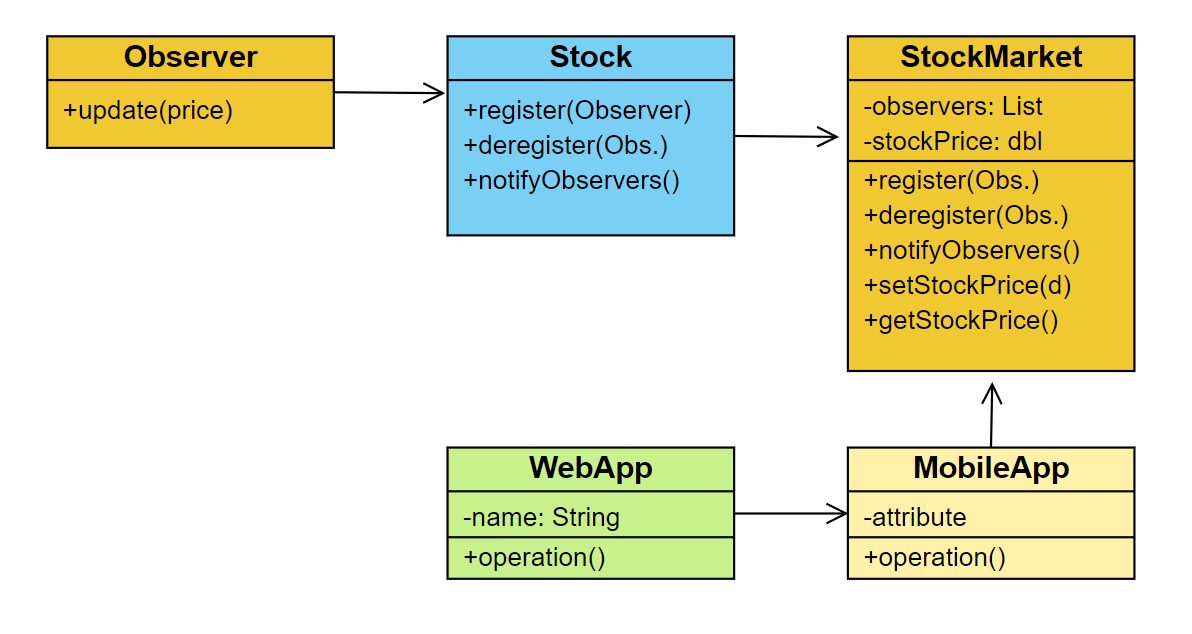
**Exercise 7: Implementing the Observer Pattern**

This code demonstrates the implementation of the Observer Pattern in Java for a stock market monitoring application. It includes an interface for the subject (Stock) and observers (Observer), a concrete subject class (*StockMarket*) to manage and notify observers, and concrete observer classes (*MobileApp* and *WebApp*) to receive updates. The main class handles user input to dynamically register observers and update stock prices, triggering notifications to all registered observers.

1. A new Java project named *ObserverPatternExample* has been created.

2. The Subject interface is defined as *Stock* with methods:

* *register(Observer o)*
* *deregister(Observer o)*
* *notifyObservers()*

3. The Concrete Subject *StockMarket* class implements the `Stock` interface and maintains a list of observers.

4. The Observer interface is defined as *Observer* with the method:

* *update(double stockPrice)*

5. Two Concrete Observers are implemented:

* *MobileApp* class
* *WebApp* class

6. The *ObserverPatternExample* class demonstrates the registration and notification of observers:

* It creates a *StockMarket* instance
* Allows user to add multiple *MobileApp* and *WebApp* observers
* Provides an interface for updating stock prices
* When stock prices are updated, all registered observers are notified

The implementation follows the Observer Pattern, allowing multiple clients (observers) to be notified when the stock price (subject) changes.

Here is the GitHub link of the repo - [link](https://github.com/Hyperstrom/Aniket-Pal_5017587/tree/main/WEEK-1/2.Design%20Patterns%20and%20Principles/Excercise-7)

**Here is the Example of the code –**

