By Hamid Gholizadeh (27 Jan 2014)
www.gholizadeh.net

Observer Patter

Observer patter is one of the responsibility patterns. When we would like to let other objects (observers) know about an event, we use this pattern. This event can be any kind of event. As a simple example imagine CDN to USD exchange rate changes event. Some people might be interested to know if the exchange rate changes more than 0.005 in my company. I become responsible for monitoring this change and notify people if that happened. I would provide a sheet and ask for people to write their name and phone number on that sheet, if they are interested to be notified when this event happens. I don't care who wrote on that sheet, I just take the phone and call and notify whoever registered on the sheet. It means that it is the subscription list that changes and my duty is not goanna change. In other words I am not supposed to be interrupted if anyone wants to subscribe, since the subscription list is exposed to the public and they can subscribe or unsubscribe. I will always take into account the last updated subscription list.

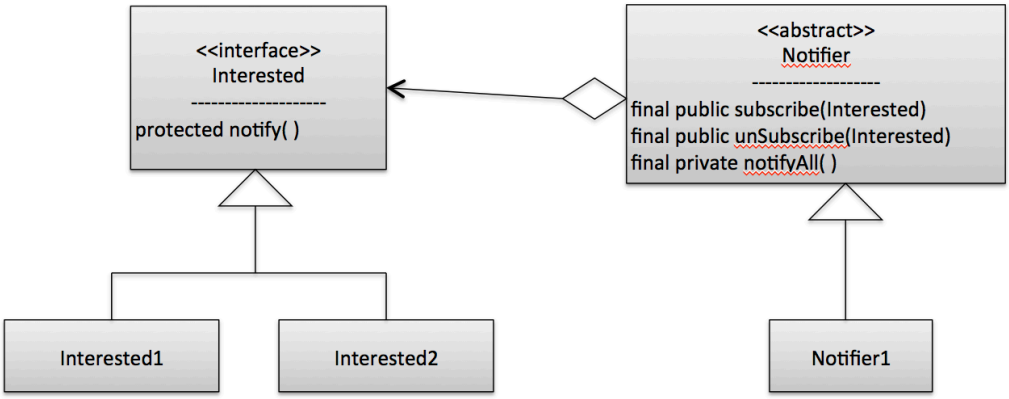

Fig1. Observer Pattern

As you see this pattern is even useful in everyday life. Below is the UML diagram for this pattern. Below is the observer pattern shown in UML diagram. As you see any design framework that would like to provide this pattern need to define two upper entities, i.e. *Interested* interface and *Notifier* abstract class. So these provide the skeleton for supporting the observer pattern. Any class, which would like to act as a notifier, would inherit from the *Notifier* class, and any class, which is interested to

be notified would implement the *Interested* Interface. *subscribe(Interested)* and *unSubscribe(interested)* method are defined final, since there functionality is just adding the class to the subscription list and removing it, and need not to be altered by inheritance. *notifyAll()* method is also need not to be altered and it is also defined private. It only reads the subscription list and call the notify() method of the subscribers.