CST8288 OOP with Design Patterns

Assignment 2

Overview

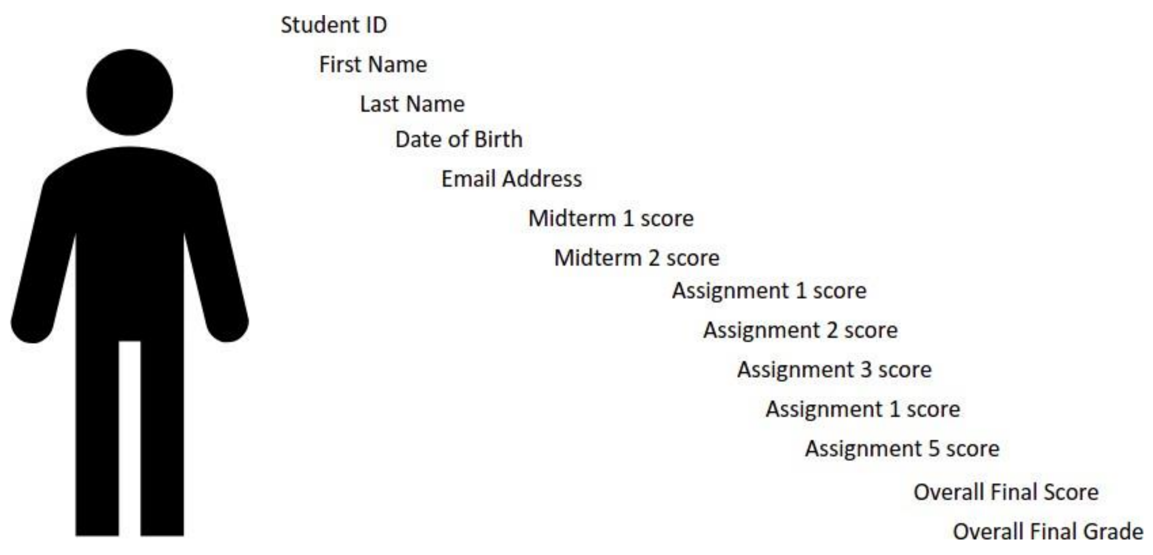
Algonquin college is developing a new course “Ethics AI”. The course instructor needs a CLI (Command Line Interface) application that allows them to manage students' information and grades throughout the semester. They required that the application keeps the student’s data in SQL database. The instructor sent us a description of what they expect from the application, which is discussed in the coming sections, but they made it clear that they will not be able to sit down with the application developer to train them on how to use the application, so they expect a self-intuitive, user-friendly application. However, they are not too picky with the UI, they are satisfied as long as they are able to perform the desired essential operations on the students’ database. The design requirements and specifications of the application are illustrated in the following section of this assignment.

The instructor provided us with the fourteen (14) properties/fields/information that they want to manage for their students. They had also provided us with the five (5) essential operations that they would use the application for. However, on a side-talk with the instructor, they mentioned a few extra functionalities that they find useful, those extra functionalities are also illustrated on the next sections of the assignment.

They also mentioned that the course consists of two (2) Midterms (25% each), and five (5) Assignments (10% each).

Design Requirements

Each student in the database will have a set of properties/fields shown in the figure below, followed by the design requirements and restrictions applied on the fourteen (14) properties.



1. Student ID: This will be the **Primary key** of the database. Each student must have a unique ID. Ideally. You want Java to auto-generate a sequence of unique numbers for you, so that the instructor doesn't have to worry about finding an unused number whenever they need to add a new student to the database.

2. First name & Last name: No restrictions apply to these fields.

3. Email: Auto generate the email using the students first and last names and their Date ofBirth. The auto-generated email consists of the first name, month of birth and last name, consecutively, followed by "@algomail.com"

   Example: John Doe was born on 24/April/1999, his auto-generated email isjohn04doe@algomail.com.

4. Scores: Each score is out of 100.

   Example: John Doe's result in Midterm 1 was 78/100, the number that appears on John Doe's row in the database under the "Midterm 1 score" column is 78. (not 78/100)

5. Calculate the Overall Final Score using each student’s scores in all midterms and assignments. Thus, saving the instructor the time spent calculating final grades for each student one by one.
6. Calculate the Overall Final Score using each student’s scores in all midterms and assignments based on Algonquin college grade scheme(https://www.algonquincollege.com/policies/files/2017/06/AA14.pdf). Thus, saving the instructor time spent calculating final grades for each student one by one.

   Example: The Overall Final Score of the student John Doe is 78. The application must automatically populate the “Overall Final Grade” field by “B+”.

The instructor will typically be using your application to perform the following five (5) essential operations:

- Add a new student
- List all students
- Fetch a single student by ID
- Change a single student information by ID
- Remove a single student from the database by ID

The instructor mentioned on a side-talk, that they usually run the application once a day, and perform all their desired operations one after another. Hence, it would be convenient of the application to return to the main menu after every operation performed rather than quitting.

By the end of the semester, the instructor would like to view a few statistics that might help them assess and improve the course. They demanded a “Stats Summary” in the application that summaries some relevant statistics mentioned below. Feel free to present the desired stats to the instructor in any way you want. Always make an easy to follow, user-friendly application your priority.

Note: The instructor seemed open to a simple single menu that displays all the required stats all at once.

7. For each deliverable in the course (midterms & assignments), calculate the class's average score for that deliverable.

8. For each deliverable in the course (midterms & assignments), find out how many students scored above the class's average, and how many students scored below that average.

9. For the Overall Final Score of the students, calculate the average Overall Final Score ofthe class and the highest and lowest Overall Final Score. Also, find out how many students passed (scored above 50) and how many failed the class.

The instructor did not mention any preferences regarding the design pattern of the application. However, for the demo of your application, you are required to very briefly illustrate your design pattern and why you chose it.

Your project should use "**defensive programming**". It means that operations from user should be evaluated (when passing parameters - ex: invalid number format or negative values for grades - and operations should be safe - ex: if there is no student "john04doe", search / deletion are supposed to print an error message.

Demo scenarios

For the demo of your application, come with a running application that is connected to a SQL database. Make sure that your database is prepopulated by the data shown below. During thedemo, you will be asked to perform all the five (5) essential operations on that database.

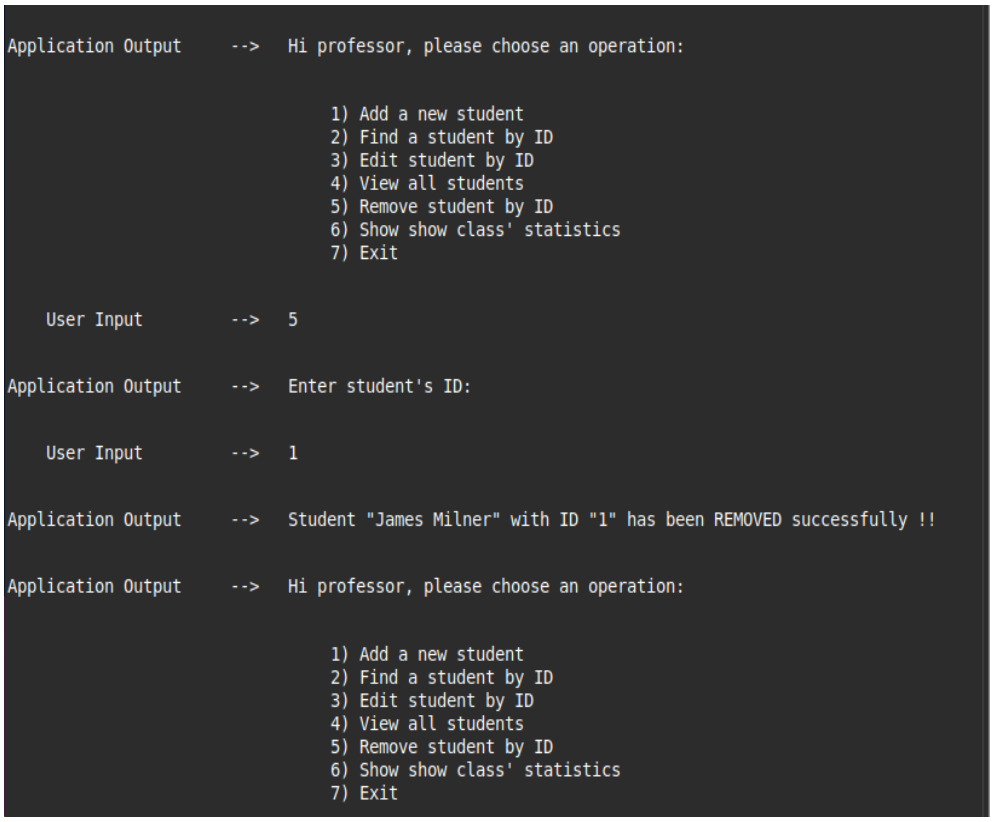
Example: Delete the student with

ID 6.Example: Find the student

with ID 3.

*** Check the Excel sheet uploaded on Brightspace ***

To give a general example of what is expected from this application, following is an example of the instructor performing one of the five (5) essential operations on the database. However, feel free to structure your application and UI in the way you prefer as long as the UI is user- friendly, this is just to give you a hint.



```
Application Output     -->   Hi professor, please choose an operation:


                                1) Add a new student
                                2) Find a student by ID
                                3) Edit student by ID
                                4) View all students
                                5) Remove student by ID
                                6) Show show class' statistics
                                7) Exit


    User Input         -->   5


Application Output     -->   Enter student's ID:


    User Input         -->   1


Application Output     -->   Student "James Milner" with ID "1" has been REMOVED successfully !!


Application Output     -->   Hi professor, please choose an operation:


                                1) Add a new student
                                2) Find a student by ID
                                3) Edit student by ID
                                4) View all students
                                5) Remove student by ID
                                6) Show show class' statistics
                                7) Exit
```

Grading

1. Use of design patterns (**20** points)
2. UML class diagram of your application (**10** points)
3. Correctness of the basic functionality (**60** points)
    1. Add a new student implemented correctly
    2. List all students implemented correctly
    3. Fetch a single student by ID implemented correctly
    4. Change a single student information by ID implemented correctly
    5. Remove a single student from the database by ID implemented correctly
    6. Show class stats.

4. Applying Java **Best Practices**, **Naming Conventions and providing meaningful definition and purpose for each method.** (**5** points)

5. User-friendliness of your application. (**5** points)

Approach

Break down your work in smaller chunks and assign them to each member.
Set deadlines for yourselves so you all can be on the same page.

DO NOT LEAVE THESE FOR THE LAST WEEK, YOU WILL NOT FINISH.

IF YOU ARE UNSURE ABOUT ANYTHING ASK, DO NOT ASSUME ANYTHING.

Suggested Weekly Schedule

Week 10 – Complete your design
Week 11 – Implementation
Week 12 – Implementation
Week 13 – Testing and documentation. Double check your work and submission

Due Date

Sunday Nov. 5th Midnight

There will **not** be any **extension** as it is the **end** of the **term**.

Penalty

-10% penalty for every day late, zero on the 5th day.