CSC 151 Assignment 2

Objectives:

- Measuring the efficiency of basic operations
- Analyzing array operations

Definition

In this assignment you will implement various operations on arrays and analyze the complexity of them. Considering the methods provided below, please write the following class.

Class ArrayOperations

- Name of your file will be **ArrayOperations.java** (case-sensitive)
- It will be in **package assignment** (it is required otherwise **Gradescope** cannot run the tests)
- **Do not use ArrayList, Vector, Hashtable or any other data structures in java.**
- Write a public static method **doubleExists** which has an integer array parameter that checks if there exists two integers N and M such that N is the double of M (i.e., N = 2 * M).

  In other words, you will find two distinct indices i and j where

  arr[i] = 2 * arr[j] or arr[j] = 2 * arr[i],

  and 0 <= i, j < arr.length

  and i ≠ j.

  Analyze the time complexity of the method.

  Example 1:

```
Parameter array:[0,4,3,2,7]

Return true since 4 is 2 * 2
```

  Example 2:

```
Parameter array:[0,5,3,2,7]

Return false since none of them is a double of the other
```

  Example 3:

```
Parameter array:[]

Return false since it is empty
```

- Write a public static method **moveGivenValue** which has an integer array and an integer parameters and moves all values that are not equal to the given value in the array to the beginning of the array while maintaining the relative order and without making a copy of the array (e.g. complete the operation in-place on the same array). **Analyze the time complexity of the method.**

  Example 1:

```
Parameter array:[0,5,0,0,1]
Parameter value: 0
After method call: [5,1,0,0,0]
```

  Example 2:

```
Parameter array:[0]
Parameter value: 0
After method call: [0]
```

  Example 3:

```
Parameter array:[1,2]
Parameter value: 0
After method call: [1,2]
```

  Example 4:

```
Parameter array:[1,2,1,3,1,6]
Parameter value: 1
After method call: [2,3,6,1,1,1]
```

- Write a public static method **sortEvenOdd** which has an integer array parameter that moves the even numbers to the front and odd numbers after the even numbers. Do not use any other array or data structure and move the numbers in place. The ordering of the even and odd numbers is not important. **Analyze the time complexity of the method.**

Example 1:

```
Parameter array:[5,10,7,2,4,8]

After method call: [8,10,4,2,7,5] or [10,8,4,2,7,5] or [4,10,8,2,5,7] …
```

Example 2:

```
Parameter array:[ 3,1]

After method call: [3,1] or [1,3]
```

Example 3:

```
Parameter array:[]

After method call: []
```

Example 4:

```
Parameter array:[2,4]

After method call: [2,4] or [4,2]
```

- The class will have
  - No **instance variables**
  - No **public constructors**
  - No **public set/get** methods.
  - No **equals** method.
  - No **toString** method.
- Test your ArrayOperations class in the main method.
- Test your class by JUnit

General Submission Instructions

- You may have groups for the assignments **if it is allowed.** If you have a group state it **explicitly in the honor code** by writing all group members' names. Please name the pdf files accordingly. **However, all group members should submit individually.**
- All submissions require a package called **assignment** and your source codes (.java files) will be under it.
- You should submit
    - Your java files to **Gradescope** as required and
    - Your java files and sample outputs for various scenarios as a **single pdf** file named as **YourFullName(s)_AssignmentN.pdf** with the **honor code** at the top as a comment to Nexus under the appropriate link. Replace N in AssignmentN with the appropriate value. (e.g., if I submit Assignment1 I will submit a pdf file whose name is **ZeynepOrhan_Assignment1.pdf.** If I have a group whose members are me and Jane Doe, then the file name will be **JaneDoeZeynepOrhanAssignment1.pdf** and the honor code will be modified). Use the template .docx file attached and modify it accordingly.
- **If you use any resource to get help, please state it in the honor code. Otherwise, it will be an honor code violation!!!**
- **Gradescope provides a feature to check code similarity by comparing all submissions. If the percentage of your code's similarity is above a threshold, then this will be an honor code violation!!!**
- Make sure that you have no compiler errors.
- When you have compiler error free codes and submit appropriately to Gradescope, your code will be tested automatically and graded. You do not need to know the test cases, but you will see the test results. Informative messages will be provided for success and failure cases.
- You can resubmit your files any number of times if you have failing tests until the due date.
- Your final submission or the one you selected will be graded after the due date of the assignment.
- Please start as early as possible so that you may have time to come and ask if you have any problems.
- Read the late submission policy in the syllabus.
- Please be aware that the correctness, obeying the OOP design and submission rules are equally important

- Gradescope testing will generally be 100% and sometimes manual grading will be done. Please be aware that **your grade can be reduced if you do not follow the instructions and apply good programming practices**.
- Use the starter code if provided.
- Do not use any predefined Collection classes of Java unless stated otherwise.

Assignment specific instructions

- General submission instructions apply.
- **Group work: NOT ALLOWED!!!**
- **Due date:** January 16, 2021 until 11:50 PM
- **Nexus:** Submit the code file (**ArrayOperations.java**) as a single pdf file named as **YourFullName_Assignment2.pdf** with the honor code at the top as a comment under Assignment 2 link.
- **Gradescope:** Submit the java files (**ArrayOperations.java**) under the Assignment 2
- **Grading:**
    - Gradescope: Yes
    - Manual: No
- **Starter code:** Use the code given below

```java
package assignment;

/*
 * Honor code
 */
/**
 * Array operations that use search and sort
 *
 * @author
 * @version
 *
 */
public class ArrayOperations {

	/**
	 * doubleExists:
	 * @param arr an integer array
	 * @return true if the array has a double of the other
	 */




	/**
	 * moveGivenValue:
	 * Given an integer array it moves the integers
	 * that are equal to val to the end
	 * and maintains the ordering
	 *
	 * @param arr an integer array
	 * @param val a target value
	 */



	/**
	 * sortEvenOdd:
	 * Given an integer array it moves the even numbers to the front
	 * and odd numbers to the end
	 * and it does not maintain the ordering
	 *
	 * @param arr an integer array
	 * @return the arr moving evens to the front
	 */




	/**
	 * display:
	 * Displays an integer array with a space in between the numbers
	 *
	 * @param arr an integer array
	 */
```

```java
/**
 * Tests the operations
 *
 * @param args
 */
public static void main(String[] args) {
	int[] arr1 = { 0, 4, 3, 2, 7 };
	int[] arr2 = { 0, 5, 3, 2, 7 };
	int[] arr3 = {};
	System.out.println("============================");
	System.out.println("doubleExists tests");
	System.out.println();
	System.out.println("doubleExists(arr1)");
	display(arr1);
	System.out.println(doubleExists(arr1));
	System.out.println();
	System.out.println("doubleExists(arr2)");
	display(arr2);
	System.out.println(doubleExists(arr2));
	System.out.println();
	System.out.println("doubleExists(arr3)");
	display(arr3);
	System.out.println(doubleExists(arr3));
	System.out.println();
	System.out.println("============================");

	int val = 0;
	int[] arr4 = { 0, 5, 0, 0, 1 };
	int[] arr5 = { 0 };
	int[] arr6 = { 1, 2 };
	System.out.println("moveGivenValue tests");
	System.out.println();
	System.out.println("moveGivenValue(arr4, 0)");
	System.out.print("Before:");
	display(arr4);
	moveGivenValue(arr4, val);
	System.out.print("After:");
	display(arr4);
	System.out.println();

	System.out.println("moveGivenValue(arr5, 0)");
	System.out.print("Before:");
	display(arr5);
	moveGivenValue(arr5, val);
	System.out.print("After:");
	display(arr5);
	System.out.println();
	System.out.println("moveGivenValue(arr6, 0)");
	System.out.print("Before:");
	display(arr6);
	moveGivenValue(arr6, val);
	System.out.print("After:");
	display(arr6);
	System.out.println();
	val = 1;
	int[] arr7 = { 1, 2, 1, 3, 1, 6 };
	System.out.println("moveGivenValue(arr7, 1)");
```

```
            System.out.print("Before:");
            display(arr7);
            moveGivenValue(arr7, val);
            System.out.print("After:");
            display(arr7);
            System.out.println();
            System.out.println("============================");

            int[] arr8 = { 5, 10, 7, 2, 4, 8 };
            System.out.println("sortEvenOdd tests");
            System.out.println();
            System.out.println("sortEvenOdd(arr8)");
            System.out.print("Before:");
            display(arr8);
            sortEvenOdd(arr8);
            System.out.print("After:");
            display(arr8);
            System.out.println();
            int[] arr9 = { 3, 1 };
            System.out.println("sortEvenOdd(arr9)");
            System.out.print("Before:");
            display(arr9);
            sortEvenOdd(arr9);
            System.out.print("After:");
            display(arr9);
            System.out.println();
            int[] arr10 = { 8, 6, 2, 4 };
            System.out.println("sortEvenOdd(arr10)");
            System.out.print("Before:");
            display(arr10);
            sortEvenOdd(arr10);
            System.out.print("After:");
            display(arr10);
            System.out.println();
            System.out.println("============================");
        }
}
```

Output of the above code:

```
============================
doubleExists tests

doubleExists(arr1)
0  4  3  2  7
true

doubleExists(arr2)
0  5  3  2  7
false

doubleExists(arr3)

false

============================
moveGivenValue tests

moveGivenValue(arr4, 0)
Before:0  5  0  0  1
After:5  1  0  0  0

moveGivenValue(arr5, 0)
Before:0
After:0
```

```
moveGivenValue(arr6, 0)
Before:1  2
After:1  2

moveGivenValue(arr7, 1)
Before:1  2  1  3  1  6
After:2  3  6  1  1  1

============================
sortEvenOdd tests

sortEvenOdd(arr8)
Before:5  10  7  2  4  8
After:8  10  4  2  7  5

sortEvenOdd(arr9)
Before:3  1
After:3  1

sortEvenOdd(arr10)
Before:8  6  2  4
After:8  6  2  4

============================
```