SE 350 - Assignment 2

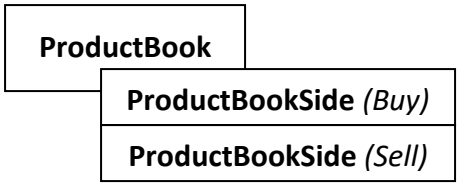
Object-Oriented Coding

Building the Product Book (200 pts)

Objectives

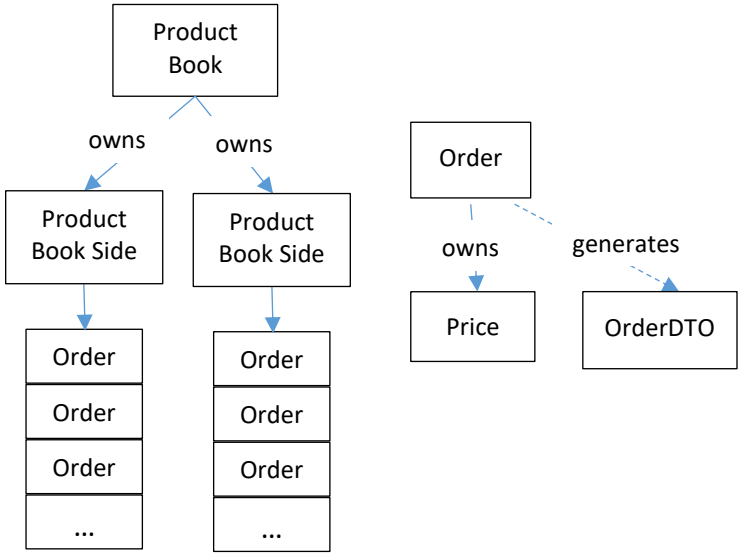
Create well-written object-oriented classes that represent a Product Book, Product Book Side, Order, and supporting classes. You should also create any new exception classes if needed to handle invalid situations.

Classes and their relationships



ProductBookSide

HashMap<Price, ArrayList<Order>>	
Key: Price	**Value: ArrayList<Order>**
$12.98 ➔	ArrayList of "Booked" Orders at this Price
$12.94 ➔	ArrayList of "Booked" Orders at this Price
Etc… ➔	ArrayList of "Booked" Orders at this Price



1) Order class

The Order class represents an order to buy or sell a volume (i.e., quantity) of stock at a certain price. The order will also maintain its processing state, detailed below.

- The Order class should have a public constructor that accepts and sets the parameters where indicated below.
- An Order must maintain certain data values – shown below (remember to make these private to enforce information hiding):
  - String user: A 3-letter user code – must be 3 letters, no spaces, no numbers, no special characters (i.e., XRF, BBT, KNT, etc.). This should be passed to the constructor. Cannot be changed once set.
  - String product: The stock symbol for the order (1 to 5 characters) – must from 1 to 5 letters/numbers, they can also have a period "." In the symbol. Otherwise, -no spaces, no special characters (i.e., WMT, F, 3M, GOOGL, WOOF, AKO.A, etc.). This should be passed to the constructor. Cannot be changed once set.
  - Price price: A Price object – cannot be null. This should be passed to the constructor. Cannot be changed once set.

- BookSide side: The "side" of the order – BUY or SELL. If an *enum* type, do not allow null. This should be passed into the constructor. Cannot be changed once set.
- String id: The order id – unique to each order. This should *not* be passed into the constructor, it should be generated and set *in* the constructor. This should be generated as: user + product + price (String) + nanotime (from System.nanoTime()). Cannot be changed once set.

  Example: "XRF" + "WMT" + "$139.59" + "157357272154300" = "XRFWMT$139.59157357272154300"
- int originalVolume: The quantity of the stock being ordered. <Must be greater than 0, and less than 10,000. This should be passed into the constructor. Cannot be changed once set.
- int remainingVolume – This should be initialized to the originalVolume value. This value is changeable as the order is filled.
- int cancelledVolume – This should be initialized to zero. This value is changeable when any part of the order is cancelled.
- int filledVolume - This should be initialized to zero. This value is changeable when part of the order is filled.

- An Order must be able to perform certain behaviors – shown below.
  - Create accessors and modifiers (that validate incoming parameters) for the private data members. Modifiers for data attributes that cannot be changed should be made private.
  - Override the toString method to generate a String like this:
    - USER order: SIDE PRODUCT at PRICE, Orig Vol: ORIGINAL VOLUME, Rem Vol: REMAINING VOLUME, Fill Vol: FILLED VOLUME, CXL Vol: CANCELLED VOLUME, ID: ID

      Example
      - XRF order: BUY AMZN at $96.38, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAAAMZN$96.38158311853491300
  - public OrderDTO makeTradableDTO() ➔ Generates and returns a DTO object (described later) for the current order

2) OrderDTO class

The OrderDTO class contains a copy of an order's data – used to transfer/return data about an order without giving out a referance to the actual order (good info hiding!).

- This class should have the same data fields as the Order class, but they should all be public (no validation checks needed since they are generated from an existing order).
- A constructor should be created that accepts and sets all the data fields (no validation checks needed since they are generated from an existing order).
- No accessors or modifiers are needed since the data is public.
- Override the toString method to generate a String just like you did in the Order class.

3) ProductBook class

A ProductBook maintains the Buy and Sell sides of a stock's "book". A stock's "book" holds the buy and sell orders for a stock that are not yet tradable. The buy-side of the book contains all buy orders that are not yet tradable in descending order. The sell-side of the book contains all sell orders that are not yet tradable in ascending order. Many of the functions owned by the ProductBook class are designed to simply pass along that same call to either the Buy or Sell side of the book. The "book" is often visually represented as follows:

MSFT (Microsoft) Book			
BUY		SELL	
1000	$29.05	$29.07	1000
120	$29.01	$29.08	120
210	$28.98	$29.10	210
80	$28.94	$29.12	80
…	…	…	…

- The ProductBook class should have a public constructor that accepts and sets the parameters as indicated below.
- The ProductBook will need the following data elements to properly represent a product's book:
  - String product: The stock symbol that this book represents (1 to 5 characters) – must be 1 to 5 letters/numbers, they can also have a period "." in the symbol (i.e., WMT, F, 3M, GOOGL, WOOF, AKO.A, etc.). Otherwise, no spaces, no special characters. This should be passed to the constructor and cannot be changed once set.
  - ProductBookSide buySide: ProductBookSide object that holds the Buy side of this book. Should be created and set in the constructor.
  - ProductBookSide sellSide: ProductBookSide object that holds the Sell side of this book. Should be created and set in the constructor.
- The ProductBook will need the following methods to properly represent a product's book:
  - public OrderDTO add(Order o): Call the "add" method of the ProductBookSide the order is for (BUY or SELL) and save the OrderDTO that is returned. Call "tryTrade", passing the side the order is for (BUY or SELL). Then return the Order DTO.
  - public OrderDTO cancel(BookSide side, String orderId): Call the "cancel" method of the ProductBookSide the order is for (BUY or SELL) and save the OrderDTO that is returned.
  - public void tryTrade(): Checks to see if the book sides are tradable, and if so, perform the trades. If not, it does nothing. See diagram in *Appendix A* for how this should work.
  - public String toString(): Override the toString method to generate a String containing a summary of the book content as follows (be sure to let the ProductBookSides generate their part of the String).

```
Product: WMT
  Side: BUY
    Price: $9.95
            CCC order: BUY AMZN at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCAMZN$9.95186495726402400
    Price: $9.90
            DDD order: BUY AMZN at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDAMZN$9.90186495726811600
  Side: SELL
    Price: $10.10
            EEE order: SELL AMZN at $10.10, Orig Vol: 120, Rem Vol: 10, Fill Vol: 110, CXL Vol: 0, ID: EEEAMZN$10.10186495727149200
    Price: $10.20
            FFF order: SELL AMZN at $10.20, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID: FFFAMZN$10.20186495727529400
    Price: $10.25
            GGG order: SELL AMZN at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: GGGAMZN$10.25186495727930400
```

4) ProductBookSide class

A ProductBookSide maintains the contents of one side (Buy or Sell) of a stock product "book". A product book "side" holds the buy or sell orders for a stock that are not yet tradable. The "book" is often visually represented as shown below (with the product book buy side circled).

- The buy-side contains all buy-side orders in descending order.
- The sell-side contains all sell-side orders in ascending order.

MSFT (Microsoft) Book			
BUY		SELL	
1000	$29.05	$29.07	1000
120	$29.01	$29.08	120
210	$28.98	$29.10	210
80	$28.94	$29.12	80
...	...	...	...

- The ProductBookSide class should have a public constructor that accepts and sets the parameters where indicated below.
- The ProductBookSide will need the following data elements to properly represent a product's book side:
  - BookSide side: The "side" of the order – BUY or SELL. If an enum type, do not allow null. This should be passed into the constructor. Cannot be changed once set.
  - final HashMap<Price, ArrayList<Order>> bookEntries: Contains the orders at each price that make up this book side. The diagram below shows how this is represented:

BUY Side	**HashMap< Price, ArrayList<Order>>**
Key: *Price*	**Value**: *ArrayList<Order>*
$12.98 →	[Order, Order, Order, Order, Order]
$12.95 →	[Order]
$12.90 →	[Order, Order, Order]

- The ProductBookSide will need the following methods to properly represent a product's book side:
  - public OrderDTO add(Order o): Add the incoming order to the bookEntries HashMap. If the price for the order does not exist as a key in the HashMap, add the price to the HashMap as the key and a new ArrayList<Order> as the value. Then add the order to that ArrayList, and return an OrderDTO made from the new order.
  - public OrderDTO cancel(String orderId): Find the order using the order id passed in (search the ArrayLists at each price key in the bookEntries HashMap to find it). If you find it, remove the order from the ArrayList, add the order's remaining volume to the order's cancelled volume, then set the remaining volume to zero. If the ArrayList is empty after removing the order, remove the price and the empty arraylist from the bookEntries HashMap. Finally, return an OrderDTO made from the cancelled order. If the order is not found, return null.

- public Price topOfBookPrice(): If the ProductBookSide is the BUY side, return the highest price in the bookEntries HashMap. If the ProductBookSide is the SELL side, return the lowest price in the bookEntries HashMap.
- public int topOfBookVolume(): If the ProductBookSide is the BUY side, return total of all order volumes at the highest price in the bookEntries HashMap. If the ProductBookSide is the SELL side, return total of all order volumes at the lowest price in the bookEntries HashMap.
- public void tradeOut(Price price, int vol): Trade out any orders at or betterthen the Price passed in, up to the volume value passed in. See diagram in *Appendix B* for how this should work.
- public String toString(): Override the toString method to generate a String containing a summary of the book side content as follows.

  BUY side example:

```
Side: BUY
    Price: $9.95
            CCC order: BUY AMZN at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCAMZN$9.95189383477035100
    Price: $9.90
            DDD order: BUY AMZN at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDAMZN$9.90189383477465600
```

  SELL side example:

```
Side: SELL
    Price: $10.10
            EEE order: SELL AMZN at $10.10, Orig Vol: 120, Rem Vol: 10, Fill Vol: 110, CXL Vol: 0, ID: EEEAMZN$10.10189383478470500
    Price: $10.20
            FFF order: SELL AMZN at $10.20, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID: FFFAMZN$10.20189383478919000
    Price: $10.25
            GGG order: SELL AMZN at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: GGGAMZN$10.25189383479288700
```

5) PriceFactory class - *changes*

You should update the PriceFactory class to implement the Flyweight design pattern. This means that the PriceFactory should never generate a Price object for the same value more than once. *(Example: 100 requests for a Price object for $10.00 would generate 1 Price object the first time it was requested, and a reference to that same object the remaining 99 times).*

Your PriceFactory should maintain a HashMap of all Prices created (the key is the cents value, the value is the Price object). Every Price object created should be added to the HashMap. Any requests to make a Price object should first check the HashMap to see if a Price object for that value has already been created – if so, return a reference to that existing Price object. If not, create the Price object, add it to the HashMap, then return the Price object.

The Price class should not need any changes when making the above changes.

6) Testing

Create a new empty class called “Main” and copy into that class the “main” method found in *Appendix C* of this document.

- Section 7 shows the resulting product book contents after each of the test orders are entered.
- Section 8 shows sample results as you might see from your application using the same test orders.

If your application generates the correct expected results using these orders, it is working. If not, you’ll want to look into what has gone wrong and fix the problem ASAP to complete the assignment.

7) Behavior example (Showing test orders and the resulting Product Book).

1) Add Buy 50@$10.00

BUY		SELL	
$10.00	50		

- Incoming Buy order for 50@$10.00 books

2) Add Buy 60@$10.00

BUY		SELL	
$10.00 [50, 60]	110		

- Incoming Buy order for 60@$10.00 books (totaling 110@$10.00 booked)

3) Add Buy 70@$9.95

BUY		SELL	
$10.00 [50, 60]	110		
$9.95	70		

- Incoming Buy order for 70@$9.95 books

4) Add Buy 25@$9.90

BUY		SELL	
$10.00 [50, 60]	110		
$9.95	70		
$9.90	25		

- Incoming Buy order for 25@$9.90 books

5) Add Sell 120@$10.10

BUY		SELL	
$10.00 [50, 60]	110	$10.10	120
$9.95	70		
$9.90	25		

- Incoming Sell order for 120@$10.10 books

6) Add Sell 45@$10.20

BUY		SELL	
$10.00 [50, 60]	110	$10.10	120
$9.95	70	$10.20	45
$9.90	25		

- Incoming Sell order for 45@$10.20 books

7) Add Sell 90@$10.25

BUY		SELL	
$10.00 [50, 60]	110	$10.10	120
$9.95	70	$10.20	45
$9.90	25	$10.25	90

- Incoming Sell order for 90@$10.25 books

8) Add Sell 200@$10.00

BUY		SELL	
$9.95	70	$10.00	90
$9.90	25	$10.10	120
		$10.20	45
		$10.25	90

- Incoming Sell order for 200@$10.00 partially trades, filling 110, leaving 90@$10.00 on the Sell side
- Resting Buy order for 50@$10.00 fully trades.
- Resting Buy order for 60@$10.00 fully trades.
- Nothing left on the Buy side at $10.00 so that entry is removed.

9) Add Buy 200@10.10

BUY		SELL	
$9.95	70	$10.10	10
$9.90	25	$10.20	45
		$10.25	90

- Incoming Buy order for 200@$10.10 partially trades, filling 90, leaving 110@$10.00 on the Sell side
- Resting Sell order for 90@$10.10 fully trades
- Incoming Buy order with 110@$10.10 remaining fully trades
- Resting Sell order for 120@$10.10 partially trades, filling 110, leaving 10@$10.10 on the Sell side
- Nothing left on the Sell side at $10.00 so that entry is removed.

10) Cancel Sell Order 45@$10.20

BUY		SELL	
$9.95	70	$10.10	10
$9.90	25	$10.25	90

- Nothing left on the Sell side at $10.20 so that entry is removed.

11) Add Sell 95@$9.90

BUY		SELL	
		$10.10	10
		$10.25	90

- Incoming Sell order for 95@$9.90 partially trades, filling 70, leaving 25@$9.90 on the Sell side
- Resting Buy order for 70@$9.95 fully trades
- Incoming Sell order with 25@$9.90 remaining fully trades
- Resting Buy order for 25@$9.90 fully trades

- Nothing left on the Buy side at $9.95 so that entry is removed.
- Nothing left on the Buy side at $9.90 so that entry is removed.

12) Add Buy 100@$10.25

BUY		SELL	

- Incoming Buy order for 100@$10.25 partially trades, filling 10, leaving 90@$10.25 on the Buy side
- Resting Sell order for 10@$10.10 fully trades
- Incoming Buy order with 90@$10.25 remaining fully trades
- Resting Sell order for 90@$10.25 fully trades
- Nothing left on the Sell side at $10.10 so that entry is removed.
- Nothing left on the Sell side at $10.25 so that entry is removed.

8) Behavior example (Showing "main" output using the same sample orders).

```
1) ------------------------------------------------------------
ADD: BUY: AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
Product: TGT
Side: BUY
   Price: $10.00
      AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
Side: SELL
      <Empty>

2) ------------------------------------------------------------
ADD: BUY: BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 60, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
Product: TGT
Side: BUY
   Price: $10.00
      AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
      BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 60, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
Side: SELL
      <Empty>

3) ------------------------------------------------------------
ADD: BUY: CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Product: TGT
Side: BUY
   Price: $10.00
      AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
      BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 60, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
   Price: $9.95
      CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Side: SELL
      <Empty>

4) ------------------------------------------------------------
ADD: BUY: DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Product: TGT
Side: BUY
   Price: $10.00
      AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
      BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 60, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
   Price: $9.95
```

CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Price: $9.90
DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Side: SELL
<Empty>

5) ----------
ADD: SELL: EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 120, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.10595271806503100
Product: TGT
Side: BUY
Price: $10.00
AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 60, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
Price: $9.95
CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Price: $9.90
DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Side: SELL
Price: $10.10
EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 120, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.10595271806503100

6) ----------
ADD: SELL: EEE order: SELL TGT at $10.20, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.20595271807621200
Product: TGT
Side: BUY
Price: $10.00
AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 60, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
Price: $9.95
CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Price: $9.90
DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Side: SELL
Price: $10.10
EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 120, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.10595271806503100
Price: $10.20
EEE order: SELL TGT at $10.20, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.20595271807621200

7) ----------
ADD: SELL: FFF order: SELL TGT at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: FFFTGT$10.25595271808923900
Product: TGT
Side: BUY
Price: $10.00
AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 50, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271751644600
BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 60, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
Price: $9.95
CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Price: $9.90
DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Side: SELL
Price: $10.10
EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 120, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.10595271806503100
Price: $10.20
EEE order: SELL TGT at $10.20, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.20595271807621200
Price: $10.25
FFF order: SELL TGT at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: FFFTGT$10.25595271808923900

8) ----------
ADD: SELL: AAA order: SELL TGT at $10.00, Orig Vol: 200, Rem Vol: 200, Fill Vol: 0, CXL Vol: 0, ID: AAATGT$10.00595271810557400

PARTIAL FILL: (SELL 110) AAA order: SELL TGT at $10.00, Orig Vol: 200, Rem Vol: 90, Fill Vol: 110, CXL Vol: 0, ID: AAATGT$10.00595271810557400
FILL: (BUY 50) AAA order: BUY TGT at $10.00, Orig Vol: 50, Rem Vol: 0, Fill Vol: 50, CXL Vol: 0, ID: AAATGT$10.00595271751644600
FILL: (BUY 60) BBB order: BUY TGT at $10.00, Orig Vol: 60, Rem Vol: 0, Fill Vol: 60, CXL Vol: 0, ID: BBBTGT$10.00595271804400700
Product: TGT
Side: BUY
Price: $9.95
CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Price: $9.90
DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Side: SELL
Price: $10.00
AAA order: SELL TGT at $10.00, Orig Vol: 200, Rem Vol: 90, Fill Vol: 110, CXL Vol: 0, ID: AAATGT$10.00595271810557400
Price: $10.10
EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 120, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.10595271806503100
Price: $10.20
EEE order: SELL TGT at $10.20, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.20595271807621200
Price: $10.25
FFF order: SELL TGT at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: FFFTGT$10.25595271808923900

9) ----------
ADD: BUY: BBB order: BUY TGT at $10.10, Orig Vol: 200, Rem Vol: 200, Fill Vol: 0, CXL Vol: 0, ID: BBBTGT$10.10595271817310900
FILL: (SELL 90) AAA order: SELL TGT at $10.00, Orig Vol: 200, Rem Vol: 0, Fill Vol: 200, CXL Vol: 0, ID: AAATGT$10.00595271810557400
PARTIAL FILL: (BUY 90) BBB order: BUY TGT at $10.10, Orig Vol: 200, Rem Vol: 110, Fill Vol: 90, CXL Vol: 0, ID: BBBTGT$10.10595271817310900
PARTIAL FILL: (SELL 110) EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 10, Fill Vol: 110, CXL Vol: 0, ID: EEETGT$10.10595271806503100
FILL: (BUY 110) BBB order: BUY TGT at $10.10, Orig Vol: 200, Rem Vol: 0, Fill Vol: 200, CXL Vol: 0, ID: BBBTGT$10.10595271817310900
Product: TGT
Side: BUY
Price: $9.95
CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Price: $9.90
DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Side: SELL
Price: $10.10
EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 10, Fill Vol: 110, CXL Vol: 0, ID: EEETGT$10.10595271806503100
Price: $10.20
EEE order: SELL TGT at $10.20, Orig Vol: 45, Rem Vol: 45, Fill Vol: 0, CXL Vol: 0, ID: EEETGT$10.20595271807621200
Price: $10.25
FFF order: SELL TGT at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: FFFTGT$10.25595271808923900

10) ----------
CANCEL: SELL Order: EEETGT$10.20595271807621200 Cxl Qty: 45
Product: TGT
Side: BUY
Price: $9.95
CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 70, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
Price: $9.90
DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 25, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$9.90595271805712300
Side: SELL
Price: $10.10
EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 10, Fill Vol: 110, CXL Vol: 0, ID: EEETGT$10.10595271806503100
Price: $10.25
FFF order: SELL TGT at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: FFFTGT$10.25595271808923900

11) ----------
ADD: SELL: CCC order: SELL TGT at $9.90, Orig Vol: 95, Rem Vol: 95, Fill Vol: 0, CXL Vol: 0, ID: CCCTGT$9.90595271826285300
PARTIAL FILL: (SELL 70) CCC order: SELL TGT at $9.90, Orig Vol: 95, Rem Vol: 25, Fill Vol: 70, CXL Vol: 0, ID: CCCTGT$9.90595271826285300
FILL: (BUY 70) CCC order: BUY TGT at $9.95, Orig Vol: 70, Rem Vol: 0, Fill Vol: 70, CXL Vol: 0, ID: CCCTGT$9.95595271804932600
FILL: (SELL 25) CCC order: SELL TGT at $9.90, Orig Vol: 95, Rem Vol: 0, Fill Vol: 95, CXL Vol: 0, ID: CCCTGT$9.90595271826285300
FILL: (BUY 25) DDD order: BUY TGT at $9.90, Orig Vol: 25, Rem Vol: 0, Fill Vol: 25, CXL Vol: 0, ID: DDDTGT$9.90595271805712300

```
Product: TGT
Side: BUY
      <Empty>
Side: SELL
   Price: $10.10
      EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 10, Fill Vol: 110, CXL Vol: 0, ID: EEETGT$10.10595271806503100
   Price: $10.25
      FFF order: SELL TGT at $10.25, Orig Vol: 90, Rem Vol: 90, Fill Vol: 0, CXL Vol: 0, ID: FFFTGT$10.25595271808923900

12) ------------------------------------------------------------
ADD: BUY: DDD order: BUY TGT at $10.25, Orig Vol: 100, Rem Vol: 100, Fill Vol: 0, CXL Vol: 0, ID: DDDTGT$10.25595271827144200
   FILL: (SELL 10) EEE order: SELL TGT at $10.10, Orig Vol: 120, Rem Vol: 0, Fill Vol: 120, CXL Vol: 0, ID: EEETGT$10.10595271806503100
   PARTIAL FILL: (BUY 10) DDD order: BUY TGT at $10.25, Orig Vol: 100, Rem Vol: 90, Fill Vol: 10, CXL Vol: 0, ID: DDDTGT$10.25595271827144200
   FILL: (SELL 90) FFF order: SELL TGT at $10.25, Orig Vol: 90, Rem Vol: 0, Fill Vol: 90, CXL Vol: 0, ID: FFFTGT$10.25595271808923900
   FILL: (BUY 90) DDD order: BUY TGT at $10.25, Orig Vol: 100, Rem Vol: 0, Fill Vol: 100, CXL Vol: 0, ID: DDDTGT$10.25595271827144200
Product: TGT
Side: BUY
      <Empty>
Side: SELL
      <Empty>
```

9) 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.

10) 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.

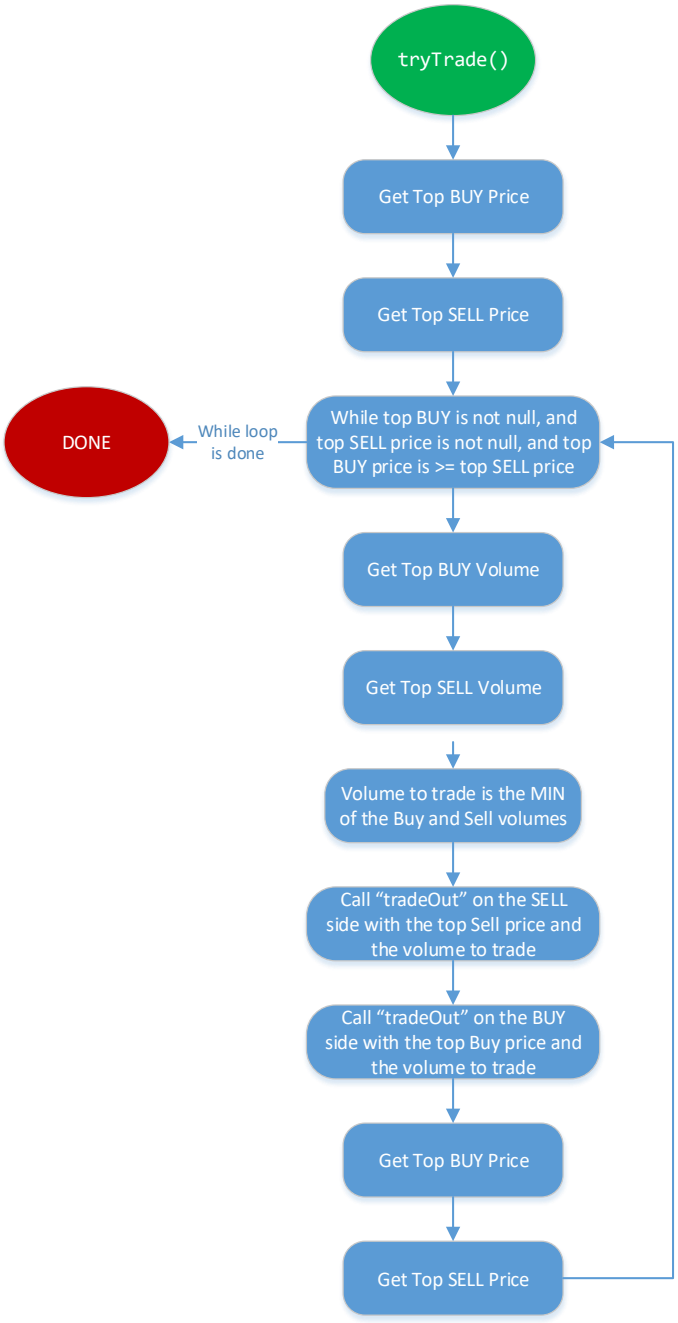
Submissions must reflect the concepts and practices we cover in class, and the requirements specified in this document.

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.

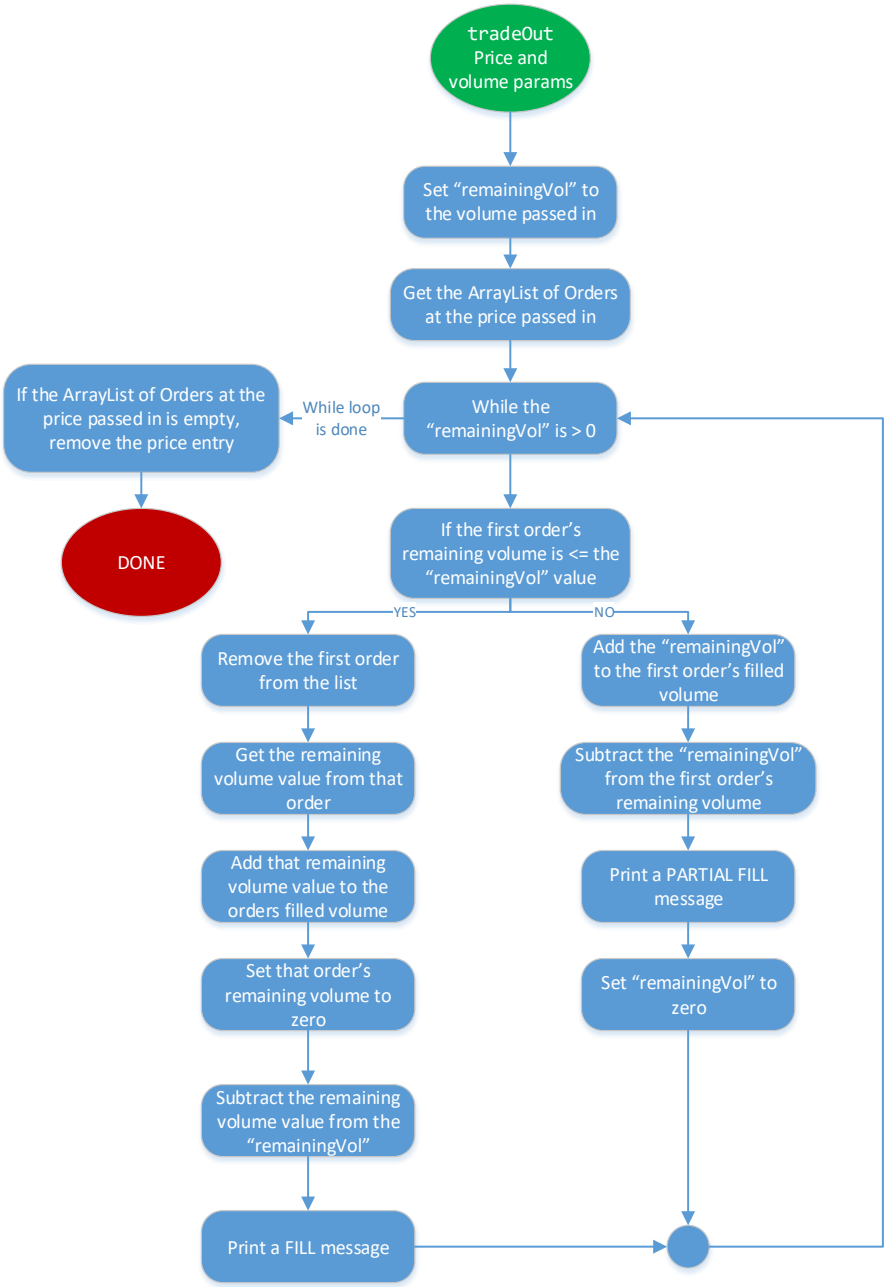
Late submission up to 1 week late will be accepted, though late submissions will incur a 10% penalty (i.e., 10 points). *No late submissions will be accepted for grading after that 1-week time period has passed.*

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

Appendix A: ProductBook `tryTrade()`



Appendix B: ProductBookSide `tradeOut(Price price, int vol)`



Appendix C: The "main" method

```
    public static void main(String[] args) {

        ProductBook pb = new ProductBook("TGT");

        try {
            System.out.println("1) ------------------------------------------------------------");
            Order o1 = new Order("AAA", "TGT", PriceFactory.makePrice(1000), 50, BUY);
            pb.add(o1);
            System.out.println(pb);

            System.out.println("2) ------------------------------------------------------------");
            Order o2 = new Order("BBB", "TGT", PriceFactory.makePrice(1000), 60, BUY);
            pb.add(o2);
            System.out.println(pb);

            System.out.println("3) ------------------------------------------------------------");
            Order o3 = new Order("CCC", "TGT", PriceFactory.makePrice(995), 70, BUY);
            pb.add(o3);
            System.out.println(pb);

            System.out.println("4) ------------------------------------------------------------");
            Order o4 = new Order("DDD", "TGT", PriceFactory.makePrice(990), 25, BUY);
            pb.add(o4);
            System.out.println(pb);

            System.out.println("5) ------------------------------------------------------------");
            Order o5 = new Order("EEE", "TGT", PriceFactory.makePrice(1010), 120, SELL);
            pb.add(o5);
            System.out.println(pb);

            System.out.println("6) ------------------------------------------------------------");
            Order o6 = new Order("EEE", "TGT", PriceFactory.makePrice(1020), 45, SELL);
            pb.add(o6);
            System.out.println(pb);

            System.out.println("7) ------------------------------------------------------------");
            Order o7 = new Order("FFF", "TGT", PriceFactory.makePrice(1025), 90, SELL);
            pb.add(o7);
            System.out.println(pb);

            System.out.println("8) ------------------------------------------------------------");
            Order o8 = new Order("AAA", "TGT", PriceFactory.makePrice(1000), 200, SELL);
            pb.add(o8);
            System.out.println(pb);

            System.out.println("9) ------------------------------------------------------------");
            Order o9 = new Order("BBB", "TGT", PriceFactory.makePrice(1010), 200, BUY);
            pb.add(o9);
            System.out.println(pb);

            System.out.println("10) -----------------------------------------------------------");
            pb.cancel(SELL, o6.getId());
            System.out.println(pb);

            System.out.println("11) -----------------------------------------------------------");
            Order o10 = new Order("CCC", "TGT", PriceFactory.makePrice(990), 95, SELL);
            pb.add(o10);
            System.out.println(pb);

            System.out.println("12) -----------------------------------------------------------");
            Order o11 = new Order("DDD", "TGT", PriceFactory.makePrice(1025), 100, BUY);
```

```
            pb.add(o11);
            System.out.println(pb);

        } catch (DataValidationException | OrderNotFoundException e) {

            System.out.println("Unexpected exception occurred: " + e.getMessage());
            e.printStackTrace();

        }
    }
```