Pre-Interview Exercise

We have been asked to implement an exchange system which matches orders on stocks. The functionality we have been asked to implement is:

1. **Add an order**
   - An order consists of direction (buy/sell), RIC (Reuters Instrument Code), quantity, price and user
   - When an order is added, it should be compared against existing open orders to see whether it can be matched. Two orders match if they have opposing directions, matching RICs and quantities, and if the sell price is less than or equal to the buy price
   - When two orders are matched they are said to be 'executed', and the price at which they are executed (the execution price) is the price of the newly added order
   - If there are multiple matching orders at the same price for a new order, it should be matched against the earliest matching existing orders
   - If there are multiple matching orders at different prices for a new sell order, it should be matched against the order with the highest price
   - If there are multiple matching orders at different prices for a new buy order, it should be matched against the order with the lowest price
   - Executed orders are removed from the set of open orders against which new orders can be matched
2. **Provide open interest for a given RIC and direction**
   - Open interest is the total quantity of all open orders for the given RIC and direction at each price point
3. **Provide the average execution price for a given RIC**
   - The average execution price is the average price per unit of all executions for the given RIC
4. **Provide executed quantity for a given RIC and user**
   - Executed quantity is the sum of quantities of executed orders for the given RIC and user. The quantity of sell orders should be negated

See examples on the following page

Example sequence of orders:

Open VOD.L BUY interest	Open VOD.L SELL interest	Average VOD.L exec. price	Executed quantity for VOD.L, User1	Executed quantity for VOD.L, User2
New Order: SELL 1000 VOD.L @ 100.2 User1				
	1000 @ 100.2		0	
New Order: BUY 1000 VOD.L @ 100.2 User2 // matches existing buy order, executed @ 100.2				
		100.2000	-1000	1000
New Order: BUY 1000 VOD.L @ 99 User1				
1000 @ 99		100.2000	-1000	1000
New Order: BUY 1000 VOD.L @ 101 User1				
1000 @ 101 1000 @ 99		100.2000	-1000	1000
New Order: SELL 500 VOD.L @ 102 User2				
1000 @ 101 1000 @ 99	500 @ 102	100.2000	-1000	1000
New Order: BUY 500 VOD.L @ 103 User1 // matches existing SELL @ 102, executed @ 103				
1000 @ 101 1000 @ 99		101.1333	-500	500
New Order: SELL 1000 VOD.L @ 98 User2 // matches existing BUY @ 101, executed @ 98				
1000 @ 99		99.8800	500	-500

Please provide an implementation of the requirements above in either Java or Scala. No persistence or UI is required; an in-memory solution providing an API is absolutely fine. The solution should include everything you would check in to your source code management system, and the code should be of production quality.

If you find any requirements are unclear, please state any assumptions in a readme file included with your solution.

Please zip up your entire solution and email it to your agent who will pass it on to us.