**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.

**Stock Interface**: The Stock interface defines methods for registering, deregistering, and notifying observers.

**StockMarket Class**: The StockMarket class implements the Stock interface. It maintains a list of observers and notifies them whenever the stock price changes. The setStockPrice method updates the stock price and calls notifyObservers.

**Observer Interface**: The Observer interface defines the update method that all observers must implement.

**MobileApp and WebApp Classes**: These classes implement the Observer interface and define the update method to receive stock price updates.

**Testing the Observer Pattern**: In the ObserverPatternTest class, we create instances of StockMarket and observer classes, register the observers, update the stock price, and observe the notifications received by the registered observers.