Stock Exchange

Version 1.0

Description

The Stock Exchange handles trading*shares*companies.*Trading system*shares allows investors to place*orders*purchase or sale. An order is a declaration of intent to buy or sell shares of a selected company. Each order must contain:

- order type: purchase / sale,
- action identifier (non-empty ASCII AZ string no longer than 5),
- number of shares (positive integer),
- price limit (positive integer).

We assume that in one order you can express the desire to buy/sell shares of one company with a price limit (upper in the case of a buy order, lower in the case of a sell order). A transaction occurs when there is a match between the buy and sell orders of the same company, i.e. when the buy offer price is equal to or higher than the sell offer price.

For example, let's imagine the stock market situation presented below.*order sheet*shares of a certain company:

Purchase				**Sale**		
Order number	**Number of shares**	**Price**		**Order number**	**Price**	**Number of shares**
4	**100**	**125**		**1**	**123**	**10**
	40	122		**2**	**124**	**25**
	10	121		**3**	**125**	**30**
	30	120			126	20
	10	119			127	60
	20	118				

For the order book shown, the only buy order that can be filled is the buy order (#4) for 100 shares with a limit price of 125. This order will be partially filled in three*transactions*(in this order): the investor will purchase 10 shares from order no. 1, 25 from order no. 2 and 30 from order no. 3. In total, the investor will purchase 65 shares. The remaining 35 shares with a limit of 125 will be waiting for execution and such an order will remain in the order book.
In our trading system, time is measured in rounds[1]. The order of order execution is determined primarily by the price reported by the investor (the buy order with the highest limit is matched with the sell order with the lowest limit), and

[1]This is a simplification for the purposes of our task.

then the order placement round. In the case of multiple orders placed in the same round, the order in which they were submitted is decisive. So in our discussed example, a buy order of at most 125 (order no. 4) will be first matched with a sell order of at least 123 (order no. 1).
When executing orders, transactions are concluded at a price equal to the limit price of the order that was placed earlier, and in the case of orders placed in the same round, the order in which they were submitted is decisive. For example, if there is an earlier sell order in the spreadsheet with an execution limit of 100 and a later buy order with a limit of 102, then at the end of the round a transaction will be concluded and the investor will acquire shares at a price of 100, i.e. at the execution price of the older sell order. Therefore, in our previously discussed example of buy order no. 4, the investor will acquire 10 shares from order no. 1 at a price of 123, 25 shares from order no. 2 at a price of 124 and 30 shares from order no. 3 at a price of 125.

Investors have several options available to determine the order validity period:

- **immediate order**-the order must be executed, at least partially, in the same round in which it was entered into the order book. The unexecuted part of the order is eliminated by the system. We allow several possibilities here - the order cannot be executed even partially in the current round and is eliminated, the order is executed partially in the current round and the remaining part is eliminated, and the order is executed in full in the current round and is eliminated from the system as executed.

- **an order without a specific expiry date**-remains in the system until it is completed in full (as a result of one or more transactions).
- **execute or cancel the order**-the order must be executed in its entirety (possibly in multiple transactions) in the same round in which it was entered into the order book (if this is impossible, it is eliminated from the system).
- **order valid until the end of a specific round**-the order remains in the system until the end of the nth round, unless it is fully executed earlier.

Implement a trading system that polls all investors (in random order for each round) for their own investment decisions during each round. Specifically, during one round, investors can:

- ask the system for the current tour number,
- query the transaction system for the round number and price of the last transaction for the shares of the indicated company (any number of times),
- for a selected listed company, decide to place at most one order to buy or sell its shares (if he or she would like to place more orders, e.g. for shares of other companies, he or she must do so in subsequent rounds).

After accepting investment decisions from all investors in one round, the trading system executes orders according to its rules described earlier.

Two types of investors should be implemented, guided by different investment strategies:

- **RANDOM**-investor making random investment decisions (implement any strategy - e.g. random buy/sell order type, random company, random number of shares, random price limit). The drawing takes place according to the conditions given in the content.

- **SMA**-an investor who makes decisions based on technical analysis of individual shares based on the Simple Moving Average (SMA n) indicator - an arithmetic moving average of prices from the last n rounds (in our task we use n=5 and n=10). The price in a given round is assumed to be the price of the last transaction of shares of a given company, this transaction took place in one of the previous rounds (if there were no transactions for several rounds, the price of shares of a given company remains unchanged).
  A buy signal occurs when the longer average - SMA 10 crosses the shorter one - SMA 5 from below, while a sell signal occurs when SMA 5 crosses SMA 10 from above.[2]. The investor starts making decisions based on the signal from the moment he can calculate SMA 10 (at least from the 10th round).

Of course, there can be more types of investors, the solution should allow you to easily add them, and you can also include (in addition to, not instead of, the two mentioned) other investors in your solution.
In order to ensure efficient trading conditions on the stock exchange, we assume that share prices are positive natural numbers and that share price limits in placed orders are positive and cannot differ from the price of the last transaction by more than 10 units.
The trading system ensures that when placing an order, the investor has the appropriate assets in their portfolio (shares in the case of a sell order or cash in the case of a buy order). We do not allow so-called short selling, i.e. selling shares that we do not physically own.

If, when attempting to execute an order, the investor does not have the necessary assets in their portfolio, the order is canceled.

Your program should take command line arguments specifying the input file and the number of simulation rounds (we start the simulation from round 0).

Example of program invocation: java
GPWSimulation input.txt 100000
Where*input.txt*is an input file defining the initial state, and*100000*is the number of simulation rounds.

Input file example(lines starting with the # character should be ignored by the parser):

# we have 6 investors: 4x RANDOM, 2x SMA RRRRSS

# 3 types of stocks traded with prices of recent transactions APL:145 MSFT:300
GOOGL:2700
# initial portfolio balance (identical for all investors) 100000 APL:5 MSFT:15 GOOGL:3

Your program should verify the correctness of the input data, in particular check whether the portfolio contains shares that are in circulation and print them to the standard

[2]Detailed description of the SMA signal generation method: https://www.bdm.com.pl/ispag/encat/sma.html

output the final portfolio status for each investor after all simulation rounds have been completed, e.g.:

```
100000 APL:5 MSFT:15 GOOGL:3
100000 APL:5 MSFT:15 GOOGL:3
100000 APL:5 MSFT:15 GOOGL:3
100000 APL:5 MSFT:15 GOOGL:3
100000 APL:5 MSFT:15 GOOGL:3
100000 APL:5 MSFT:15 GOOGL:3
```

For the purposes of writing and solving, we recommend adding the ability to list all stock market operations. It can be assumed that the number of simulation rounds, the number of investors, the stock price, and the contents of investors' portfolios during the simulation are within the range of type int. The simulation invariant is the total sum of all shares and cash in portfolios.
Your solution should include implementing JUnit tests and testing the correctness of the trading system by comparing the results obtained in the simulation with the expected results. JUnit tests should cover various cases, including scenarios with different numbers of investors, different types of investors, and different initial conditions.

Make sure your solution is object-oriented, in particular that you can easily add other types of orders and types of investors.

Source:
https://www.gpw.pl/pub/images/prezentacje/system_obrotu.pdf