CSE1322L Assignment 7 - Spring 2023

Intro:

This week's assignment will allow you to test your new Thread and Data Structure skills. You'll be using the built in LinkedList class in Java and C#, you will not have to implement a linked list class from scratch.

This program will demonstrate a real world application for threads. Often the front end of the system (the part that the user interacts with) will run in one or more threads, while the backend (that's doing calculations etc) runs in another.

In this case you are going to write a program that reads in a file of grades for students, it'll create a gradebook, however, the file doesn't include quiz averages, or homework averages. Both of them need to be calculated. You could do this as you read in the file, but that slows down when the user is first able to utilize your application, so the solution is that you have one thread (the main one) which interacts with the user, while another thread calculates the averages in the background. Ideally by the time the user asks for info it's already calculated.

Tasks:

1. Create a class called Student. It must have;
    a. Attributes
        i. An array of 10 quiz scores as integers.
        ii. An array of 10 homework scores as integers
        iii. A midterm grade stored as an int
        iv. A final exam grade stored as an int
        v. A quiz average stored as a double
        vi. A homework average stored as a double
        vii. An overall average stored as a double.
        viii. A name stored as a string
        ix. An ID stored as a int
    b. All attributes must have the appropriate permissions.
    c. Write a constructor which takes in a string (one line from the file).
        i. The string passed in is a comma separated string. Split it into many separate parts by splitting on a comma.
        ii. The first cell of the array holds the student's name. Store it in the object variable.

      iii. The second cell holds the student ID. Store it in the id attribute of the object.
      iv. The next 10 cells hold quiz grades, Quiz 1 - Quiz 10. Add them to the quiz array.
      v. The next 10 cells hold homework grades. HW1 - Hw10. Add them to the homework array.
      vi. The next cell holds the midterm exam grade. Store it in the midterm attribute
      vii. The last cell holds the final exam grade. Store it in the final exam attribute
   d. Create getters for name, and ID
   e. Write a method calcQuizAverage(). It should take no parameters and return void.
      i. Calculate the Quiz average. Much like this class, the lowest quiz grade is dropped. Average the remaining 9 grades
   f. Write a method calcHWAverage(). It should take no parameters and return void.
      i. Calculate the HW average. Again, we'll drop the lowest hw grade, and average the remaining 9.
   g. Write a method calcOverallAverage().
      i. It will set the Overall Average attribute.
      ii. The overall average is made of 40% is the quiz average, 10% is the homework average, 20% is the midterm exam, and 30% is the final exam. There are no substitutions.
   h. Write a method getGrade()
      i. It should return a string with all the values from the attributes. See output below for an example of how it should be formatted.

2. Write a class called GradeBook.
   a. It must have a linked list of students.
   b. It must have a constructor that takes in a filename as a string.
      i. It should open the file
      ii. Read in a line from the file. Create a new Student object by passing it the line from the file.
      iii. Add the new Student object to your linked list.
      iv. Deal with any exceptions.
   c. Write a method getStudent which takes in a student name (string) and returns a Student.
      i. Search the linked list for a student with that name. Return that student

   d. Write a method getStudentGrade. It should take in a student name as a string and return void.
      i. Search the linked list for a student with that name. Print students' grades.
   e. Write a method getAllStudentNames which takes in no parameters.
      i. It must return a linked list of strings
      ii. Get the name out of each student in the linked list of students, add it to the linked list of strings and return it.
3. Write a class StatisticGradeBook which is a child of Gradebook. It will be used as a Thread.
   a. If you are a java student, implement Runnable. If you are C#, you don't have to do anything special.
   b. It must have a constructor that takes in a string and passes the string to its parent's constructor. This is a filename of grades.
   c. Write a method called run (Java) or anything you want (C#) which takes no parameters and returns null.
      i. Get all the student's names and store them in a linked list of strings
      ii. Figure out how many students you have by finding the length of the linked list.
      iii. Keep count of how many students you've processed.
      iv. Iterate over all the Students in the Student LinkedList.
         1. Call calcQuizAverage() on each student
         2. Call calcHWAverage() on each student
         3. Call calcOverallAverage() on each student
         4. If this is the 100th, 200th, 300th…900th student print out "Calculating grades 100 out of 1000"
         5. When you are done with all students print "All grades calculated"
4. In your main method:
   a. Instantiate a StatisticGradeBook object
   b. Make a Thread of it
   c. Start the Thread
   d. Ask the user for a students name
   e. Call getStudentGrade on that student

Note there is no loop here, you only get to ask about one student, then the program ends.

Sample Output:

[Note since there are 2 threads here the question "What student would you like to see grades for” can happen before, during or after the statements telling you the grades have been calculated.]

First Run:

```
What student would you like to see grades for
Calculating grades 0 out of 1000
Calculating grades 100 out of 1000
Calculating grades 200 out of 1000
Calculating grades 300 out of 1000
Calculating grades 400 out of 1000
Calculating grades 500 out of 1000
Calculating grades 600 out of 1000
Calculating grades 700 out of 1000
Calculating grades 800 out of 1000
Calculating grades 900 out of 1000
All grades calculated
Bob
Quiz 0: 54
Quiz 1: 0
Quiz 2: 63
Quiz 3: 53
Quiz 4: 71
Quiz 5: 27
Quiz 6: 13
Quiz 7: 40
Quiz 8: 81
Quiz 9: 92
 Quiz Avg: 54.888888888888886Hw 0: 89
Hw 1: 4
Hw 2: 52
Hw 3: 86
Hw 4: 9
Hw 5: 82
Hw 6: 45
Hw 7: 34
Hw 8: 68
Hw 9: 31
HW Avg: 55.11111068725586Midterm: 84 Final Exam: 73Overall Average:
66.16666662428113
```

Second Run:

```
What student would you like to see grades for
Calculating grades 0 out of 1000
Calculating grades 100 out of 1000
Calculating grades 200 out of 1000
Calculating grades 300 out of 1000
Calculating grades 400 out of 1000
Calculating grades 500 out of 1000
Calculating grades 600 out of 1000
Calculating grades 700 out of 1000
Calculating grades 800 out of 1000
Calculating grades 900 out of 1000
All grades calculated
Aaliyah
Quiz 0: 9
Quiz 1: 42
Quiz 2: 87
Quiz 3: 88
Quiz 4: 87
Quiz 5: 80
Quiz 6: 24
Quiz 7: 45
Quiz 8: 50
Quiz 9: 46
 Quiz Avg: 61.0Hw 0: 98
Hw 1: 5
Hw 2: 50
Hw 3: 70
Hw 4: 56
Hw 5: 58
Hw 6: 73
Hw 7: 37
Hw 8: 48
Hw 9: 90
HW Avg: 64.44444274902344Midterm: 29 Final Exam: 19Overall Average:
42.34444427490235
```

Submission Guidelines:

You'll submit four classes (Student, GradeBook, StatisticGradeBook and your main class) on gradescope as Assignment 7.

Please follow the posted submission guidelines here:
https://ccse.kennesaw.edu/fye/submissionguidelines.php

Ensure you submit before the deadline listed on the lab schedule for CSE1322L here:
https://ccse.kennesaw.edu/fye/courseschedules.php