Lab 2: Java Arrays & Strings

Lab Objectives: After this lab, the students should be able to:

- Manipulate arrays
- Manipulate Strings
- Solve problems using Strings and Arrays

Part 1: Working with Java Arrays

An array is a collection of similar types of data. For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 names:

```
String[] array = new String[100];
```

Java arrays are declared as follows:

```
dataType[] arrayName;
```

- *dataType* can be primitive data types like int, char, double, byte, etc. or Java objects
- *arrayName* is an identifier

For example,

```
double[] data;
```

To define the number of elements that an array can hold, we have to allocate memory for the array in Java. For example,

```
// declare an array
double[] data;
// allocate memory
data = new double[10];
```

The declaration and initialization can be done at once:

```
// declare and allocate an array
double[] data = new double[10];
```

We can initialize arrays during declaration. For example,

```
//declare and initialize and array
int[] age = {12, 4, 5, 2, 5};
```

This will allocate space with number of initialization elements.

When you pass an array to a function, it is passed by ***reference***, not by value. Because arrays can be quite big, java don't allow big arrays to be copied by default. To avoid changing array content one can create his own copies using the clone method:

```
int[] copyOfArray = array.clone()
```

1. Write helper methods `getArray` and `printArray` which reads and prints an array of n integers, respectively. Use these methods to test the following problems.
2. Write a method called `areTheSame` that takes as input two arrays of integers and tells if all of the elements of the two arrays are the same. The order does not matter. For example, [4, 3, 1] and [1, 3, 4] are the same.
3. Write a method `areSorted` that checks if all the elements in the array are sorted, either incrementally or decrementally.
4. Write a method `getIndexMinValue` that takes as parameters an array of n integers and an index i and returns the index of the minimum value found in the array starting from i. In case, the min value exists twice, return the smallest index. For example: if the array contains [44, 1, 125, 38, 189], `getIndexMinValue(array, 2)` returns 3 because 38 is the minimum starting from index 2 i.e. [125, 38, 189]
5. Write a method called `sortValues` that takes as input an array of integers and sort it. The algorithm is simple. It is as follows:
   - First start at position 0 of the array and find the index of the minimum value between 0 and n-1, and return its index, use `getIndexMinValue(array, 0)`
   - Then swap the value at position 0 with the value at position index,
   - Increment one step (position should be now 1), search for the minimum value between 1 and n-1, and return its index (`getIndexMinValue(array, 1))`.
   - Then swap the value at position 1 with the value at position index
   - Repeat the process until the position is at n-1 (last element).

Part 2. Working with strings

Strings are an **object** variable type. They are **immutable**, which means they can not be changed. That doesn't mean you can never modify a string when writing a program, it just means that when you're modifying it, it's actually creating a new string each time a change is made.

Strings are stored as arrays of chars behind the scenes, which allows strings to be indexed. Their indexing is 0-based, so the first character of the string is at index 0.

Because strings are objects made from the String class, they also have some handy associated methods. Some of the most commonly used string methods for this class will be:

```
length, equals, equalsIgnoreCase, toLowerCase, toUpperCase,
trim, charAt, substring, indexOf, lastIndexOf, compareTo,
compareToIgnoreCase, etc.
```

Explore the methods of the String class offers.

Creating strings

```
String name =  "test";
String  name  =  new  String("test");
```

Both give a reference variable called `name` pointing to the String object `"test"`. However, they are subtly different! The former says to use the string **pool**. The second creates a new object and it is less efficient."

What is a string pool?

In some production applications, strings can use up 25–40 percent of the memory in the entire program. Java realizes that many strings repeat in the program and solves this issue by reusing common ones. The string pool, also known as the intern pool, is a location in the Java virtual machine (JVM) that collects all these strings. The string pool contains literal values that appear in the program. For example, "test" is a literal and therefore goes into the string pool. myObject.t oString() is a string but not a literal, so it does not go into the string pool.

1. **Testing for equality of strings using equals()**

You have to be very careful when testing for equality of strings in Java. Run the following statements and explain what happens.

```
String x = "Hello World";
String y = "Hello World";
System.out.println(x == y);
String z = " Hello World".trim();
System.out.println(x == z);
x = new String("Hello World");
```

```
System.out.println(x == y);
System.out.println(x.equals(z));
```

How would you compare the equality of strings?

2. Use string methods

Given the String: “This is Lab #2 of EE423 Advanced Programming Course”, write a Java method which:

- converts all alphabets to capital letters and print out the result;
- converts all alphabets to lower-case letters and print out the result; and
- prints out the length of the string.
- prints out the index of Advanced.

3. Generate a password from user information.

Write a Java method which read the first name, last name and age of the user and generates a password as follows :

The password contains the last number of age, the first and last chars of the first name, two middle chars of the family name and the first number in age

4. Sorting array of Strings

Write a method `String [] sortArray(String [] a)` that takes as input an array of strings and sorts it. The passed array must remain unchanged. The returned array must be sorted.

5. Palindrome

Write a function `public String findPalindrome(String s)` that returns the longest palindrome in a string. You can use the function `toLowerCase()` to convert the string to lower case.

- Return "The string is too short" if the string is less than 3 characters.
- Return "No palindrome found" if there is no palindrome of at least 3 characters,
- Return "Only alphabetical characters are allowed" in case there is a different character than space or alphabetical characters.
- In case there are many possible palindromes with the highest length, return the leftmost one.

```
findPalindrome("ok") // The string is too short

findPalindrome("ok!") // Only alphabetical characters are allowed
findPalindrome("coucou") // No palindrome found

findPalindrome("good morning Madam") // Madam

findPalindrome("nurses run"); // nurses run (notice the spaces)

findPalindrome("madam good morning Madam") // madam

findPalindrome("room llevell morning refer") // om llevell mo
```