# Observer Pattern

Problem: A lot of Stock market customers wants to keep watch of the stock prices. Checking it manually every few minutes is not a feasible option since the stock prices vary rapidly.

Solution: Customers (observers) can subscribe to watch those stock prices and get notified every time a new stock price is set. The stock exchange (Subject) can have many observers and all the attached observers gets notified. If the customer is no longer interested, they can unsubscribe (detach) from the Stock Exchange and gets not further notifications. The main intent of using this pattern is to define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

# Proxy Pattern

Problem: There can be a lot of employees in the company that needs access to database to view files but not all of them should have access to add, edit or delete files.

Solution: In order to limit the access, a protection proxy can be created. All the employees with limited access (DatabaseUsersProxy ) who does not have valid username and password can only read the files from the database. The employees with full access (DatabaseUsers) can add, edit or delete files by providing correct username and password and accessing the database. The main intent of using this pattern is to provide a surrogate or placeholder for another object to control access to it. In this scenario, the surrogate object checks that the employee has the access permissions required prior to letting employee perform add, edit or delete action on database.