Order matcher

Introduction,
Most financial markets have moved away from the traditional open out-cry mode (with people shouting at each other) towards an electronic system with participants sitting behind their computer screens. Only electronic systems are considered capable of meeting today's challenges of globalisation and rapidly increasing turn-over. This evolution is by no means limited to financial markets, but virtually applies to all commodities which can be bought and sold.

Electronic trading has given rise to a new price-fixing mechanism. In open out-cry trading the price derives from a process in which every seller seeks the highest bidding buyer and every buyer looks for the seller asking for the lowest price; this process is also referred to as two-sided auction. In an electronic exchange an order book is kept: All buy and sell orders are entered into this order book and the prices are set according to specific rules. Bids and asks are matched and trades occur. We would like you to design a program, called OrderMatcher, that can determine in real-time the current market price and combine matching orders to trades. Each order has a volume and a price.

Kungsgatan 2, SE-111 43 Stockholm, Sweden, Phone: +46 8 588 977 00
405 Lexington Avenue, New York, NY 10174, USA, Phone: +1 212-907-6489
info@aphelion.se, www.aphelion.se

Input

There are three types of commands: BUY, SELL and PRINT. The commands should be entered at the console (stdin) and one command is entered on each line, thus a command ends with a newline.

BUY

"BUY volume@price" enters a buy order with the specified volume and price. Both price and volume are positive (> 0) integers. Example "BUY 1000@25" means buy 1000 shares at the price of 25.

SELL

"SELL volume@price" enters a sell order with the specified volume and price. Both price and volume are positive (> 0) integers. Example "SELL 500@30" means sell 500 shares at the price of 30.

PRINT

"PRINT" means that all orders in the order book should be printed to the console (stdout), sorted with the best price at the top (lowest for sell and highest for buy). The order output is the same as the order input, example

```
--- SELL ---
SELL 100@55
SELL 500@67
SELL 200@88
--- BUY ---
BUY 100@44
BUY 300@33
```

Trading Rules

It is a match if a buy order exist at a higher price or equal to a sell order in the order book. The volume of both orders is reduced as much as possible. When an order has a volume of zero it is removed. An order can match several other orders if the volume is big enough and the price is correct. The price of the trade is computed as the order that was in the order book first (the passive part).

The priority of the orders to match is based on the following:
1. On the price that is best for the active order (the one just entered)
2. On the time the order was entered (first come first served)

Whenever there is a trade a "TRADE price@volume" should be printed out to the console (stdout).

Example ("> " is the prompt):

```
> SELL 100@10
> SELL 100@15
> BUY 120@17
TRADE 100@10
TRADE 20@15
> PRINT
--- SELL ---
SELL 80@15
--- BUY ---
```

In the example above the buy order first matched the sell order at the price of 10 which is the best sell order and then a part of the sell order at the price of 15. If two orders have the same price the order entered first have higher priority (is better) and will be matched first and should be printed at the top.

Implementation

The OrderMatcher should be implemented in Java (preferred) or C++ and should be possible to compile and execute on Linux.

Regards,

Aphelion development