Design patterns and principles

SuperSet ID:6412063

Exercise 7: Implementing the Observer Pattern

Code:

public interface Stock {

void registerObserver(Observer observer);

void removeObserver(Observer observer);

void notifyObservers();

}

import java.util.ArrayList;

import java.util.List;

public class StockMarket implements Stock {

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

private double stockPrice;

public void registerObserver(Observer observer) {

observers.add(observer);

}

public void removeObserver(Observer observer) {

observers.remove(observer);

}

public void notifyObservers() {

for (Observer observer : observers) {

observer.update(stockPrice);

}

}

public void setStockPrice(double price) {

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

this.stockPrice = price;

notifyObservers();

}

}

public interface Observer {

void update(double stockPrice);

}

public class MobileApp implements Observer {

private String name;

public MobileApp(String name) {

this.name = name;

}

public void update(double stockPrice) {

System.out.println("Mobile App [" + name + "] - New stock price: " + stockPrice);

}

}

public class WebApp implements Observer {

private String name;

public WebApp(String name) {

this.name = name;

}

public void update(double stockPrice) {

System.out.println("Web App [" + name + "] - New stock price: " + stockPrice);

}

}

public class StockMarketTest {

public static void main(String[] args) {

StockMarket stockMarket = new StockMarket();

Observer mobile1 = new MobileApp("User1");

Observer web1 = new WebApp("ClientA");

stockMarket.registerObserver(mobile1);

stockMarket.registerObserver(web1);

stockMarket.setStockPrice(150.75);

stockMarket.setStockPrice(155.10);

stockMarket.removeObserver(web1);

stockMarket.setStockPrice(160.00);

}

}

Output:

A computer screen shot of text

Description automatically generated