**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.

1. **Define Subject Interface:**

* Create an interface Stock with methods to register, deregister, and notify observers.

1. **Implement Concrete Subject:**

* Create a class StockMarket that implements Stock and maintains a list of observers.

1. **Define Observer Interface:**

* Create an interface Observer with a method update().

1. **Implement Concrete Observers:**

* Create classes MobileApp, WebApp that implement Observer.

1. **Test the Observer Implementation:**

* Create a test class to demonstrate the registration and notification of observers.

**SOLUTION:**

**Step 1:** Create a New Java Project

Create a new Java project named ObserverPatternExample.

**Step 2:** Define Subject Interface

interface Stock {

void register(Observer obs);

void deregister(Observer obs);

void notifyObservers();

}

**Step 3:** Implement Concrete Subject

class StockMarket implements Stock {

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

@Override

public void register(Observer observer) {

observers.add(observer);

}

@Override

public void deregister(Observer observer) {

observers.remove(observer);

}

@Override

public void notifyObservers() {

for (Observer observer : observers) {

observer.update();

}

}

}

**Step 4:** Define Observer Interface

interface Observer {

void update();

}

**Step 5:** Implement Concrete Observers

class MobileApp implements Observer {

private String username;

private String email;

private String phone;

private StockMarket stockMarket;

public MobileApp(String username, String email, String phone, StockMarket stockMarket) {

this.username = username;

this.email = email;

this.phone = phone;

this.stockMarket = stockMarket;

}

@Override

public void update() {

System.out.println(username + ", you received a new notification! Check the stock price changes. (Mobile App)");

}

}

class WebApp implements Observer {

private String username;

private String email;

private String phone;

private StockMarket stockMarket;

public WebApp(String username, String email, String phone, StockMarket stockMarket) {

this.username = username;

this.email = email;

this.phone = phone;

this.stockMarket = stockMarket;

}

@Override

public void update() {

System.out.println(username + ", you received a new notification! Check the stock price changes. (Web App)");

}

}

**Step 6:** Test the Observer Implementation

public class ObserverPatternExample {

public static void main(String[] args) {

StockMarket stockMarket = new StockMarket();

Observer webApp1 = new WebApp("Kaushal", "kaushal01@gmail.com", "9876543210", stockMarket);

Observer webApp2 = new WebApp("Vicky", "vicky12@gmail.com", "9876543210", stockMarket);

Observer mobileApp = new MobileApp("Tushar", "tusharsingh@gmail.com", "1234567890", stockMarket);

stockMarket.register(webApp1);

stockMarket.register(webApp2);

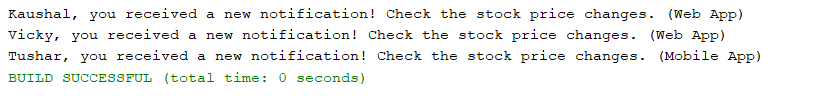
stockMarket.register(mobileApp);

stockMarket.notifyObservers();

}

}

**SAMPLE OUTPUT:**

****