Final Project: Stock Market

-------Jingjing Wang

The ultimate goal of this project is to use distributed objects to implement a stock market game for several human players. The implementation has been done by the use of Java RMI and Callback technology. And also in this final project, I have used the central server algorithm for the mutual exclusion. And the server designed in this project also has the function of the fault tolerant. The detailed information will be described as below.

1 The design and the architecture of this game

In this design, there are four main classes which interoperate with each other.

The first class is Bank. A player can request a new account by use of his name, and deposit and withdraw money from the account. In this game, one player is allowed to have at most one account in the bank. If the player has deposited the money in this account, then once every 5 minutes during the game, the money in the account accumulates 3% interest. Account class is used to manage the financial action of deposit and withdraw for each account. In this class, I use the lock technology to do mutual exclusion for the action of deposit and withdraw. When the balance of the account is less than the number of the withdraw, then the withdraw action needs to wait until there is enough money in this account.

The second one class is Company. A company has a name and issues stock. Once a minute, the company's price of the stock is adjusted based on a random process. In this case, I defined a random variable for this two actions. That is if this variable equals with zero, then the price of the stock will increase by 1% to 3%, and if it equals one, then the price will decrease by 1% to 3%. Also the adjust of the price of the company is also based on the law of supply and demand, that is if a player buys the stocks from the company, then the price of the stock will be up, and vice versa. The finally price after each transaction is based on the following rule:

The price = (the bid posted by the player X the stock number posted by the player + the stock price of the company before the transaction X the stock number of the company) / (the stock number posted by the player + the stock number of the company).

When the price of the stock of the company or the number of the stock is changed, then the update method will be incurred and the current information will be put into
the StockExchange which the players are able to notice. So in this case, at the beginning of the game, the company will need to register into the StockExchange. A company buys or sells shares to a player by accepting or rejecting the player’s bid. The process of deciding to accept or reject is random. Initially each company has 1000 shares of the stock which starts at $30 a share. There are three companies for this game and each company has one kind of stock.

The third main class is the player. In this game, we could allow several players to take part in (of course, the more the better). A player starts with $2000 and 20 shares for each stock (because I think it can make the game be more interesting) and buys and sells stock from the companies or other players by trading through the StockExchange. The goal of a player is to make more and more money. A player buys or sells shares from another player or a company by posting bids on the StockExchange. Since the player can get the updated information of the StockExchange and enter the bank. So at first, the player should register it into both server and the bank. There are several functions in the player, I will describe them in the Player Graphic Interface.

The finally main class is called Stock Exchange. Players and companies must register with the stock exchange. All trading of the stocks must go through the centralized stock exchange. In the Stock Exchange, I use three ArrayList to record the stock information. They respectively includes the current information of the company, sellers and buyers. For the company, one ArrayList records the name of the company, the name of the stock, the price of the stock, the number of the stock and the callback object of the company which is used to update the information of the company and remotely incurs some methods in the company object. For the players, the other two ArrayLists respectively record the information of the sellers and buyers. That means if one wants to sell, then the stock Exchange will record the information including the name of the player, the name of the stock that the guy wants to sell, the sellprice (bid), the number of the stock as well as the callback object of the player. The rules or the strategy of the transaction will be described detailed as below. Finally, I design the total time of the game is 21 minutes. Once the time is out, the Stock Exchange will callback the players listed in the callback objects who is the winner. The decision of the winner is the player with the highest total value in cash and deposit in the bank as well as stock value. Meanwhile, the Stock exchange reminds the players the time of the game on every 7 minutes. Both of them are implemented by the timers and callback technology.

2 The strategy or the rules of the game

There are several rules or the strategy in of the game to make the game more interesting and realistic.
1) A player can not buy or sell shares of stock that do not exist and he or she must have enough cash to purchase the desired shares.

2) If a player wants to sell one kind of the stock, only when he or she has enough number of the stock to sell.

3) The finally price between a buyer and seller follows the following rule:

   The price= (the bid posted by the player1 X the stock number posted by the player1 +the bid posted by the player2 X the stock number of the player2) /(the stock number posted by the player1+ the stock number of the player2).

4) When the player wants to buy(sell) the stock, the Stock Exchange server will automatically find the post information stored in it ,and find one who has posted the lowest sellprice (highest buyprice ). Of course, the buyprice must be greater than the sellprice. If it can not be satisfied, then no transaction can occur and the buy/sell information will store into the stock exchange.

5) Since the company is designed to be able to accept the bid from the players randomly, so if the company accepts the bid of the player, then it also need to follow the rule (4). However, if it rejects, then the stock exchange will find the second most available choice for the buyer or the seller.

6) If the buyer(the seller) posts the number of the stock that is larger than the one sellinformation (buyinformation), then the stock exchange will find several players that stores the sellinformation (buyinformation) for the buyer(the seller) to stratify the buyer(seller) ’s request as much as possible. However, if the buyer’s(the seller’s) request for the number of the stock is larger than the total stock number of the sellinformation(buyinformation) in the stock exchange. Then the stock exchange will put the rest stock number of the buyer(the seller) into the buyinformation(sellinformation).

7) In this game, there will be the following case that we must prevent. If in the timestamp t1, one buyer called player1 posts the buyinformation called T1 into the stock exchange. And in the timestamp t2, this buyer spends a lot of money on another transaction called T2. However, in timestamp t3, another player called player2 wants to sell some stock that satisfy T1, but at this moment the stock exchange checks the player1 does not have enough money to buy. Then the stock exchange will consider this transaction can not be finished, and then it removes the buyinformation of the player1 in it, and records the sellinformation of the player2. In the case, the relationship of the timestamp is as below: t1<t2<t3.
3 The Graphic User Interface of the game

The following is the Graphic User Interface for Stock Exchange.

“Transaction information” shows the information of each finished transaction. And “Financial Information” shows the information including current price of the stock in the companies and the posted information from the players which are very beneficial and convenient for the player to be familiar with the current situation in the market. When the player see this graphics, he or she need to press “Start Game” button to start the game. There are three different kinds of the stock for the player to choose—-stock1, stock2 as well as stock3. The button called “request player information” is useful when the player wants to know whether there is some other players have posted some proper buy information or sell information in the stock exchange. The button named “request company information” is convenient to check the information of the company. The button called “show my information” is for checking the information of his or her such as the cash, the hold of the stocks and the deposit.
If the player wants to do some actions that has something to do with bank, then he or she needs to press the button called “Enter the bank”, and he or she will see another graphic interface. See below:

Before doing some actions, the player should first press “CreateAccount ” as told in the suggestion.

4 The Algorithm designed in this game

Since a collection of the players share resources, mutual exclusion is needed to prevent the interface and ensure consistency. For this reason, I used the Central Server Algorithm for managing a mutual exclusion token for a set of processes to enter the stock exchange. The algorithm is as below:

A process sends a request message to server and awaits a reply from it.

If a reply constitutes a token signifying the permission to enter the stock exchange.
If no other processes has the token at the time of the request, then the server replied immediately with the token.

If the token is currently held by other processes, the server dose not reply but queues the request.

Client on exiting the stock exchange, a message is sent to the stock exchange server, giving it back the token.

5 The fault tolerate designed in this game

In this game, the stock exchange server is able to detect whether some certain player fails to connect during the game. On every one minute, the stock exchange will detect each player by sending the request. If it finds such players, then it will send the information to other players and remove the information of this player stored in the stock exchange.

6 The simulation of this game and how to implement

At first, start the StockExchange.java. Then start Bank.java. After that, start Company.java as well as Player.java.

The following is the part simulation of the game:
Account's balance is: 1159.28