National University of Computer and Emerging Sciences

Abeer Zehra (16k4068)

Section:8E

Design Defects And Refactoring

Assignment #01

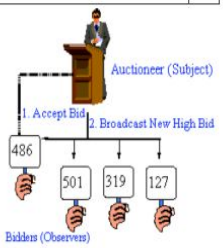
Assignment #1

Design Pattern, Spring 2020

Question 1: In this assignment, you need to describe your own example of any design pattern of your choice and give answers to all of the parts of the question 1. (Part A to Part D – see below) But your scenario / example must be a non-programming example explaining any real-life scenario, like the one given below.

Observer Example

The Observer defines a one to many relationships, so that when one object changes state, the others are notified and updated automatically. Some auctions demonstrate this pattern. Each bidder possesses a numbered paddle that is used to indicate a bid. The auctioneer starts the bidding, and "observes" when a paddle is raised to accept the bid. The acceptance of the bid changes the bid price, which is broadcast to all of the bidders in the form of a new bid.



Part (A) Give your own example (textual + graphic representation):

Describe your own example of any designs pattern that we have covered in the class and give a substitute case of the design pattern being utilized in some other situation. (as shown in the above example) It very well may be a regular day to day example (e.g., how online goods are shipped, how to leave a place of employment and join another organization, ... whatever). In a manner similar to above example, briefly describe your example with a diagram and a brief textual description.

Important Note: The model you pick ought not be utilized in the GoF book (also minor varieties of the given examples are not allowed) or one that was utilized in the lecture slides or notes, neither it should be taken from any web article.

Part (B) The problem, The Intent, The motivation and The Applicability

Clarify how your case "coordinates" the example's essential thought, that is, the way your model is closely resembling the structure example's model. Typically, you should clarify how your model synchronizes with (1) the intent and (2) the problem being discuss. From the GoF patterns, it should coordinate the aim or intent, inspiration and applicability.

Part (C) UML

Draw a class diagram and interaction diagrams for the model you picked. What you sketch will rely upon the pattern you decided for your model. Utilize standard notations of UML.

Part (D) Consequences

Document a sensible situation that features one of the consequences (disadvantages) of the design pattern. That is, your case study must be feature an outcome that is a downside in the situation. Iyou're your situation should highlight how the solution failed to address your requirements. Else you should portray a situation wherein neglecting to utilize the pattern would bring about negative outcomes. For example, in strategy pattern there is a communication overhead between Strategy and its Context, you'll need tighter coupling between Strategy and Context.

Question 2 – Pattern identification:

Consider each of the scenarios below and identify the design pattern which is most directly addresses the problem described. Briefly explain your reasoning.

Part (A): You've developed a new implementation of the List interface and you'd like to test the behavior of your new data structure. You've written an algorithm to perform your speed tests, but the algorithm needs to make a great many instances of your list class. You want to test the performance of your list against the performance of ArrayList and LinkedList, but you don't want to have to write your algorithm three times in order for it to be able to create the right kind of list to test.

Part (B) You've completed a compiler for a new language! There are many parts to your compilation process: parsing, transformation, assembly code generation, and so forth. You'd like to allow other programmers to use an interface to compile their code without resorting to system calls or other command-line invocations – your compiler can just run in their processes – but you don't want those users to have to know how to bring all of the steps of compilation together in order to use your compiler.

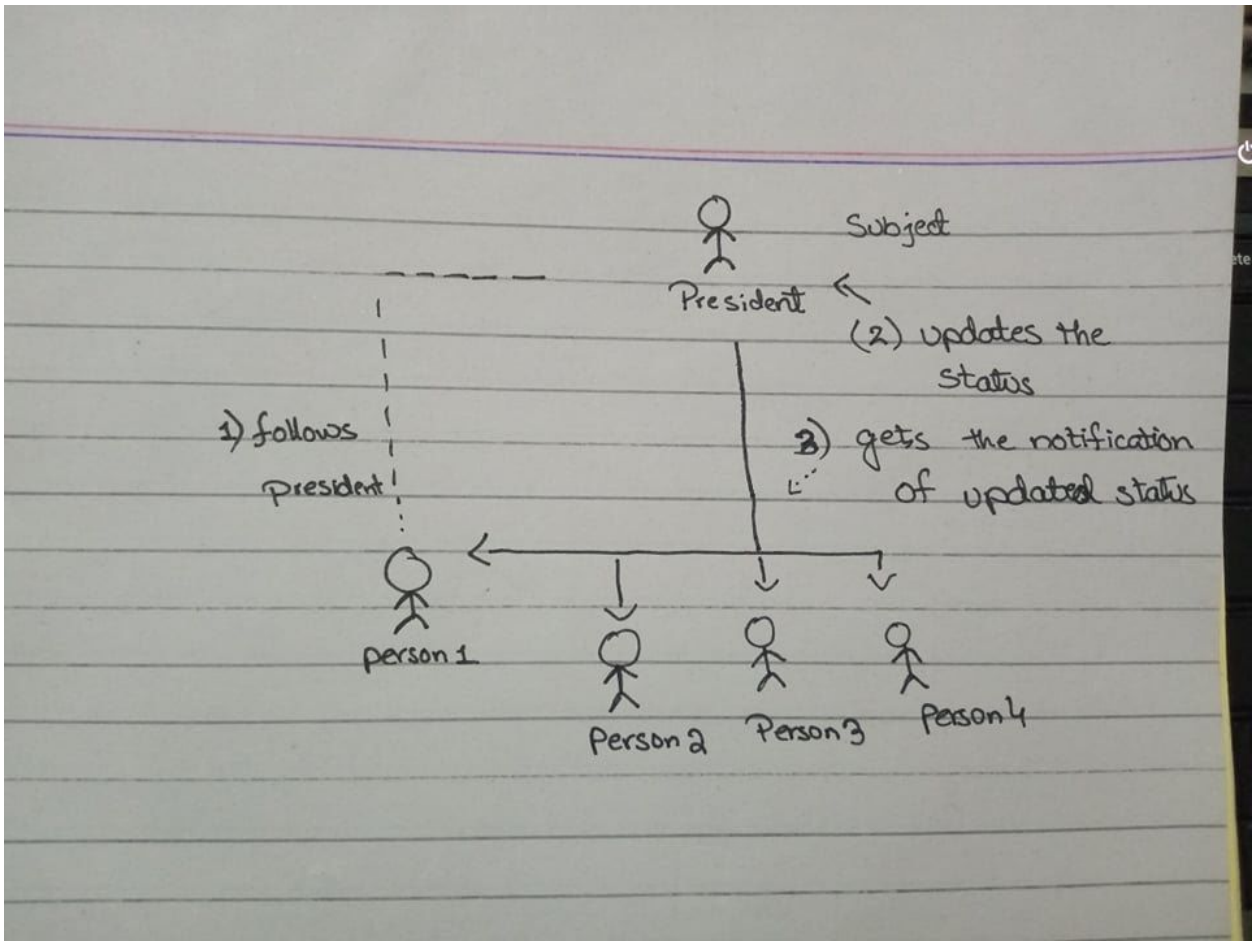
Question #01

Part (A) Give your own example

Observer Pattern!

Example :- The real world example of observer pattern is social media platforms. Like on twitter or facebook when one person updates a status all its followers gets the notification.

It's graphical representation is as follow :-



Part (B) The problem, The Intent, The motivation and The Applicability

Intent :- Define a one-to-many dependency between objects, so that when one object changes its states all the dependents are notified and updated automatically.

Problem :- A large monolithic design does not scale well as new graphing and monitoring requirements are levied.

Discussion :- Defined an object (an interface called “The Subject”) that is the keeper of the data model. Then delegated all the “view” functionality to the observer objects. Observers follow and unfollow by themselves. Whenever a subject updates a status, all it’s follower observers will get a notification about the update of the status.

This allows the number and type of view objects to be configured dynamically, instead of being statically specified at the compile time.

Part (C) UML

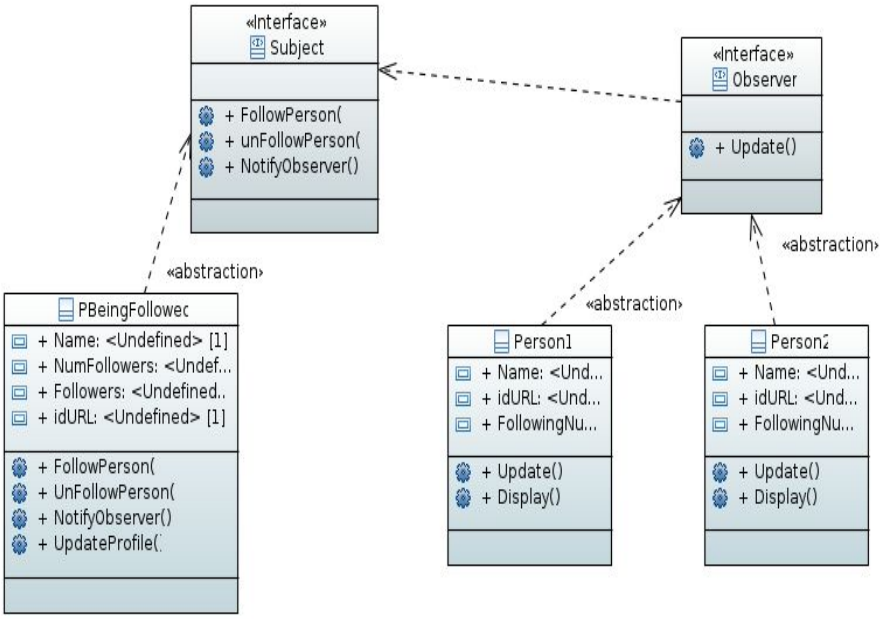


Fig. Class Diagram

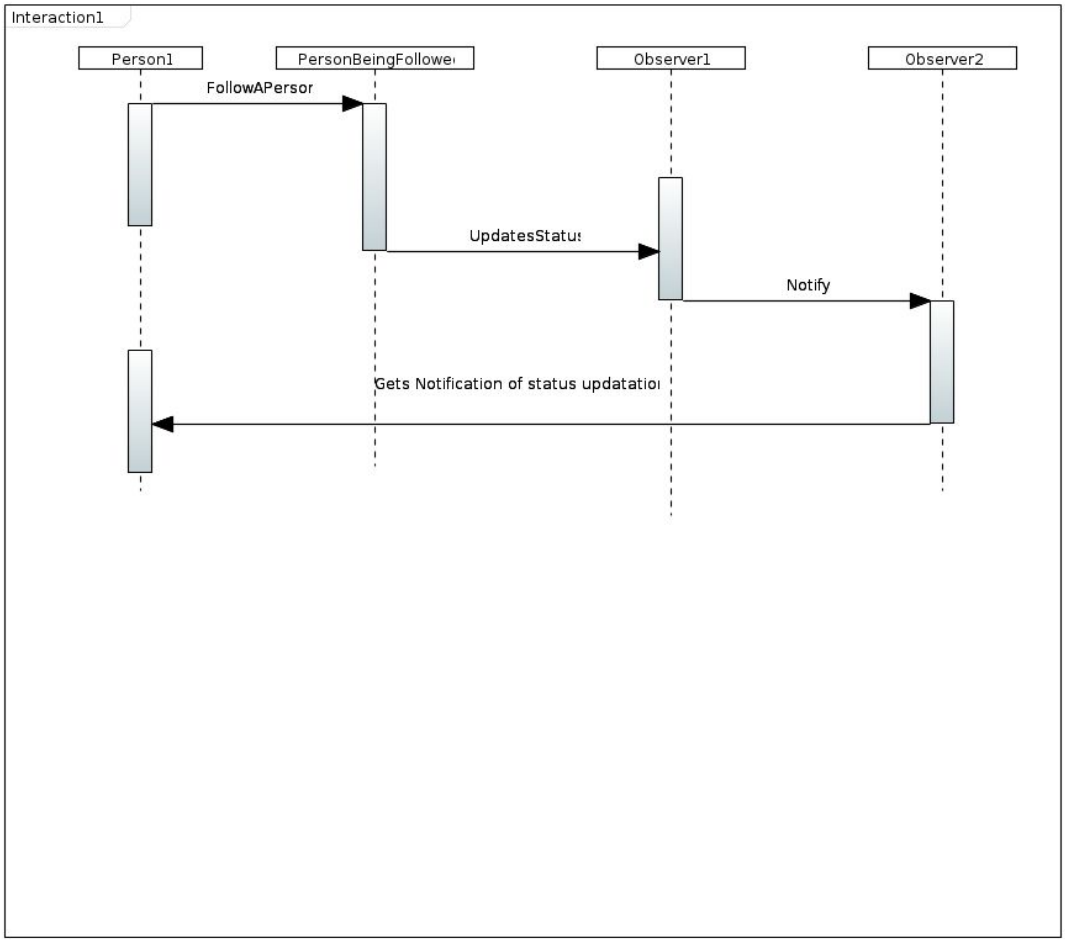


Fig:Sequence Diagram

Part (D) Consequences

Though observer pattern solves many issues that occur due to tight coupling.

But there are some disadvantages of using observer pattern. Firstly we need to have a concrete class to instantiate as it is using an interface which involves inheritance. There is no option for composition. If the observer pattern is not implemented correctly it can add complexity. Plus it is difficult to figure out why and where the observer is called. Most of the time, observer pattern calls different codes of concerns which should not happen.

Question #02

Part (A)

Adapter Pattern

Adapter pattern is used when we have to implement different categories that are closely related.

In this present scenario,we wants to test the performance of the list against the performance of ArrayList and LinkedList, meanwhile we want to write the algorithm one time in order for it to be able to create the right kind of list to test.For this, we can make an interface of list and then depending upon the scenario we can implement that interface. So, **Adapter Pattern** will be suitable for this whole scenario.

Part (B)

Facade Pattern

Facade pattern is used to hide the implementation details. It provides a simplified interface that is required to access the complex system.

Now in the given scenario we a new language compiler is implemented whose implementation details will be quite complex and we need to allow programmers to come and compile their code without resorting to system calls.Facade pattern will provide a simplified interface and will hide the compilation steps. Moreover, it also says that client will call compile function only and it will compile, so we need a simplified interface also, So **Facade Pattern** is the best suit.