A Simple Stock Exchange

You're given the task of creating a simple stock exchange. It is so simple that it only supports a limited set of functionality. However, there might be changes in the future.

Here are the functions that it should support:

1. User is able place a buy order for a particular stock
2. User is able place a sell order for a particular stock
3. User is able to get the bid/ask and last price from the exchange
4. User is able to place a limit order
5. User is able to place a market order
6. The exchange should be able to resolve the order. E.g. placing a buy limit order at price of $10 when the asking price is $9.99 will complete the trade.
7. The user should be able to view all order status. E.g. filled, partially filled, pending.
8. The user is able to exit the exchange program. (In real life, you exit the client)

Below is a sample run of the program. User input is prefixed with "Action:".

Acton: BUY SNAP LMT $30 100
You have placed a limit buy order for 100 SNAP shares at $30.00 each.

Action: VIEW ORDERS
1. SNAP LMT BUY $30.00 0/100 PENDING

Action: BUY FB MKT 20
You have placed a market order for 20 FB shares.

Action: VIEW ORDERS
1. SNAP LMT BUY $30.00 0/100 PENDING
2. FB MKT BUY 0/20 PENDING

Action: SELL FB LMT $20.00 20
You have placed a limit sell order for 20 FB shares at $20.00 each

Action: VIEW ORDERS
1. SNAP LMT BUY $30.00 0/100 PENDING
2. FB MKT BUY $20.00 20/20 FILLED
3. FB LMT SELL $20.00 20/20 FILLED

Action: SELL SNAP LMT $30.00 20
You have placed a limit sell order for 20 SNAP shares at $30.00 each

Action: VIEW ORDERS
1. SNAP LMT BUY $30.00 20/100 PARTIAL
2. FB MKT BUY $20.00 20/20 FILLED
3. FB LMT SELL $20.00 20/20 FILLED
4. SNAP LMT SELL $30.00 20/20 FILLED

Action: SELL SNAP LMT $31.00 10
You have placed a limit sell order for 10 SNAP shares at $31.00 each

Action: QUOTE SNAP
SNAP BID: $30.00 ASK: $31.00 LAST: $30.00

Action: QUIT
No output

Here are some general instructions on how to approach this exercise:

- You have **two** weeks from the date you received the assessment to implement the solution.
- You may use any one of the languages: Python, Kotlin, Java.
- Solving the problem itself is easy but don't treat this as a '*code golf*' exercise.
- A good submission demonstrates sound object oriented (or functional) design, SOLID principles and patterns, thought about edge and failure cases, clean readable code and is thoroughly covered by tests.
- We would rather you produce a polished solution, than a fast one, but equally there's no need to over engineer, so do take your time and focus on high quality code before submitting it.
- Add a readme file with instructions on how to run the program
    - If there are special cases etc., which are not covered as part of the requirement, please make your own decision on how this should be handled.
    - If you do make any assumptions, please detail them along with how you handled them in the readme file.
- Both the test and implementation code will be assessed.
- Assessment will primarily, but not solely, based upon:
    - How comprehensively the code is tested.
    - The design choices made when implementing the solution (maintainability, legibility, refactorability, etc.)
    - Exception and special case handling.
    - You should aim to implement up to a level where you would be happy to have somebody judge your result.
- Please refrain from plagiarizing existing solutions that may have been shared on the Internet.
- **Once you are done, please upload the source code to Dropbox (as a single archive) and send us the link.**
    - Please keep the archive simple with no credential/ access/ passwords in order to download.
    - Acceptable archive formats are **zip, tar, gz and tar.gz**
    - **Kindly note that your submitted archive doesn't exceed 10 Mb.**