CS2401 – Week 4-5

With this lab assignment, you are going to get a chance to practice on arrays and review some early concepts of OOP (Object-Oriented Programming). We hope this lab helps you feel more comfortable manipulating again concepts from CS1 and embarking on CS2 material. Let's get started!

What is the goal of this lab?

We expect that, by the time you complete this lab you will:

- Be more confident about using arrays (1D and 2D), including arrays of objects;
- Remember essential elements of classes and objects; and
- Feel more comfortable about approaching and solving problems.

What problem will you be addressing in this lab? In this lab, you are going to put yourself in my shoes as an instructor and handle student information, gather it, store it, and then manipulate it as a whole. Let me explain.

As an instructor, you will handle individual student's information. For this, you will create a class Student as follows:

Student: (java file Student.java) contains

- The student's first and last name: a string
- The student's major: a string
- The list of grades over the semester: a 1D array of integers

Of course, you need to be able to edit the student's information: first and last names, major, grades.

With this information, you need to be able to compute the grade average of the student. There should be two ways to compute the average: plainly (all grades are equal) or with special weights (given as an array of weights).

Testing using JUnit: Most methods of Student.java need to be tested using JUnit tests. Those methods to be tested are clearly indicated directly in the starter code of Student.java. Do justify the test cases you use in java comments.

Now in another java file (java file Execute.java)

As an instructor, you will be able to create instances of the class Student (reading the information from a file: example provided to you, studentsInfo.txt) and store them in a 1D array of Students. You will also be able to create a 2D array of all students' grades (one row of grades per student, the first row corresponding to the grades of the first in your 1D array of students), which will represent an excel sheet. Given this 2D array of grades (excel sheet of grades), you will be able to identify the student with the best average (with either average you want to use), identify the assignment in which the whole class did best (best average over the whole classroom), conduct the same

activities only on a subset of the classroom by major (e.g., only computer science students, only math students, only history students, etc.).

Testing using the main method: All of the 2D functionalities will be tested inside the main method that is available in Execute.java. Instructions and guidelines are available in comments directly in the code for you to follow.

What do you have to do?
1/ Complete the file Student.java where prompted and following methods description provided directly in the code. Look for "COMPLETE CODE HERE" and "CHANGE?".
2/ Complete the file Execute.java where prompted and following methods description provided directly in the code. Look for "COMPLETE CODE HERE" and "CHANGE?".
3/ Answer the following questions in a separate word file called YourLastName-YourFirstName-Week4-5.docx.
Q1. In Student.java, what changes would you need to make if we were not to have any getters or setters available? Note: saying that we would need to add getters and setters is not a valid answer.
Q2. In Execute.java; let's say that you have 50 students and 20 grades each. How would you go about gathering all the grades of the students in a 2D array without using more than about 50 memory cells?

What should you turn in?
You should submit 1/ your word document, 2/ Student.java that you have completed, 3/ Execute.java that you have completed, and 4/ the new file you created to run your JUnit tests: StudentTester.java.

How should you submit your work?
You should follow the specific submission instructions and guidelines given by your own TA in lab.
Failing to follow submission instructions and guidelines given by your respective TA will result in up to 15 points off your overall grade in this lab. So please pay attention.
Additionally, your word and java files are expected to be neat and clear (organization, grammar, and spelling for the word file / indentation and clear variable naming for the java file). Failing to do so will result in up to 15 points off. On the other hand, extra neat and clear work will be rewarded by up to 10 extra points.

By when should you submit your work?
Due date: Friday September 27 at 11:59pm
Lateness rule: -10 pts for 1 day of lateness / - 20 pts for 2 days of lateness / 0 after that, but you still have to turn in your work

---

Grading:

16 pts Student.java
18 pts StudentTester.java
61 pts Execute.java
5 pts Word file with answers to Q1 and Q2