**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:

1st file: Stock.java

public interface Stock {

void register(Observer o);

void deregister(Observer o);

void notifyObservers();

}

2nd file: Observer.java

public interface Observer {

void update(String stockName, double price);

}

3rd file: StockMarket.java

import java.util.ArrayList;

import java.util.List;

public class StockMarket implements Stock {

private List<Observer> observers = new ArrayList<>();

private String stockName;

private double stockPrice;

public void setStock(String stockName, double stockPrice) {

this.stockName = stockName;

this.stockPrice = stockPrice;

notifyObservers();

}

@Override

public void register(Observer o) {

observers.add(o);

}

@Override

public void deregister(Observer o) {

observers.remove(o);

}

@Override

public void notifyObservers() {

for (Observer o : observers) {

o.update(stockName, stockPrice);

}

}

}

4th file : MobileApp.java

public class MobileApp implements Observer {

private String name;

public MobileApp(String name) {

this.name = name;

}

@Override

public void update(String stockName, double price) {

System.out.println(name + " (Mobile): " + stockName + " updated to $" + price);

}

}

5th file: WebApp.java

public class WebApp implements Observer {

private String name;

public WebApp(String name) {

this.name = name;

}

@Override

public void update(String stockName, double price) {

System.out.println(name + " (Web): " + stockName + " updated to $" + price);

}

}

6th file: ObserverPatternExample.java

public class ObserverPatternExample {

public static void main(String[] args) {

StockMarket market = new StockMarket();

Observer mobileUser = new MobileApp("Alice");

Observer webUser = new WebApp("Bob");

market.register(mobileUser);

market.register(webUser);

market.setStock("AAPL", 185.50);

market.setStock("GOOG", 2732.80);

market.deregister(mobileUser);

market.setStock("TSLA", 710.22);

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.