CMC Development Task (Core Services)

Prior to your technical interview at CMC, we would like you to implement a small test application. Reviewing your implementation and discussing how it can be enhanced may form part of the interview. The task is to build order books from a stream of orders and be able to calculate the current price based on the best available orders.

Order Book Definition

An **Order** (identified by an **orderId**) is a request to buy or sell (**Side**) a certain **quantity** of a financial instrument (identified by a **symbol**, e.g. a stock or share), at a certain **price**.

- The price of a **BUY** order is called the bid price.
- The price of a **SELL** order is called the ask price.

An ‘order book’ is the collection of all buy and sell orders for a single symbol. It contains two sides – bid (for buy orders) and ask (for sell orders). Each side contains multiple levels which are sorted by price so that orders with the highest bid price or lowest ask price are on top. Orders may be removed and the price/quantity of an order may be changed after creation.

For example, an order to buy 12 Microsoft shares (symbol MSFT) for a price of 19 means that a trader wants to buy 12 MSFT shares for $19.00 per share. Multiple traders place individual orders at different prices and quantities, leading to the buildup of an order book.

Typically, order books are represented by ‘order book level’. An order book level is the total quantity of all orders with the same price and side for a given financial instrument. Orders that have the same price are aggregated; each Level has the price and the total size of all orders at that price. Table 1 shows what the MSFT order book might look like after several orders have been created.

A few important notes:

1. Order books are built for each individual symbol. It is a mistake to aggregate orders for different symbols in the same order book.
2. A typical order book might have anywhere from 5 to 500 price levels at any time.
3. It is common for new price levels to be created during the day as the stock price moves.
4. A typical order might have 4 or more modifications applied to it before being removed.

Order Count	Ask Quantity	Ask Price	Level	Bid Price	Bid Quantity	Order Count
2	12	$19	1	$15	10	1
2	17	$21	2	$10	26	2
1	7	$22	3			

Table 1 – MSFT order book for the orders in the ExampleData class

Your requirements

You should implement the **OrderHandler** interface according to the rules outlined below.

addOrder, **modifyOrder**, and **removeOrder** should all update the internal state of your **OrderHandler** and should be **thread safe**.

getCurrentPrice should return the best available average price for the given symbol, quantity and side. The best price is obtained from first order book level (Level 1). If the total quantity in Level 1 is less that the requested quantity, we must go to the next level and calculate the average price. Here are some examples using the MSFT order book in Table 1:

```
getCurrentPrice("MSFT", 6, Side.SELL)  = $19

getCurrentPrice("MSFT", 17, Side.SELL) = ($19*12 + $21*5)/17 = $19.588…

getCurrentPrice("MSFT", 30, Side.SELL) = ($19*12 + $21*17 + $22*1)/30 = $20.233…

getCurrentPrice("MSFT", 10, Side.BUY)  = $15
```

Note that for the sell orders, the price is getting worse as the quantity increases (from the point of view of a prospective buyer). Also, assume that getCurrentPrice will always be called with a quantity that can be filled by the current order book. Finally, note that while the order price is an int, the getCurrentPrice method returns a double.

You should not simply loop through all orders every time getCurrentPrice is called, i.e. try to maintain some information in real time and update it every time addOrder, modifyOrder or removeOrder is called.

Please consider the operational aspects of the implementation.

ExampleData::buildExampleOrderBookFromReadMe submits a series of orders that should produce the order book shown in Table 1.