Assignment 4

Object-Oriented Software Development

Adding Current Market Publications (200 pts)

Objectives

Create well-written object-oriented classes needed to support the publication of Current Market messages (the top price and top volume (at that price) for the Buy and Sell sides). This will involve the addition of a few new classes and a few small changes to some existing classes. You should also create any new exception classes if needed to handle invalid situations.

New Classes

1) **CurrentMarketSide** class

This class holds the top price and top volume (at that price) for one market side. It is used with current market publications. The CurrentMarketSide will need the following data elements to properly represent the price and volume (private):

- Price price ➔ The top-of-book price for one market side (set on construction, not changeable)
- Int volume ➔ The top-of-book volume for one market side (set on construction, not changeable)

The CurrentMarketSide will need a constructor that accepts and sets the price and volume values.

The CurrentMarketSide needs a toString that generates and returns a String as follows:

public String toString() ➔ returns a String with the price and volume, as: `$98.10x105`

2) **CurrentMarketObserver** interface

This interface is implemented by classes that want to be able to register for Current Market updates (currently the User class will implement this). The CurrentMarketObserver will need to define the following method:

- void updateCurrentMarket(String symbol, CurrentMarketSide buySide, CurrentMarketSide sellSide);

3) **CurrentMarketTracker** class

This class is a Singleton that receives Current Market updates from the ProductBooks, and sends the information on to the CurrentMarketPublisher. This class has one method, as follows:

- public void updateMarket(String symbol, Price buyPrice, int buyVolume, Price sellPrice, int sellVolume) ➔ This method should do the following:
    - Calculate the market width as the difference in price between the sellPrice and the buyPrice. Note, if either the sellPrice or the buyPrice are null, the width should be considered 0.
    - Create a CurrentMarketSide object for the buy side using the buyPrice and buyVolume.
    - Create a CurrentMarketSide object for the sell side using the sellPrice and sellVolume.

- Print the current market as follows:

```
********** Current Market **********
* Symbol   buy price x buy volume – sell price x sell volume [market width]
************************************
```

Example:

```
********** Current Market **********
* WMT   $70.70x35 - $70.75x40 [$0.05]
************************************
```

- Call the CurrentMarketPublisher's *acceptCurrentMarket* method, passing the symbol, and the buy and sell side CurrentMarketSide objects.

4) **CurrentMarketPublisher** class

This class is a Singleton that uses the Observer pattern. It maintains a list of observers, and filters (what observers want the Current Market for what stocks). It also publishes the Current Market updates to the subscribed "observers". The CurrentMarketPublisher needs the following (private) data fields to maintain the observers and filters:

- HashMap<String, ArrayList<CurrentMarketObserver>> filters ➔ Holds data on what stocks the subscribed CurrentMarketObservers want to receive. The key is the stock symbol, the value is a list of CurrentMarketObservers that want to receive Current Market updates for that stock symbol.

The CurrentMarketPublisher requires the following (public) methods in order to maintain subscriptions and perform publications:

- void subscribeCurrentMarket(String symbol, CurrentMarketObserver cmo) ➔ Using the stock symbol as the key to the *filters* HashMap, get the associated ArrayList. If the symbol is not a key in the HashMap, add that key with a new ArrayList to the HashMap. Then add the CurrentMarketObserver object to that ArrayList.
- void unSubscribeCurrentMarket(String symbol, CurrentMarketObserver cmo) ➔ Using the stock symbol as the key to the *filters* HashMap, get the associated ArrayList. Then remove the CurrentMarketObserver passed in from that ArrayList. If the symbol is not a key in the HashMap, then simply return.
- void acceptCurrentMarket(String symbol, CurrentMarketSide buySide, CurrentMarketSide sellSide) ➔ This method is called by the CurrentMarketTracker and includes buy and sell side current market values for a specified stock. This is done as follows:
  - If the symbol is not a key in the filters HashMap, then simply return (there are no CurrentMarketObservers registered for that symbol).
  - Get the ArrayList of CurrentMarketObservers from the *filters* HashMap using the stock symbol passed in as the key.
  - For each CurrentMarketObserver in that ArrayList, call their *updateCurrentMarket* method passing the stock symbol, and the buy and sell side CurrentMarketSide objects passed in.

Changed Classes:

1) **ProductBook** class

- New method: private void updateMarket() ➔ This method should report the current market (top-of-book price & volume for the buy & sell sides). This is done as follows:
  - Get top-of-book Price and volume for both the buy and sell sides.
  - Call the CurrentMarketTracker's *updateMarket* method, passing the String stock symbol, the buy-side top-of-book Price, top-of-book volume, the sell-side top-of-book Price, top-of-book volume.
- Changed method: public OrderDTO add(Order o) ➔ Add a call to the *updateMarket()* method immediately after the existing call to *tryTrade().*
- Changed method: public OrderDTO cancel(BookSide side, String orderId) ➔ Add a call to the *updateMarket()* method immediately after the existing call to the ProductBookSide's *cancel(…)* method.

2) **User** class

The User class needs updates to receive and save the current market values for certain stocks. Changes needed are:

- Change declaration: make your User class implement the CurrentMarketObserver interface. This requires you to implement the *updateCurrentMarket* method (see below).
- Add private data field: HashMap<String, CurrentMarketSide[]> currentMarkets ➔ Initialize this to a new HashMap. This field is used in the *updateCurrentMarket* method (see below).
- New method (required by the CurrentMarketObserver interface): public void updateCurrentMarket(String symbol, CurrentMarketSide buySide, CurrentMarketSide sellSide) ➔ This method should create a 2-element array of CurrentMarketSide objects ([0]=Buy, [1]=Sell). Then put the values in the *currentMarkets* HashMap, using the stock symbol as the key and the array of CurrentMarketSide objects as the value.
- New method: public String getCurrentMarkets() ➔ Method should generate and return a String summary of the values in the *currentMarkets* HashMap. This String should be in the format:

  Symbol TopBuyPrice x TopBuyVolume - TopSellPrice x TopSellVolume (one line per stock symbol)

  Example:

```
TGT   $87.45x210 - $87.65x75
WMT   $70.20x170 - $70.65x15
```

3) **UserManager** class

The UserManager needs one new method to access the Users:

- New method: public User getUser(String id) ➔ The method should return the User from the HashMap of Users, using the String id as the key.

4) **TrafficSim** class

The TrafficSim class need a few updates to account for the new CurrentMarket publishing feature. These changes are as follows:

- In the TrafficSim's *runSim* method:
  - After the call to the UserManager's *init* method, create 5 User references and set them to each of the 5 test users (ANN, BOB, CAT, DOG, EGG). You get the references be using the UserManager's *getUser* method.
  - After you do the above, next add the calls to the CurrentMarketPublisher's *subscribeCurrentMarket* method to subscribe users for current market publications. Subscribe the users as follows:
    - ANN: WMT, TGT
    - BOB: TGT, TSLA
    - CAT: AMZN, TGT, WMT
    - DOG: TSLA
    - EGG: WMT
  - Next, calls the CurrentMarketPublisher's *unSubscribeCurrentMarket* method to un-subscribe user BOB from current market for TGT. (We do this to be sure the un-subscribe is working).
  - Set your simulation loop limit to 10000
  - After the loop is done, print the current market values for each user (use the User's *getCurrentMarkets* method)
  - Finally, add calls to the CurrentMarketPublisher's *unSubscribeCurrentMarket* method to un-subscribe the users from all symbols the previously subscribed to.

5) Testing

Once again as in the previous assignment, call the `TradingSim.runSim()` function. The below is a sample (subset) of the expected output. The Current Market messages are in **blue**.

```
ADD: BUY: BOB order: BUY TSLA at $98.70, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID:
       BOBTSLA$98.70294002029160300
*********** Current Market ***********
* TSLA   $98.70x45 - $0.00x0 [$0.00]
************************************
ADD: BUY: CAT order: BUY WMT at $70.15, Orig Vol: 245, Rem Vol: 245, Fill Vol: 0, CXL Vol: 0, ID:
       CATWMT$70.15294002076994000
*********** Current Market ***********
* WMT   $70.15x245 - $0.00x0 [$0.00]
************************************
ADD: BUY: DOG order: BUY WMT at $70.15, Orig Vol: 275, Rem Vol: 275, Fill Vol: 0, CXL Vol: 0, ID:
       DOGWMT$70.15294002077578900
*********** Current Market ***********
* WMT   $70.15x520 - $0.00x0 [$0.00]
************************************
ADD: SELL: BOB order: SELL TSLA at $97.95, Orig Vol: 250, Rem Vol: 250, Fill Vol: 0, CXL Vol: 0, ID:
       BOBTSLA$97.95294002078197500
   PARTIAL FILL: (SELL 45) BOB order: SELL TSLA at $97.95, Orig Vol: 250, Rem Vol: 205, Fill Vol: 45, CXL
       Vol: 0, ID: BOBTSLA$97.95294002078197500
   FULL FILL: (BUY 45) BOB order: BUY TSLA at $98.70, Orig Vol: 45, Rem Vol: 0, Fill Vol: 45, CXL Vol: 0,
       ID: BOBTSLA$98.70294002029160300
*********** Current Market ***********
* TSLA   $0.00x0 - $97.95x205 [$0.00]
************************************
ADD: BUY: ANN order: BUY WMT at $70.30, Orig Vol: 185, Rem Vol: 185, Fill Vol: 0, CXL Vol: 0, ID:
       ANNWMT$70.30294002083989700
*********** Current Market ***********
* WMT   $70.30x185 - $0.00x0 [$0.00]
************************************
ADD: BUY: CAT order: BUY TGT at $87.25, Orig Vol: 160, Rem Vol: 160, Fill Vol: 0, CXL Vol: 0, ID:
       CATTGT$87.25294002084746300
*********** Current Market ***********
```

```
* TGT   $87.25x160 - $0.00x0 [$0.00]
**************************************
CANCEL: BUY Order: DOGWMT$70.15294002077578900 Cxl Qty: 275
*********** Current Market ***********
* WMT   $70.30x185 - $0.00x0 [$0.00]
**************************************
ADD: BUY: EGG order: BUY TSLA at $98.85, Orig Vol: 305, Rem Vol: 305, Fill Vol: 0, CXL Vol: 0, ID:
       EGGTSLA$98.85294002089932400
    FULL FILL: (SELL 205) BOB order: SELL TSLA at $97.95, Orig Vol: 250, Rem Vol: 0, Fill Vol: 250, CXL Vol: 0,
       ID: BOBTSLA$97.95294002078197500
    PARTIAL FILL: (BUY 205) EGG order: BUY TSLA at $98.85, Orig Vol: 305, Rem Vol: 100, Fill Vol: 205, CXL Vol:
       0, ID: EGGTSLA$98.85294002089932400
*********** Current Market ***********
* TSLA   $98.85x100 - $0.00x0 [$0.00]
**************************************
ADD: SELL: BOB order: SELL WMT at $70.00, Orig Vol: 130, Rem Vol: 130, Fill Vol: 0, CXL Vol: 0, ID:
       BOBWMT$70.00294002090612300
    FULL FILL: (SELL 130) BOB order: SELL WMT at $70.00, Orig Vol: 130, Rem Vol: 0, Fill Vol: 130, CXL Vol: 0,
       ID: BOBWMT$70.00294002090612300
    PARTIAL FILL: (BUY 130) ANN order: BUY WMT at $70.30, Orig Vol: 185, Rem Vol: 55, Fill Vol: 130, CXL Vol: 0,
       ID: ANNWMT$70.30294002083989700
*********** Current Market ***********
* WMT   $70.30x55 - $0.00x0 [$0.00]
**************************************

[…]

ADD: BUY: DOG order: BUY TGT at $87.85, Orig Vol: 115, Rem Vol: 115, Fill Vol: 0, CXL Vol: 0, ID:
       DOGTGT$87.85294357585953100
*********** Current Market ***********
* TGT   $87.85x115 - $88.00x45 [$0.15]
**************************************

ANN:
TGT   $86.70x230 - $87.00x30
WMT   $70.00x280 - $70.15x155

BOB:
TSLA   $97.60x195 - $98.30x95

CAT:
TGT   $86.70x230 - $87.00x30
AMZN   $50.90x435 - $51.05x135
WMT   $70.00x280 - $70.15x155

DOG:
TSLA   $97.60x195 - $98.30x95

EGG:
WMT   $70.00x280 - $70.15x155
```

Project Assistance

If you are stuck on some design or code related problem *that you have exhaustively researched and/or debugged yourself*, you can email me a ZIP file of your entire project so that I can examine the problem.

All emailed assistance requests must include a detailed description of the problem, and the details of what steps you have already taken in trying to determine the source of the problem.

Submissions & Grading

When submitting, you should submit a ZIP file of your entire project so that I can compile and execute it on my end. To reduce the size of the zipped file, please remove the "out" folder from your project (if present) before zipping. (This folder is automatically regenerated each time you compile/run so It's safe to delete)

The following are the key points that will be examined in Project when graded:

- Good object-oriented design & implementation
- Proper use of java language conventions
- Proper object-oriented coding practices
- Correct, accurate application execution

Submissions must reflect the concepts and practices we cover in class, and the requirements specified in this document.

Please review the Academic Integrity and Plagiarism section of the syllabus before, during, and after working on this assignment. All work must be your own.

If you do not understand anything in this handout, please ask. Otherwise the assumption is that you understand the content.