ENSF 593/594 – Principles of Software Development

Fall 2019

Lab Assignment 0 Part II: Introduction to Java

Due Dates
Before 11:59 on Tuesday September 17th

The objectives of this lab are to:

1. Simple memory management in Java
2. Arrays
3. Review of methods (pass by value vs. pass by reference)

<u>The following rules apply to this lab and all other lab assignments in future:</u>

1. Before submitting your lab reports, take a moment to make sure that you are handing in all the material that is required. If you forget to hand something in, that is your fault; you can't use `I forgot' as an excuse to hand in parts of the assignment late.
2. **<u>20% marks</u>** will be deducted from the assignments handed in up to **<u>24 hours</u>** after each due date. It means if your mark is X out of Y, you will only gain 0.8 times X. There will be no credit for assignments turned in later than 24 hours after the due dates; they will be returned unmarked.

Exercise 1 Arrays and Functions I (16 marks)

Task 1.1: insertAt (16 marks)

Develop function InsertAt with according to the following requirements.

Function Requirements: `insertAt` which accepts the following 3 parameters: an array of floats, a float value to be inserted, and an int which is the index where the element is to be inserted in the array floats. This function then inserts the element at the index it has received. The array in setup must be changed after calling the `insertAt` function.

Note: Your function `insertAt` must check to make sure that the index provided is not greater than the length of the array, and is not less than 0. If this is the case, the array should not be changed, and an appropriate error message stating that the operation couldn't be done should be displayed to the console by this function.

```
void main ()
{
   float [] array = {6.7, 5.9, 10, 23, 44, 4.6, 9.1, 100, 79};

  //call InsertAt function here

  //pirnt array here
}
```

So, if the index passed to function `insertAt` is 3, and the value to be inserted is 255, value 255 will be inserted in index 3 and all other elements will move up in the array

6.7	5.9	10	23	44	4.6	9.1	100	79
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Therefore, the length of the array will increase by 1.

6.7	5.9	10	255	23	44	4.6	9.1	100	79
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

Task 1.2: swapArrays (8 marks)

Function Requirements:
`SwapArrays` accepts the two arrays of floats and swaps their contents.

Note: Your function `swapArrays` must check to make sure that both arrays have the same length. If the lengths of the arrays are different, the two arrays should not be changed, and an appropriate error message stating that the operation couldn't be done should be displayed to the console by this function.

```
void main ()
{
   float [] first_array = {6.7, 5.9, 10, 23, 44, 4.6, 9.1};
   float [] second_array = {11, 12, 13, 14, 15, 16, 17};

  //call SwapArrays function here

//print both arrays here
}
```

For example, if we have the following 2 arrays in setup:

first_array	6.7	5.9	10	23	44	4.6	9.1
	0	**1**	**2**	**3**	**4**	**5**	**6**

second_array	11	12	13	14	15	16	4.6
	0	**1**	**2**	**3**	**4**	**5**	**6**

After calling the `swapArrays` function, the content of the arrays in setup would be as follows:

first_array	11	12	13	14	15	16	4.6
	0	**1**	**2**	**3**	**4**	**5**	**6**

second_array	6.7	5.9	10	23	44	4.6	9.1

<u>Task 1.3: allUnique (8 marks)</u>

Write a function called allUnique that will take an array of integers as input and returns true if all elements in the array are unique (i.e. no number appears twice). The function should return false otherwise.

You can define the following array in your main function to send to this method and to testt (i.e. Declare an array inside your program and initialize it to the above values):

3	5	4	20	10

The above sample array is an example of an array which contains all unique elements. You should test your program with an array that is unique like above and one that isn't.

What to Submit:
Submit all .Java files on D2L in one folder.