Disclaimer

Introduction

We would like you to implement a simple Spring-based microservice (delivered in the form of a Maven project) according to instructions given below.

System Behavior

1. Admin user opens (creates) an order book for a given instrument (string). In return gets a unique order book identifier.
2. Regular users start adding (creating) buy orders to the selected open order books. An order has a quantity (long), type (market or limit) and a price (number) if order is of limit type. In return, users get a unique order identifier.
3. Admin user closes an order book, passing in the process total executed quantity (long) and execution price (number). Distribution algorithm kicks in. First validates orders (limit price must be equal or greater than execution price). Then distributes executed quantity proportionally among valid orders, but not above their ordered quantity.
4. Regular users query their order status, price and effective executed quantity.

Example (for a single order book)

Orders:
```
id: 1, quantity: 50, type: limit, price: 17.0
id: 2, quantity: 130, type: limit, price: 14.0
id: 3, quantity: 150, type: market
```

Execution (order book closure):
```
quantity: 80, price: 15.0
```

Query result (orders):
```
id: 1, status: executed, quantity: 20, price: 15.0
id: 2, status: invalid, quantity: 0, price: 15.0
id: 3, status: executed, quantity: 60, price: 15.0
```

Technical Requirements

- Expose REST API.
- Assume that users (admin, regular) will call only endpoints dedicated to them.
- Persistence is not required; in-memory solution (H2, HashMap) will suffice.
- Write tests (focus on distribution algorithm).
- Add Swagger.
- Use Java 8.

Delivery

Pack your Maven project into a zip archive, but without any binaries (will not get through email filter otherwise).