Multithreaded Application and Observer Design Pattern

**The quiz open book and notes and you can use the Internet. You are not allowed to consult other individuals or your classmates.**

PLEASE UPLOAD YOUR ANSWER USING THIS **WORD** FILE. DO NOT UPLOAD ANY ZIPPED OR PDF FILE. ONE WORD FILE ONLY.

The question(s) have been written in a manner such that it is not possible for two students to have the same solution. Therefore, please refrain from the temptation of copying code and changing order of and names of variables, etc. AUS code of conduct will be strictly enforced, and no violation will be tolerated as per AUS policy. Please note that the AUS code does not discriminate between who copied from whom so it is not advisable to share your solution.

**Q1.** (10 points) You hire two stock market assistants to help you decide whether to buy Amazon stocks or not. Amazon publishes the stock price every **other second.**

**Stock market assistant 1** makes his decision to buy or sell stocks based on the current and past stock price. If the current stock price is greater than the past stock price, stock assistant 1 thinks it is wise to sell the stocks, otherwise he thinks it is wise to buy the stock.

**Stock Assistant 2** is more cautious. He suggests to sell the stocks when the last two stock prices have an upward trend and to buy the stocks when the last two stock prices see a downward trend. Otherwise, he suggests to do nothing.

Both Stock market assistants give you their suggestions every **5 seconds.**

Implement Java classes StockAssistant1 and StockAssistant2 to carry out the above functionality. The classes should implement the Observer interface (given below) and run in their own thread.

public interface Observer {

void update(int sharePrice);

}

You must run and provide output on the following main program (without any changes to the program). The implementation of the Subject (AMZN) is given below.

You are not allowed to make any changes to the code given below

public class Main {

public static void main(String[] args) {

AMZN A = new AMZN();

ConcreteSubject[] subjects = {A};

StockAssistant1 sa = new StockAssistant1(subjects);

StockAssistant2 sb = new StockAssistant2(subjects);

}

}

**Interface Subject (subject.java)**

public interface Subject {

public void registerObserver(Observer o);

public void removeObsever(Observer o);

public void notifyObservers();

}

**Class ConcreteSubject (ConcreteSubject.java)**

import java.util.ArrayList;

import java.util.Random;

public class ConcreteSubject implements Subject {

private ArrayList<Observer> observers;

Random r = new Random();

public ConcreteSubject() {

observers = new ArrayList<Observer>();

}

public void notifyObservers() {

int val = r.nextInt(100);

System.out.println("Current Stock Value: " + val);

for (int i = 0; i < observers.size(); i++) {

Observer observer = observers.get(i);

observer.update(val);

}

}

public void registerObserver(Observer o) {

observers.add(o);

}

public void removeObsever(Observer o) {

observers.remove(o);

}

}

**Class AMZN (AMZN.java)**

public class AMZN extends ConcreteSubject implements Runnable{

AMZN() {

Thread t = new Thread(this);

t.start();

}

**Expected Output:**

Current Stock Value: 65

Current Stock Value: 70

Current Stock Value: 96

StockAssistant1@bf5e453 Sell the stocks

StockAssistant2@213b1b71 Sell the stocks

Current Stock Value: 26

Current Stock Value: 61

StockAssistant1@bf5e453 Sell the stocks

StockAssistant2@213b1b71 Do nothing

Current Stock Value: 70

Current Stock Value: 83

Current Stock Value: 58

StockAssistant2@213b1b71 Do nothing

StockAssistant1@bf5e453 Buy the stocks

Current Stock Value: 81

Current Stock Value: 26

public void run() {

while (true) {

notifyObservers();

try {

Thread.sleep(2000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}