
    // TODO Auto-generated method stub
    //create an instance of your class
    HelloWorld helloWorld = new HelloWorld();

    //create an array of 3 words
    String[] stringArray = {"This", "is", "Mark"};

    System.out.println(helloWorld.joinWords(stringArray));

    //Another way of creating an array
    String[] anotherStringArray = new String [3];
    anotherStringArray[0] = "This";
    anotherStringArray[1] = "is";
    anotherStringArray[2] = "Mark";

    System.out.println(helloWorld.joinWords(anotherStringArray));
}
```

- **Adding an element to an array**

In this part, you would be learning how to add elements to an array.

Step 5: Create a method called **getEvenNumbers** which would accept an **integer** as a parameter and return an array of **all the even numbers that is less than or equal to the integer**. Set the integer passed as the size of your array.

- If the array passed is null, return null.
- if the integer passed is less than 0, return null.

Step 6: Create a method called **getPrimeNumbers** which would accept an **integer** as a parameter and return an array of **all the prime numbers that is less than or equal to the integer**. Set the size of your array to the integer passed.

- If the array passed is null, return null.
- if the integer passed is less than 0, return null.

- **Searching an array for an element**
  Since you know how to use conditional statements (if, else if & else) and loop through an array, you should be able to search for an element in an array.

Step 7: Create a method called **findSmallestAndLargest** that accepts an **array of numbers (double)** and returns an **array of the smallest and largest number**. For example, if an array containing [1, 5, 4, 9, 8] was passed into the method, the method should return [1, 9]. Test your code in the main method.

- if the array passed to the method is null or empty, you should return null.

Step 8: Create a method called **findDuplicate** that accepts an **array of strings** and returns the **first string that appear twice** in the array or **null** if there is none. For example, if the array is [“tom”, “mark”, “john”, “tom”], you should return “tom”.

- If the array passed to the method is null or empty, you should return null.

- **Removing an element from an array**
  Since we can add elements to an array, we should be able to remove an element from the array. For example, if we have an array [1, 2, 3, 4] and we want to remove 3, the new array would be [1,2,4]. Looking at this example, you will notice the size of the new array was **decreased** by 1.

  Therefore to remove an element from an array, you are to create a `newArray` with `oldArray.length - 1` as the array size and add the elements in the `oldArray` (without the element you want to remove) to the `newArray`.

Step 9: Create a method called **remove** that accepts an **array of integers** and **an integer to remove.** The method returns **a new array** of integers **without** the integer to remove.

- If the array passed to the method is null or empty, you should return null.

Step 10: Copy the **BasicArrayTest** file to your *src* folder and run the Junit test to validate the code you have written. Refer to lab 02 if you have forgotten how to run Junit tests.

- **Exercise 2: Grouping of objects in Java:**
  In this exercise, you would be learning how arrays can be used with objects in Java.

Step 1: Create a class named **Person** in the *src* folder of your project.

Step 2: Add a Javadoc description to your class specifying the description of the class, your name, student number and today's date.

Step 3: Add the following field to your Person class:

- **`firstName`**
- **`lastName`**
- **`age`**
- **`height`**

Step 4: Create a constructor that accepts all the fields you created in step 3 and initialize the fields.

Step 5: Create another constructor that accepts the **firstName** & **lastName** and initialize the instance variables of the `Person` class. Don't forget to use the **this ()** method.

Step 6: Create a **default constructor** for the `Person` class.

Step 7: Add the getters and setters for all the fields you created in Step 3. A simple way of doing this is to **right-click on your class -> source -> generate getters and setters -> check the fields -> ok.**

- **Note:** The shortcut is discouraged for people who do not know the syntax for getters and setters.

Step 8: Create an **equals** method for your Person class.

Now that we have created an Object, person, we would perform some array operations on it.

Step 9: Create a method called **getSiblings** which accepts an **array of Person,** and returns an array of the people that have age less than the **current instance of person i.e this**. For example, if an array of [Person 2, Person 3, Person 4] is passed to the method, the method will compare the age of person 2, 3 and 4 to **this.age** and return an array of all the people whose age is less than this.age.

- If the array passed to the method is null, you should return null.

Step 10: Create a method called **OrderPeopleByHeight** which accepts an **array of Person,** orders the people according to their height and returns the **ordered array of people**.

Step 11: Create a method called **getPeopleWithSameFirstName** which accepts an **array of Person,** loops through the array, find the Persons with the same first name and returns the **array of those Persons.**

Step 12: Copy the **PersonTest** file to your src folder and run the Junit test to validate the code you have written.

- **Exercise 3: Create a Student Class:**

  This exercise is similar to exercise 2 and would help you in getting a deeper knowledge with grouping of objects in Java.

Step 1: Create a class named **Student** in the src folder of your project.

Step 2: Add a Javadoc description to your class specifying the description of the class, your name, student number and today's date.

Step 3: Add the following field to your Person class:

- **`ID`**
- **`name`**
- **`gpa`**

Step 4: Create a constructor that accepts all the 3 fields you created and initialize the fields.

Step 5: Create another constructor that accepts the **id** & **name** and initialize the instance variables of the Student class. Don't forget to use the **this()** method.

Step 6: Create a **default constructor** for the **Student** class.

Step 7: Add the getters and setters for all the fields you created in Step 3.

Step 9: Create a method called **getSmartStudents** which accepts an **array of Student,** and returns an **array of the smart students**. A student is considered smart if he or she has a GPA of above 11.0 out of 12.0.

- if the array of student is null, return null

Step 10: Create a method called **removeStudent** which accepts an **array of Student** and an integer **index.** The method will remove the student at the **index** from the array and returns an array of the remaining students.

Step 11: Create a method called **createRandomStudents** which accepts no parameters but it returns **an array of 100 students** with a consecutive even numbers as the id of the students starting from 2. This means **student 1 would have id 2, student 2 -> 4, student 3 -> 6, student 4 -> 8 ….. And student 100 -> 200**.

Step 12: Copy the **StudentTest** file to your src folder and run the Junit test to validate the code you have written.