EECS 297/397 Special Topics: C/C++ Programming Spring 2020

Assignment 4: Classes

Instructor: Orhan Özgüner Due: March 16 before 11:59 PM

This assignment consists of four required components and one bonus. You will be constructing the framework of a student-management system for university classes (very similar to the in-class example of the car dealer management system). You are provided `main.cpp` which includes all of the object creations and function calls. When you open it, there will be errors because the classes it calls are missing. You will create classes and source/header files to fill in those errors. Your submission (including the bonus) should only include `student.cpp, student.h, instructor.cpp, instructor.h, course.cpp and course.h` files mentioned below.

Part 1

Create a Student class (`student.cpp` and `student.h`) with the following properties:

- Student ID
- Name
- Graduation year
- Major
- GPA (number that will normally take on a range between 0.0 and 4.0)

You should be able to construct new Student objects and fill in these properties as needed.

Part 2

Create an Instructor class (`instructor.cpp` and `instructor.h`) with the following properties:

- Instructor ID
- Name
- Department

You should be able to construct new Instructor objects and fill in these properties as needed.

Part 3

Create a Course class (`course.cpp` and `course.h`) with the following properties:

- Title (i.e. "Special Topics - C/C++ Programming)
- Department (i.e. "EECS")

- Number (i.e. 297)
- Building (i.e. "Bingham")
- Room Number (i.e."140")
- Instructor name (Each course should have a single Instructor. If a new Instructor is added it should overwrite the old one.).

You should be able to construct new Course objects and fill in these properties as needed.

Part 4

Add functionality to the Course class to be able to add and remove Students and add Instructor dynamically. That is to say, Course should implement the following methods:

- `void addStudents(std::vector<Student> students_to_add)`
  If a Student is already in the Course and is added a second time, it shouldn't create a duplicate student in the Course but other students in the vector should be added normally (it shouldn't give up or throw an error just skipped the duplicate).
- `bool dropStudent(Student student_to_remove)`
  This should find the student it is supposed to drop based on ID number. If that student is not in the Course, it should return `false`. It should return `true` if the student is successfully dropped.
- `void addInstructor(Instructor instructor_to_add)`
  Adds the Instructor to the Course even if the Course has already an Instructor. This is basically overwriting the current Instructor.
- `void printStudents(void)`
  Prints all of the students registered into the class.

Part 5 (Bonus)

Implement a *capacity* for Courses, which are defined when the Course is constructed; there is a constructor which sets that member as the last argument in the extra credit portion of the main file. If someone tries to add more Students to a Course than its capacity allows, `addStudents` should not add the extra Students and return a vector of those it didn't add. If all students were added successfully it should return an empty vector.

Next, implement a capacity for Instructors. Instructors should be able to teach more than one Course, but not an arbitrary number of them. Instructor will need an additional "capacity" member that indicates the maximum number of classes it can teach; there is a constructor which sets that member as the last argument in the extra credit portion of the main file. If someone tries to add an Instructor to more Course than its capacity allows, addInstructor should return `false`.

You will need to change the return type of `addInstructor` to `bool` and `addStudents` to `std::vector<Student>`.

Code to test the extra credit portions is commented out at the bottom of the `main.cpp`. You just need to uncomment it to submit both the regular and extra credit portions of the assignment.