Bryan Edman

10482719

Homework 3

Many-to-Many Observer Pattern

Starting this assignment I wanted to do an observer pattern with multiple Subject implementations. Newspaper1 and Newspaper2. The using a typical observer pattern have an observer interface with multiple customers. To make subscriptions work both ways I wanted to add a third interface for an Intermediary Table where Each subject and each Observer could be added to either a 2d List or two lists or maybe a dictionary. Where each subject could be associated with multiple observers and each observer can be associated with multiple subjects. This quickly got out of hand for me and became jumbled and confusing. Here is where I got in the UML.

A close up of text on a white background

Description automatically generated

I couldn’t think of a super great way to easily and understandably keep track of the associations between Subjects and Observers. I’m not the best at designing using only UML then coding I’ve made the bad habit of coding as I design to see what works. This could work with a proper Table implementation. It would work something like a database entry. The hard part is then how do you notify only observers of one subject? It’s probably possible but this is what I see as a Con to this setup. Also if anything goes wrong with your Table or it doesn’t handle something correctly the whole system fails. So it adds complexity and room for more errors. The Pros of this approach are that new subjects can be added and any observer can subscribe to it. Also any new observer can subscribe to many subjects.

I kept thinking about a better way to do this and tried just making the Subject the Table. Then we can keep the same general form of a classic observer pattern. Because each Subject is an observer and each Observer is a subject if we treat them all as observers but give the Subject Table the ability to hold lists of these observers Linking the different types together. We can then use methods in the table that notify the observers of a given observer based on it’s ID. Or simply notify every observer. Instead of subscribing and unsubscribing we will add and delete observers out right by ID. Then when one wants to subscribe to another we use the linkSubjectObserver(SubjectObserver subOb1, SubjectObserver subOb2) method which will add both to a Dictionary< SubjectObserver, List<SubjectObserver>> so each observer is associated with a list of it’s observers. I believe this would work though I haven’t tried it in code yet.

The Cons of this method are searching through and manipulating the observers might be a challenge. Using Linq queries would probably make it much easier though. Also it may be hard for someone else to understand exactly how you are treating observers as Subjects and vice versa.

The Pros of this method are that we keep the classic structure of the observer pattern. Which people should know and understand. It treats a many to many relationship as a one to many, hopefully simplifying the overall complexity. This requires no new table interface for example.

Here is my second UML. There is probably more issues to iron out with this but I think It’s a good start to the problem.

A close up of text on a white background

Description automatically generated