**Exercise 7: Implementing the Observer Pattern**

**Stock.java**

public interface Stock {

void registerObserver(Observer o);

void removeObserver(Observer o);

void notifyObservers();

}

**Observer.java**

public interface Observer {

void update(double price);

}

**StockMarket.java**

import java.util.ArrayList;

import java.util.List;

public class StockMarket implements Stock {

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

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 o : observers) {

o.update(stockPrice);

}

}

public void setStockPrice(double price) {

this.stockPrice = price;

System.***out***.println("\nStock price updated to: " + price);

notifyObservers();

}

}

**MobileApp.java**

public class MobileApp implements Observer {

private String user;

public MobileApp(String user) {

this.user = user;

}

*@Override*

public void update(double price) {

System.***out***.println("MobileApp [" + user + "] - New Stock Price: " + price);

}

}

**WebApp.java**

public class WebApp implements Observer {

private String user;

public WebApp(String user) {

this.user = user;

}

*@Override*

public void update(double price) {

System.***out***.println("WebApp [" + user + "] - New Stock Price: " + price);

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

StockMarket market = new StockMarket();

Observer mobileUser1 = new MobileApp("Alice");

Observer mobileUser2 = new MobileApp("Bob");

Observer webUser = new WebApp("Charlie");

market.registerObserver(mobileUser1);

market.registerObserver(mobileUser2);

market.registerObserver(webUser);

market.setStockPrice(120.75);

market.setStockPrice(125.50);

market.removeObserver(mobileUser2);

market.setStockPrice(130.25);

}

Output:

