**Netty.io Matching Engine (TCP Client & Server). Implement a TCP client and a TCP serve  
r (a basic 101 matching engine) using Netty.io.**

**Requirements**

* The TCP client will send MARKET orders (only BUY or SELL orders).
* The TCP server maintains an order book. When the TCP server is loaded for the first time, initial orders are pre-set in the order book, consisting of BUY and SELL orders sorted by price

**Order Matching Rules and engine logic:**

* The server maintains an order book, for initial data.
* Buy orders are sorted by highest price first.
* Sell orders are sorted by lowest price first.
* FIFO applies for orders at the same price.
* When a MARKET order arrives, the server:
  + Checks the order book for a match.
  + If a match is found (there are bids or asks), execute the trade at the best available price.
  + If no match exists, reject the order.
* BUY orders match the lowest available SELL order first.
* SELL orders match the highest available BUY order first.
* If multiple orders exist at the same price, FIFO (First-In, First-Out) applies.
* Only fully matched orders are processed (no partial fills).
* If for MARKET orders (sent from the client) no match exists, they are rejected.

**Order Format JSON (Client → Server)**

Clients send MARKET orders in the following JSON format:

{ "type": "BUY", "quantity": 10, "accountId": "1233" }

{ "type": "SELL", "quantity": 5, "accountId": "5678" }

**Execution Report Format (Server → Client)**

If an order is fully matched, the server sends an execution report to buyer (or seller):

{

"type": "exe\_report",

“initialQuantity”: “100”

"executedPrice": 100,

"executedQuantity": 10,

"accountId": "1233",

"status": "FILLED"

}

If an order is rejected, the engine sends a response like this  
{

"type": "exe\_report",

“initialQuantity”: “100”

"executedPrice": null,

"executedQuantity": null,

"accountId": "5678",

"status": "REJECTED"

}