Lab 11 -- Observer

The observer design pattern creates a notification scheme between objects that contain data, and other objects that may be interested in monitoring that data and act accordingly. The primary consequence of using this pattern is that observers do not have to *continuously poll* the object they are monitoring to see if they have changed: they simply wait to be notified when the change occurs.

The important elements of this design pattern are:

|  |  |
| --- | --- |
| 1. **Subject:** This is the object that contains the data of interest to other objects.   The Subject may be represented as an interface that contains the method that observers call to register themselves with it. This method usually takes an observer as a parameter.  An object makes itself observable by implementing this interface. This object remembers all its registered observers. When its data changes, it notifies all its observers.   1. **Observer:**These are objects that are interested in the subject.   An Observer may be represented as an interface that contains the notify method that is called by the subject. This method may take other arguments that give more information about what changed.  Each object that wishes to observe the subject implements this interface. It implements the notify method to take its own action in response to the change in the subject. The observer usually keeps track of the object it is observing, so that when it is notified, it can call methods to get more information. |  |

The names of the interfaces and methods above are only examples: they are always customized to the specific problem.

The Problem: Requirements of the MSCS program

Each MSCS student has to fulfill several requirements throughout the program, to remain in good standing, to go on co-op and to graduate on time. For example, the cumulative GPA must be at 3.0 or above at any point in the program to be in good academic standing. A student is eligible to go on co-op at any time after taking at least 16 credits and maintaining a cumulative GPA of at least 3.0. A student must have a GPA of at least 3.0 in all core courses combined (CS5010, CS5800 and one of CS5500 and CS5600) in order to graduate. For ALIGN students, CS5004 substitutes CS5010 as a core course. It is desirable to maintain a record of a student and monitor these requirements, so that the student can take necessary action (such as retaking a core course, or improving GPA, etc.). The GPA is calculated using [a standard procedure (Links to an external site.)](https://registrar.northeastern.edu/article/how-to-calculate-your-gpa/) .

Why observer?

All of the above requirements can also be implemented by having the grade record monitor directly various criteria using its own data. Are there any advantages that the observer pattern provides in this application?

What to do

Start with [the provided "observer.zip" file](https://northeastern.instructure.com/courses/63372/files/8776412?wrap=1)[download](https://northeastern.instructure.com/courses/63372/files/8776412/download?download_frd=1)that contains the interfaces for the observer pattern designed for this problem. It also contains an enumeration for grades that you should use and a class called GradeRecord that represents a grade that a single student has earned. It should track the course for which a student has received a letter grade.

1. Write a class called GradeSubjectImpl which implements the GradeSubject interface. A GradeSubjectImpl only needs a default constructor.
2. Implement the checking of criteria described above by implementing three observers -- GoodStanding, CoopEligible, and GraduationEligible -- each of which determines whether or not the criteria isSatisfied for the provided grade record when it is notified of a change in the grade record. These observers should also rely only on the default constructor.