EE441 HOMEWORK 1

Classroom Interface

Due: November 13, 2017, 23:55

Ferhat Gölbol	**D-227**	**ferhatg@metu.edu.tr**
Kamil Sert	**D-215**	**ksert@metu.edu.tr**
Aycan Doğa Hakyemez	**ARC-201**	**hdoga@metu.edu.tr**

***You may ask HW#1 related questions to hdoga@metu.edu.tr**

Introduction

You are asked to implement a user interface and the corresponding database of students for a single section course. The database entries will be instances of a custom-made Student class. The database itself will be an instance of a Course class. Student and Course classes and the user interface will be designed and implemented in this homework.

Student Class

The class will have the following contents:

- Private members:
  - ID
  - name
  - surname
  - mt1_score (First midterm score)
  - mt2_score (Second midterm score)
  - final_score (Final score)
- Public methods:
  - Constructor
  - Getter and setter methods for **all** members (e.g. getID, setName, etc.)
  - overallScore (calculates overall score of a student considering exam scores and exam weights)

Course Class

- Private members:
  - entries[MAX_SIZE] (an array for Student instances, MAX_SIZE is 10)
  - num (the current number of students in the course)
  - exam weights:
    - weight_mt1 (default 25%)
    - weight_mt2 (default 25%)
    - weight_final (default 50%)
- Public methods:
  - Constructor
  - getNum

- addStudent
- changeWeights
- getStudent (return a Student given an index)
- calcAverage (calculates course averages for MT1, MT2, final and overall score)

User Interface

- Functions:
  - showByID (show all information of a student given their ID)
  - showAbove (show all student information above a certain overall score threshold)
  - showBelow (show all student information below a certain overall score threshold)
  - showAverage (show classroom average for each exam and overall score)
  - updateStudentScore (change a student score given their ID and exam name)

The program should ask the user for his/her input by offering different options as a menu:

- Add a student
- Search a student by ID
- Show students with overall score above a threshold
- Show students with overall score below a threshold
- Show classroom average
- Change a student's score
- Exit

Remarks

- You should check for illogical (e.g. negative age) and invalid (e.g. invalid ID, insufficient number of inputs) cases, and give appropriate warnings.
- If necessary, you can implement additional methods and functions (but **not** class data members).

Example

```
Classroom information interface

Choose your option:
1) Add a student
2) Search a student by ID
3) Show students with overall score above a threshold
4) Show students with overall score below a threshold
5) Show classroom average
6) Change a student's score
7) Exit

Enter your option: 1
Enter ID, name, surname and exam scores (MT1, MT2, final):
123 Alice Evans 65 28 71

Enter your option: 1
Enter ID, name, surname and exam scores (MT1, MT2, final):
265 Bob Crane 10 6 42

Enter your option: 1
Enter ID, name, surname and exam scores (MT1, MT2, final):
1602 Charlie Doe 100 51 22
```

```
Enter your option: 5
MT1 average: 58.3333
MT2 average: 28.3333
Final average: 45
Overall average: 44.1667

Enter your option: 2
Enter ID: 122
Invalid ID

Enter your option: 2
Enter ID: 123
  123     Alice     Evans        65        28        71     58.75

Enter your option: 4
Enter maximum score: 50
  265       Bob     Crane        10         6        42        25
 1602   Charlie       Doe       100        51        22     48.75

Enter your option: 6
Enter ID, exam type and updated score: 265 mt2 20

Enter your option: 5
MT1 average: 58.3333
MT2 average: 33
Final average: 45
Overall average: 45.3333

Enter your option: 8
Invalid option

Enter your option: 7

Process returned 0 (0x0)   execution time : 67.974 s
Press any key to continue.
```

Submission

- Use Code::Blocks IDE and choose GNU GCC Compiler while creating your project. Name your project as "e1XXXXXX_HW1" where Xs are the digits of your student ID number. Send the whole project folder compressed in a rar or zip file. Name your submission as e1XXXXXX_ee441_hw1.rar. You will not get full credit if you fail to submit your project folder as required.
- Your C++ program should follow object oriented principles, including proper class and method usage and should be correctly structured including private and public components. Your work will be graded on its correctness, efficiency and clarity as a whole.
- You should insert comments in your source code at appropriate places without including any unnecessary detail.
- Late submissions are welcome, but are penalized according to the following policy:
    - 1 day late submission: HW will be evaluated out of 70.
    - 2 days late submission: HW will be evaluated out of 50.
    - 3 days late submission: HW will be evaluated out of 30.
    - 4 or more days late submission: HW will NOT be evaluated.
- It is **not** allowed to prepare homework as groups. METU honor code is essential.
- Any involvement in any cheating will be graded 0.