ICSI201/ICEN141 Fall 2022 – Project 2

Students GPA

Due Sunday, November 6th, 2022, at 11:59 pm via Blackboard
Note, this is a firm deadline, and no extensions will be given (see syllabus for more detail).
(100 points)

ASSIGNMENT OVERVIEW

Your program should read an input file of the format <First Name, Last Name, GPA>, and allow the user to choose among five options: (1) find the lowest GPA, (2) find the highest GPA, (3) calculate an average GPA, (4) show all students' records, and (5) quit the program.
It is a multiple-file project that includes a driver file that runs the project, a class file that defines the structure of your data, and a text file that contains the data. In this project you will practice several programming concepts such as methods, files, arrays, and global variables.

Objectives:

- Build experience programming with methods.
- Build experience creating classes.
- Build experience working with files and arrays.
- Practice the concept of the variable's scope.

Reading:

Chapters 4 (section 10), 5, 6, and 7 are the absolute minimum of reading you need to complete to confidently tackle this assignment. You should also be comfortable with the skills you acquired in Labs 6-8 and zyLabs #4-#6.

GRADING GUIDELINES

The breakdown of points for this assignment is in the table below.
In addition to the correctness, some points count towards how well your code is written and documented. Well-written code uses proper indentation that follows the scope of your statements and employs intuitive and self-documenting variable names. To gain points for good documentation, your code should include comments that detail the purpose of blocks of code or individual statements. Good code/documentation does not imply that more is better. The goal is to be efficient, elegant, and succinct!

Grading item	**Points**
1. The software meets the specification: • Data records are provided in the text file • A class for data records is created as a separate file • An array is used to process the data • Methods are implemented	10
2. The software is well documented: • Required comments are included • Necessary comments are included • Javadoc format and tags used	10
3. The implementation is error free and well-tested: • There is no compilation error • The software produces no runtime error even if the input data is corrupted	20
4. The project is complete and works correctly • A data record from the file successfully converts into a new object • The program reads all data from the file, but not more than 100 records • The number of elements in the array counts correctly • The menu and methods are implemented in full • All statistics calculate correctly	50
5. The programming style is efficient: • Indentations are correct • Naming is descriptive • Methods are appropriate	10

~~Cheating policy:~~

~~This assignment is to be completed individually. Cheating is not tolerated.~~

~~We will be comparing your code against that of other students in the class and similar assignments from online student support platforms. All students involved in a cheating accident (i.e., whether sharing or receiving code) will be penalized. Please, read the syllabus for our policies on cheating and refer to slides from Lecture 1 for examples of what we consider cheating, as those are not only limited to what you submit. Students caught cheating will receive 0 points for the assignment and will be reported.~~

ASSIGNMENT TURN-IN

You must submit your work on Blackboard. There are three attempts, and you can use the first one as a draft, the second as the main submission, and the third as a backup.

Your project submission must have three files –

- A driver source code file named `DriverGPA.java`
- A class source code file named `Student.java` which implements a class for objects of type `Student` (i.e., represents student's name and GPA).
- A text file with sample data

IMPORTANT: no files with extension `.class` are accepted.

File naming conventions and instructions compliance:

We will auto-run your code and if your file and class naming conventions deviate from the above, you will lose points.

In your driver source code before the first line of the code you must place a brief description of the project and information about the author: **course**, **semester**, **first** and **last name**, **Net ID**, and UAlbany **email**.

ASSIGNMENT DETAILS

Step #1

Start the development process by planning your data and program organization. Students will be the main objects in your project, the corresponding class must have fields for first name, last name, and GPA. Decide how you will create a new object and how you will access its data. Create to `toString()` method for the class.
Consider a UML diagram before programming the class. This is an example of the UML diagram that you can use and modify if necessary.

```
Student
-firstName: String
-lastName: String
-valueGPA: double

+Student(fN: String, lN: String, vGPA: double):
+getFirstName( ): String
+getLastName( ): String
+getGPA( ): Double
+toString( ): String
```

To test your class, make the first draft of a driver program. For example, the program may request the user to input the student's info, create an object, and print the new object.

Step #2

Advance the driver by replacing the user's input with the input from the file. Consider the following file format – each student's record takes three lines in the text file:

- <First Name>
- <Last Name>
- <GPA>.

There is an example of the data file attached to the project description. You can use it for debugging but have to create and submit different data file.

You are free to use alternative file formats, e.g., each student's record takes one line in the file, then you must handle the parsing of the string or use a delimiter.

To create a text file according to the selected format you may use any text editor including one that is embedded in Eclipse to write the code. The file must have at least 10 records.

To read the text file, you can use either the `Scanner` class or the buffered stream (used in labs).

After this step, your program must be able to read the data file, create objects and echo them to the console.

Step #3

Because the data file contains a collection of repeating information, we can create an array with these data to manipulate the data efficiently. For this project, you should not process more than 100 data records in the file. If the number of records is greater than 100, the rest must be ignored.
Use an array to keep and process data in your project. Though the maximum size of the array is given, the actual number of records must be counted while file reading and used during the runtime. Because this value must be available for all methods in the driver, it may be declared as a global variable.[1]

Step #4

The last step for this project is creating a menu that provides the following options: find the lowest GPA, find the highest GPA, calculate an average GPA, show all students' records, and quit the program.

[1] Note that it is good practice to avoid global variables unless necessary. ~~`ArrayList`~~ can be an alternative solution for this project, it may be less efficient and is not allowed by the project spec.

The program must run constantly until the user chooses to quit. All menu options except quitting must be implemented as methods. You can structure your program with other methods, but these are mandatory. The average GPA must be shown in two decimal digits.

The example below uses letters as a user's choice. You can use numbers as we did in labs.

Program outline:

1. Read file data into the array of objects
2. Output menu
3. Execute the user's choice until the user decides to quit.

Recommendations:

- Read the description carefully. If necessary, clarify requirements with the instructor or TA.
- Ask yourself what your program's input and output are. Then, think of how you can divide the problem into smaller tasks, and plan to create methods for each task.
- Use Piazza to discuss the problem with your peers (remember the **acceptable level of collaboration**).
- Develop the test plan for your project, i.e., examples of inputs and expected outputs. Remember about critical points. Though the test plan is not required for your submission, it will be appreciated and may add some points to your programming style.
- Use the instructors' and TAs' office hours to get help if you are stuck.

EXAMPLES OF PROGRAM EXECUTION

```
This program processes the file with students' GPAs.



The cap letter corresponds to the option. What is your choice?
        Mi(N) GPA
        Ma(X) GPA
        (A)verage GPA
        (P)rint all
        (Q)uit
P

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The full roster:

Guybrush Threepwood 2.0

Gordon Freeman 2.8

Samus Aran 2.4

Leon Kennedy 2.6

John Marston 3.8

BJ Blazkowicz 3.0

Ezio Auditore 3.2

Lara Croft 4.0

Nathan Drake 3.6
```

```
Venom Snake 2.5
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

The cap letter corresponds to the option. What is your choice?
        Mi(N) GPA
        Ma(X) GPA
        (A)verage GPA
        (P)rint all
        (Q)uit
N
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The lowest GPA is Guybrush Threepwood 2.0
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

The cap letter corresponds to the option. What is your choice?
        Mi(N) GPA
        Ma(X) GPA
        (A)verage GPA
        (P)rint all
        (Q)uit
A
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The average GPA is 2.99
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

The cap letter corresponds to the option. What is your choice?
        Mi(N) GPA
        Ma(X) GPA
        (A)verage GPA
        (P)rint all
        (Q)uit
Q
```