**Exercise 7: Implementing the Observer Pattern**

**Source Code:**

**Stock.java**

public interface Stock {

void register(Observer o);

void deregister(Observer o);

void setPrice(double p);

void notifyObservers();

String getSymbol();

double getPrice();

}

**StockMarket.java**

import java.util.\*;

public class StockMarket implements Stock {

private final String symbol;

private double price;

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

public StockMarket(String symbol) { this.symbol = symbol; }

public void register(Observer o) { obs.add(o); }

public void deregister(Observer o){ obs.remove(o); }

public void setPrice(double p) {

price = p;

notifyObservers();

}

public void notifyObservers() {

for (Observer o : obs) o.update(symbol, price);

}

public String getSymbol() { return symbol; }

public double getPrice() { return price; }

}

**Observer.java**

public interface Observer {

void update(String symbol, double price);

}

**MobileApp.java**

public class MobileApp implements Observer {

public void update(String symbol, double price) {

System.out.println("MobileApp: " + symbol + " $" + price);

}

}

**WebApp.java**

public class WebApp implements Observer {

public void update(String symbol, double price) {

System.out.println("WebApp: " + symbol + " $" + price);

}

}

**StockObserverDemo.java**

public class StockObserverDemo {

public static void main(String[] args) {

StockMarket tesla = new StockMarket("TSLA");

Observer mobile = new MobileApp();

Observer web = new WebApp();

tesla.register(mobile);

tesla.register(web);

tesla.setPrice(750.45);

tesla.setPrice(760.10);

tesla.deregister(web);

tesla.setPrice(770.00);

}

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**