**CS 4322 – Coding Assignment 4**

You can work individually or in a group of two for this assignment.

**Problem**

Design an abstract generic custom observable class (*CustomObservable*) that:

1. Allows observers to be added and removed.
2. Has a *notify* method that (a) only passes an instance of the generic type to observers and does not pass a reference to observable, (b) cannot be called by client code, only subclasses.
3. Do one of these two:
4. (-15 points) Provides a parameter that can be specified in the constructor or changed at any time that indicates which observers should be notified when *notify* is called: all observers in the order they were added, or every other observer, or every third, *etc.* The default should be every observer.
5. Provides a mechanism using marker interfaces to implement Priority 1, 2, and 3 observers so that when notify is called so that only Priority 1 observers are notified, only Priority 1 or 2 are notified, or all three levels are notified. Thus, *CustomObservable* needs a mechanism where this can be set.

Design a generic custom observer interface (*CustomObserver*) to support the requirements above.

**Requirements:**

1. Provide a write-up explaining:
2. Your design which includes a class diagram(s).
3. What works and what doesn’t
4. Time Log(s)
5. Write the code including a tester

**Final Deliverables**

Zip (or Rar) the items described above into a file named: *ca4\_lastName1\_lastName2.zip* and submit on Blazeview. Please only submit once, under one person’s account if working in a group.

**Write-up**

After doing quite a bit of reading and learning about the pattern, we attempted to implement the Observable Pattern using a CustomObservable abstract class, a ConcreteObservable class that extends the CustomObservable abstract class, a CustomObserver interface with 3 CustomOberserver classes, 3 Priority marker interfaces, and an ObserverTester class.

In the tester class, we first instantiated a generalized ConcreteObserver object. Somewhere in there, we figured out that you cannot instantiate an abstract class, but that’s neither here nor there. We decided to use Strings as the type of data being passed around. Next, we instantiated four ConcreteOberserver classes, two with a Priority1 marker, one with a Priority2 marker, and one with a Priority3 marker. Finally, we used a setPriority() method in the ConcreteObservable object to set the priority marker of the observers we wished to update.

In essence, our program sets the priority marker, and then we call a method in the ConcreteObserable object called setObs() that passes a String. The setObs() method takes a generic type, and we used Strings. This method also contains the setChanged() method, which is a protected method in CustomObservable, and changes the value of the changed variable to “true.” The notifyObservers() method is then called within setObs() which the generic type “t” is passed. The notifyObservers() method in CustomObservable is overridden by the notifyObservers() method in the ConcreteObservable class.

At this point, the notifyObservers() method runs a check on what priority is set through the getPriority() method, as long as the setChanged() method has set “changed” to “true”, and determines which ArrayList<> to traverse based on the priority set. If priority “1” is set, priority1List is traversed in order to update the observers with a Priority1 marker. The same is done for “2” and “3”. It is also possible to include a “1, 2” in the original setPriority() method in the ObserverTester class, a “1, 3”, “2, 3” or “no priority” which will traverse the observers ArrayList<> that contains every observer.

The program does not currently have the functionality to remove observers from the ArrayLists. Our class diagram is included in the file ObservingDiagram.ucls.

**Time Log**

If working in a group, ONLY submit a single timelog if you worked face-to-face for the entire assignment; Otherwise submit one for each member (copy table below).

* **Delete empty rows, add if needed.**
* **Put the total time at bottom.**

|  |  |
| --- | --- |
| **Time (hrs)** | **Task** |
| 1 | Looked at Observer Pattern examples |
| 2 | Started initial Observer Pattern classes |
| 3 | Worked on ConcreteObservable and CustomObservable |
| 5 | Continued work on ConcreteObservable and CustomObservable, worked on ConcreteObserver1, 2, 3, added Priority Interfaces |
| 2 | Worked on notify methods for notifying observers in AL<>, Tester class |
| 1 | Class diagram and write-up |
| **14** | **Total Time (hrs)** |

**Grading Criteria**

|  |  |  |
| --- | --- | --- |
| **Worth** | **Criteria** | **Description** |
| 10 | 1 | UML is correct |
| 15 | 2 | CustomObservable & Observer are generic |
| 5 | 3 | *notify* method passes an instance of the generic type to observers and does not pass a reference to observable |
| 10 | 4 | Can skip observers when notifying |
| 5 | 5 | Clients can't access notify |
| 25 | 6 | Code is correct and runs |
| 15 | 7 | Meaningful test cases |
| 15 | 8 | Enhancements |