**Exercise 7: Implementing the Observer Pattern**

**Scenario:**

You are developing a stock market monitoring application where multiple clients need to be notified whenever stock prices change. Use the Observer Pattern to achieve this.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **ObserverPatternExample**.
2. **Define Subject Interface:**
   * Create an interface **Stock** with methods to **register**, **deregister**, and **notify** observers.
3. **Implement Concrete Subject:**
   * Create a class **StockMarket** that implements **Stock** and maintains a list of observers.
4. **Define Observer Interface:**
   * Create an interface Observer with a method **update().**
5. **Implement Concrete Observers:**
   * Create classes **MobileApp**, **WebApp** that implement Observer.
6. **Test the Observer Implementation:**
   * Create a test class to demonstrate the registration and notification of observers.

**Solution:**

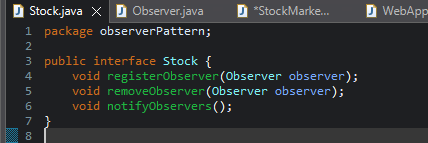
| **Component** | **Role** | **Description** |
| --- | --- | --- |
| **Stock** | Subject Interface | Declares methods to register, remove, and notify observers |
| **StockMarket** | Concrete Subject | Maintains stock price and list of observers; notifies them on change |
| **Observer** | Observer Interface | Declares update() method for receiving notifications |
| **MobileApp, WebApp** | Concrete Observers | Implement Observer and define how to handle updates |

**Code:**

**Stock.java**

**Explanation:**

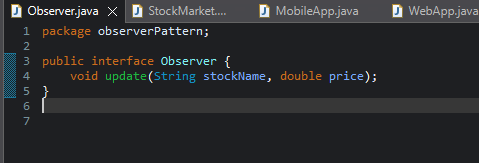
* Declares methods to add, remove, and notify observers.
* All concrete subjects (like StockMarket) will implement this.



**Observer.java**

**Explanation:**

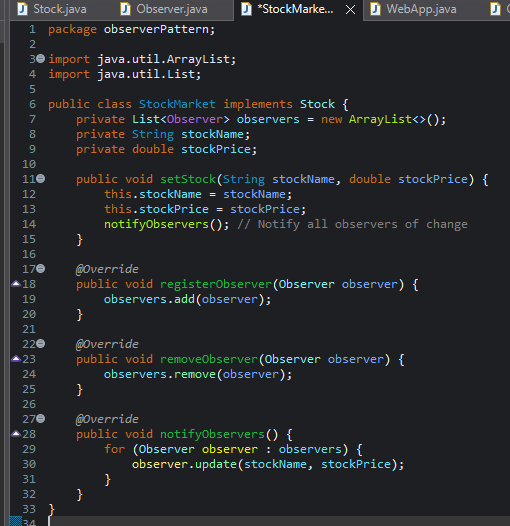
* All observers (clients) implement this interface to receive updates.

****

**StockMarket.java**

**Explanation:**

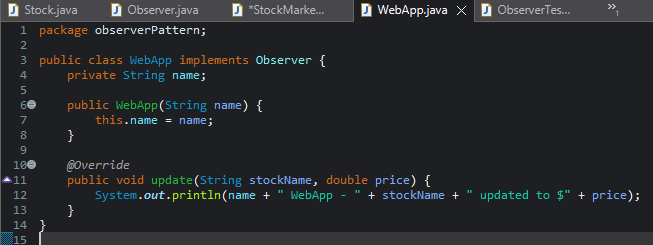
* Stores a list of all registered observers.
* When setStock() is called (stock data changes), it calls notifyObservers().

****

**MobileApp.java**

****

**WebApp.java**

****

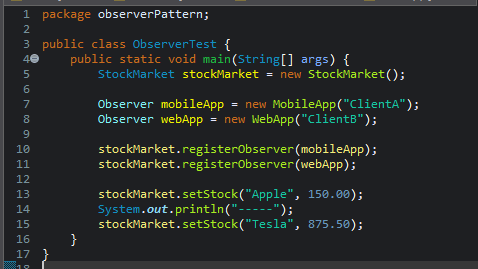
**Explanation:**

* These classes implement Observer and define how updates should be displayed.
* You can customize behavior per client.

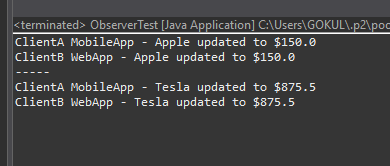
**ObserverTest.java**

**Explanation:**

* Registers multiple observers.
* When stock data is updated, all registered observers are notified in real time.

****

**Output:**

****