Exercise 7: Stock Market Live Price Updater using Observer Pattern

Approach and Understanding:

In this problem, I made a **live stock market system** where:

* Users (like Mobile and Web apps) will **get instant stock price updates.**
* I used the **Observer Pattern** because it helps to **send updates automatically to all users.**

**Code**

import java.util.ArrayList; import java.util.List;

class StockMarket {

List<Observer> friends = new ArrayList<>(); double price;

void updatePrice(double newPrice) { this.price = newPrice;

for (Observer friend : friends) { friend.receiveUpdate(price);

}

}

void addFriend(Observer friend) { friends.add(friend);

}

void removeFriend(Observer friend) { friends.remove(friend);

}

}

interface Observer {

void receiveUpdate(double newPrice);

}

class MobileApp implements Observer { public void receiveUpdate(double price) {

System.out.println("Mobile App: The stock price is now " + price);

}

}

class WebApp implements Observer { public void receiveUpdate(double price) {

System.out.println("Website: The stock price is now " + price);

}

}

public class StockObserver {

public static void main(String[] args) { StockMarket market = new StockMarket();

MobileApp mobile = new MobileApp(); WebApp website = new WebApp();

market.addFriend(mobile); market.addFriend(website);

System.out.println("Price changes to 100..."); market.updatePrice(100);

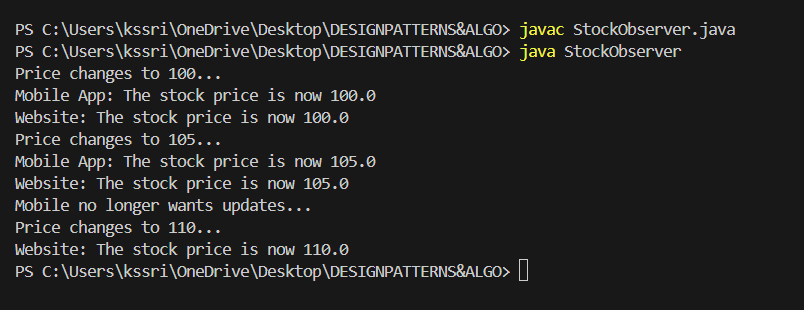
System.out.println("Price changes to 105..."); market.updatePrice(105);

System.out.println("Mobile no longer wants updates..."); market.removeFriend(mobile);

System.out.println("Price changes to 110..."); market.updatePrice(110);

}

}

**OUTPUT:**