Financial Computing H/W #4

2012, Spring

(Prof. Eran Fishler)

(Grader : Ankit Mittal / Shinan Zhang)

Name : **Jinil Jang**

Univ ID : **N10917970**

1 . Develop Environment

- Program Language : Java (JDK 1.7.0)
- Program IDE : IntelliJ IDEA 11.0.2

2. Purpose of each Class

- Book : Structure of a symbol book
  Contain Ask Book and Bid Book of the symbol
- MarketOrder : Implements of IMarketOrder interface
  Class for the Market Order
- LimitOrder : Implements of ILimitOrder interface
  Class for the Limit Order
- CxRxOrder : Implements of ICxRxOrder interface
  Class for the Cancel / Replace Order
- Order : Wrapper structure of various order type
- OrderList : Order List of same price orders
- Simulator : Simulate server to transact orders.

- Trader : Observer of the Simulator

  Trader can replay with the saved transaction data in the Simulator

  Trader can transact in the Simulator with other Traders

- Runner : Test main class

  Print the each transaction result

- SilentRunner : Test main class

  Not print any transaction result

3. How to run the program

I. Java Run

1) unzip the file, and go into "src₩edu₩nyu₩cims₩compfin09₩hw4" directory.

2) Type "javac -d . *.java" on the command line,

then you can get all the classes in a subordinate folder, "edu".

3) Type "java -classpath . edu.nyu.cims.compfin09.hw4.Runner" to run a test.

If you want to silent mode, input this command.

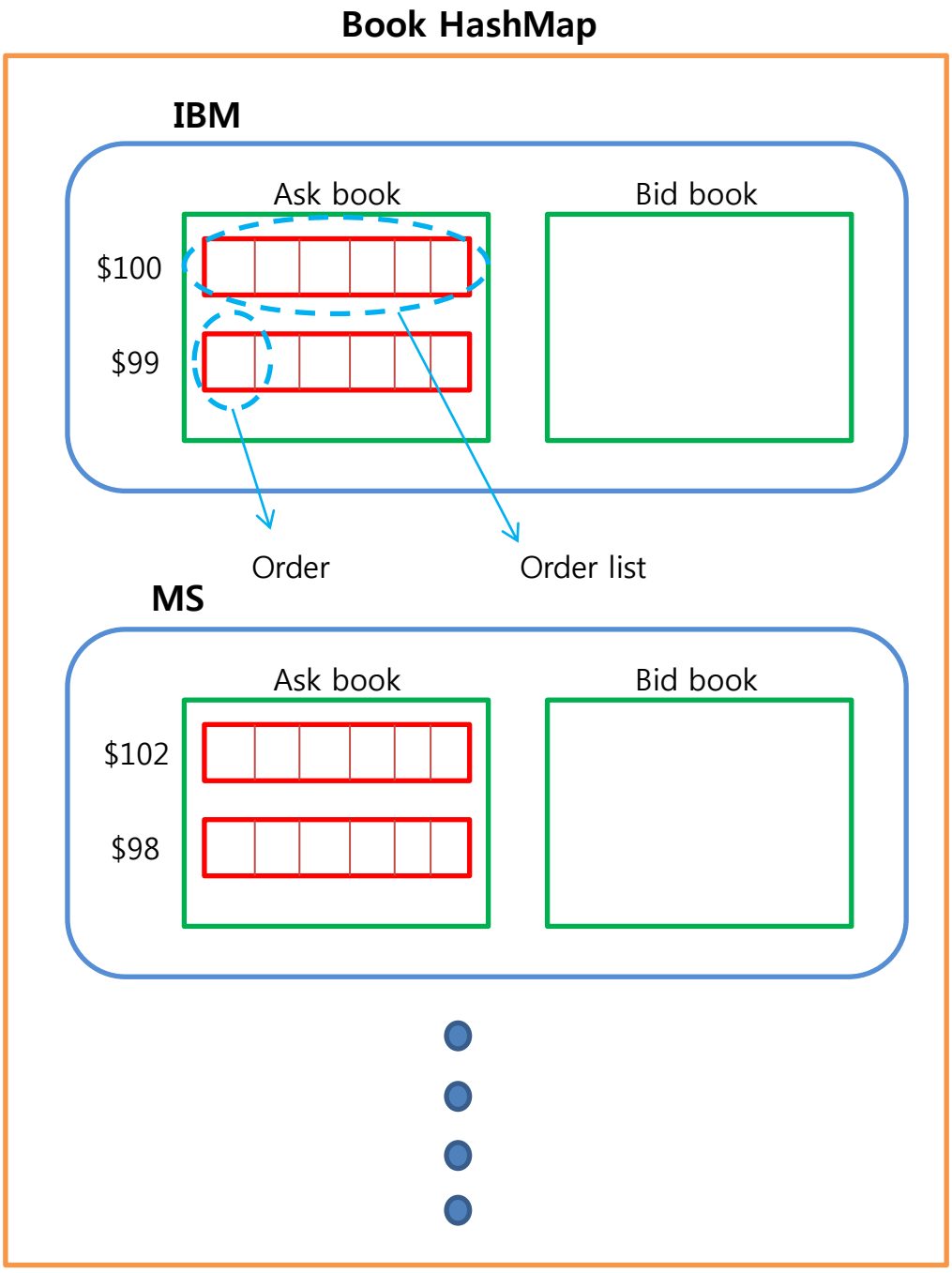
"java -classpath . edu.nyu.cims.compfin09.hw4.SilentRunner"

II. If you use intelliJ IDEA, you can run this program like this :

① Press 'F9'

② Press '1. Run'

4. Design



(1) Book HashMap : HashMap < Symbol, Book >

(2) Ask book : TreeMap < Price, OrderList >

Bid book : TreeMap < Price, OrderList >

(3) Order List : HashMap < Order ID, Order >

5. Special things in my program.

- **Decorate Design Pattern**

  - I made many wrapper classes like upper diagram. Each Limit / Market / CxRx orders are wrapped with Order. And the Orders are wrapped with Order List if they have same price. The bid book and ask book are consist of these Order List. Like this way, all objects are managed in upper wrapper object. With this Decorate design patter, each objects are much conceptualized, and more understandable. And the connection of upper objects' functions and lower objects' functions are much better in program.

- **Object Design Pattern**

  - I expanded this program much more than the homework description. That is, this program is not just for the replay of the transaction data. In the program, the Simulator class means the server of transaction system, which is subject. And the Trader is object. Thus the Trader registers to the server, and got the notification from the server. The Trader can trade with other Traders in the Simulator as well as replay with saved transaction data. And a lot of Traders can register this Simulator, and they can choose whether get the notification or not.

- **Object Oriented Programming**

  - I considered the ability of the extension in this program. This is because, this program reflects on real trading world. Thus, it is always possible to augment or modify in both order types and trading rules. And also, there are exception processing as many as possible to prevent potential error.

6. JavaDec

- The Java Document is located in the 'homework4/doc' folder. Please read it for more specific information of this program. To display in the JavaDoc, all functions' controlling access are expressed in 'Public' (Other controlling access does not display in JavaDoc). But in the program source, it does not. That is, I use 'Public', 'Private' and 'Protected' as function's controlling access appropriately.

Thank you very much !