Project 2 – Classes and Top-down Design

Overview

In software development projects, it is important to be able to identify objects and classes in a project description. Equally important is the ability to follow a systematic approach to creating a solution to the problem. In this project, you are asked to utilize the design skills discussed in class to develop a solution to a programming problem.

Learning Objectives

The focus of this assignment is on the following learning objectives:

- Be able to identify class and objects from a problem description
- Be able to identify class functionality
- Be able to identify an incremental approach to developing the program solution
- Be able to implement the program solution based on the identified approach
- Understand how the program solution can evolve over time

Prerequisites

To complete this project, you need to make sure that you have the following:

- C++ and the g++ compiler
- A C++ IDE or text editor (multiple editors exist for developers)
- An understanding of the material presented in class.
- An understanding of the material covered in ZyBooks.

Problem Description

You are to implement a program for creating a single course gradebook. For this project, you will need to utilize the concepts of classes and top-down design that has been discussed in class. The list of requirements and constraints for the system are as follows:

1. The system must be able to manage multiple students (max of 5) and their grades for assignments from three different assignment groups: homework (total of 5), quizzes (total of 4), and exams (total of 2).
2. For each student record, the user should be able to change the grade for any assignment. These grades should be accessible to the gradebook for calculations, but not visible or directly mutable outside of the student record.
3. The gradebook should know the name of the course. It should also know the weights of each assignment group (which should add up to 1.0 when changed) and be able to print a report to the screen. This report should contain the course name and a well-formatted table containing columns for the student names, and for each assignment group. The group weights should be displayed (as a percentage) next to the name of each assignment group in the table's header. It should be possible to add and remove students from the gradebook. Removals from the gradebook should shift each successive student "up" in the list. For example, if student 3 is removed from the gradebook, the remaining students starting with

student 4 should be moved up a slot (i.e. student 4 to student 3's old slot, student 5 to student 4's old slot, etc.)

4. A user interface should be created with a menu system used to navigate the functionality necessary to complete the stated requirements. It is recommended to use a simple numbering scheme when prompting the user for their desired action. See the end of this project sheet for a reference you should use for the menus.
5. All names (student names, gradebook name) should be set to "<noname>" prior to being set by the user. This could also be used in logic to determine whether or not a student entry in the gradebook is "valid". Empty student slots should not be displayed in menus that require the user to identify a student record to modify.
6. The gradebook should be able to calculate class averages from the student records. The averages are the sum of the valid grades divided by the total number of valid grades for that group (i.e. ungraded assignments are not part of the total number of valid grades).
    a. Since each student record could contain a different number of grades for each category, the class average should be computed as the average of each student's average for each assignment group as well as their final grade. For example, if one student if missing a single homework grade (note, not a zero), then their homework average should be comprised of 4 grades instead of 5.
    b. When computing the class average homework grade, use each student's average homework grade for the running total, then use that total to compute the average. Use the same method for the other categories.
    c. When computing the average final grade, don't forget to use the group weights.
7. See the grading breakdown for additional requirements.

Development Diary

For this project, you must complete the following tasks in the order that they are described:

1. From the problem description, create a list of all classes that you can identify. For each class, list the associated member variables and identify an initial set of member functions.
2. List out a set of steps that you will take to implement your solution to the problem. Each step refers to an increment of the program that you will be creating. It is recommended to complete the implementation of a single logical action per step.
3. Once you have finished implementing your solution, reflect on the process that you followed. Did you wind up with the same classes as you initially identified? Did you need to change any of the functionality or add unexpected details? Did you have to deviate from your plan? Write a description of any details that needed to change as you worked on your solution.

Your responses to these tasks should be submitted as a PDF document called lastname_firstname_project02.pdf.

Grading Breakdown

- **[10 pts]** Working menu system to take in user input and call functionality
- **[6 pts]** Add a student to the gradebook (up to max)
- **[6 pts]** Remove a student from the gradebook (should move remaining students "up")
- **[6 pts]** Change a homework grade for a student (between 0-100, for each up to total)
- **[6 pts]** Change a quiz grade for a student (between 0-100, for each up to total)
- **[6 pts]** Change an exam grade for a student (between 0-100, for each up to total)
- **[6 pts]** Change gradebook weights for each assignment group (should total 1.0)
- **[4 pts]** Change gradebook name
- **[10 pts]** Display contents of gradebook as specified: course name / group weights / student count, followed by each student's name and individual grades, homework / quiz / exam average, and final grades using group weights
- **[7 pts]** Display class averages for each assignment group and the final grade using group weights
- **[15 pts]** Sufficient and clear class divisions (with main() responsible for I/O only)
- **[8 pts]** Clear, easy-to-read code (e.g. class declarations in .h files, class definitions in .cpp files, working makefile, etc.)
- **[10 pts]** Development "diary" (report) that details design and implementation of each class, as well as the project as a whole. Should include a reflection of the process as a whole upon completion of the project (identify the challenges, lessons learned, ways to improve in the future, etc.)

Submission

Points will be deducted for not following these instructions. Before submitting this project in eLearning make sure that you follow the following steps:

1. Make sure that your name appears at the top of each file. This should be done as a comment for any source code files.
2. Put your development diary document (.pdf), source code (.h & .cpp), and working makefile into a .zip file with the file name "lastname_firstname_project02.zip".

Turn your zipped up project into eLearning.

Sample UI

```
-=|  MAIN MENU  |=-
1. Add a student
2. Remove a student
3. Change a student's grade
4. Change group weights
5. Change the gradebook name
6. Display class averages
7. Display full report
0. QUIT

Enter an action: 1
```

```
-=|  ADDING STUDENT  |=-
Please enter the student's name: Name Here
Name Here was successfully added to the gradebook!
```

```
-=|  ADDING STUDENT  |=-
Students cannot be added because the gradebook is full!
```

```
-=|  REMOVING STUDENT  |=-
1. John Doe
2. Jane Doe
3. Third Person

Enter student to remove: 1
"John Doe" has been successfully removed! (New class size: 2)
```

```
-=|  REMOVING STUDENT  |=-
Cannot remove students because the gradebook is empty!
```

```
-=|  CHANGING GRADE  |=-
1. Change a homework grade
2. Change a quiz grade
3. Change an exam grade

What type of grade would you like to change: 2


1. John Doe
2. Jane Doe
3. Third Person
Which student's grade would you like to change? 3

-=|  CHANGING Third Person's QUIZ GRADE  |=-
1. 68
2. <ungraded>
3. 97
4. <ungraded>
5. <ungraded>

Which quiz grade would you like to change: 2
What would you like to change it to (-1 for ungraded): 89
Third Person's quiz grade 2 was changed from <ungraded>
to 89!
```

```
-=|  CHANGING WEIGHTS  |=-
Enter the weights, separated by spaces, in the order of
homework, quizzes, and exams (total must add up to 1.0):
0.15 0.25 0.7
Weights do not add up to 1.0, try again...


Enter the weights, separated by spaces, in the order of
homework, quizzes, and exams (total must add up to 1.0):
0.15 0.25 0.6
Weights updated successfully!
```

```
-=|  CHANGING NAME  |=-
Please enter the new name for the gradebook: COP3014


Gradebook name changed from “<noname>” to “COP3014”
```

```
-=|  CLASS AVERAGES  |=-
Homework average of class: 85.13%
Quiz average of class: 81.57%
Exam average of class: 77.40%
Final average of class: 79.60%
```

-=| GRADEBOOK REPORT |=-

Course: COP3014

Homework average of class: 85.72%

Quiz average of class: 81.57%

Exam average of class: 77.40%

Final average of class: 79.60%

Student	Homework (15%)	Quizzes (25%)	Exams (60%)	HW Avg	Quiz Avg	Exam Avg	Final Grade
John Doe	82 71 87 94 UG	. . .	. . .	83.50	. . .	. . .	. . .
Jane Doe	100 77 92 UG UG	. . .	. . .	89.67	. . .	. . .	. . .
Third Person	83 UG 85 UG UG	. . .	. . .	84.00	. . .	. . .	. . .