**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.

**Main.java :-**

public class Main {

    public static void main(String[] args) {

        StockMarket market = new StockMarket();

        Observer mobileUser1 = new MobileApp("Naresh");

        Observer webUser1 = new WebApp("InvestorX");

        market.registerObserver(mobileUser1);

        market.registerObserver(webUser1);

        market.setStockPrice("TCS", 3725.50);

        System.out.println("---------------------------");

        market.setStockPrice("Infosys", 1450.75);

        System.out.println("---------------------------");

        // Remove an observer and update again

        market.removeObserver(webUser1);

        market.setStockPrice("Wipro", 412.60);

    }

}

**MobileApp.java:-**

public class MobileApp implements Observer {

    private final String user;

    public MobileApp(String user) {

        this.user = user;

    }

    @Override

    public void update(String stockName, double stockPrice) {

        System.out.println("MobileApp [" + user + "] -> " + stockName + " updated to INR " + stockPrice);

    }

}

**Observer.java:-**

public interface Observer {

    void update(String stockName, double stockPrice);

}

**Stock.java :-**

public interface Stock {

    void registerObserver(Observer o);

    void removeObserver(Observer o);

    void notifyObservers();

}

**StockMarketjava :-**

import java.util.ArrayList;

import java.util.List;

public class StockMarket implements Stock {

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

    private String stockName;

    private double stockPrice;

    @Override

    public void registerObserver(Observer o) {

        observers.add(o);

    }

    @Override

    public void removeObserver(Observer o) {

        observers.remove(o);

    }

    @Override

    public void notifyObservers() {

        for (Observer observer : observers) {

            observer.update(stockName, stockPrice);

        }

    }

    // Method to change stock price and notify all observers

    public void setStockPrice(String stockName, double stockPrice) {

        this.stockName = stockName;

        this.stockPrice = stockPrice;

        System.out.println("Stock updated: " + stockName + " - INR" + stockPrice);

        notifyObservers();

    }

}

**WebApp.java :-**

public class WebApp implements Observer {

    private final String user;

    public WebApp(String user) {

        this.user = user;

    }

    @Override

    public void update(String stockName, double stockPrice) {

        System.out.println("WebApp [" + user + "] -> " + stockName + " updated to INR" + stockPrice);

    }

}

**Output :-**

