COMP2007 (part II) *Group* Coursework

Attempt in groups of 3 or 6 (see required extension for groups of 6) and submit by the Friday 12th February. It is worth 5% towards the coursework total.

Goal To write a distributed Stock Broker system that serves a number of users wishing to play the stock market, and where the server connects automatically to the Internet to update its current stock prices.

Groups: You should form into groups of 3 people or 6 people and register your group using the form available under the Group Coursework section of Moodle. One person should try to take the 'team leader' role, registering the group with Moodle, assigning tasks and organizing regular group meetings – to ensure the project progresses.

Project Organization: To allow project teams to work collaboratively, you should probably employ a source control system such as Subversion. There are a number of 'free subversion hosting' sites which allow you to manage collaborative software projects. These typically allow collaborative (team) source version control using many modern IDEs (Eclipse, Netbeans, IDEA, etc.) They also generally include bug and issue tracking, Wikis and Blogs for collaborative discussion. The site http://www.xp-dev.com/ would be a reasonable choice since it allows private projects with any number of users per project (some free hosting sites restrict number of users). Although xp-dev is quite a basic one compared to many newer hosting sites (Google 'free subversion hosting' to find others … although many only allow a maximum of two registered users per project for free).

Submission: Submission will be online via Moodle. You should submit a zip archive named GroupN.zip (where N is your group number). This should contain your source code, a README file **and also a description of how concurrency aspects are handled in your system (max 3 pages) together with a UML static class diagram giving the overall architecture of the system which can be used during the description of concurrency aspects.** The UML class diagram should provide a useful structural **overview** of the complete system, outlining key classes and methods (hence is should ***not*** give all the methods for all the classes). The zip archive should ***not*** contain any class files or library .jar files. Please supply an Ant build file that can be used to compile the system and a README file giving basic instructions for compiling the system and running it. It is essential that your zip format preserve your directory structure. Essentially I should be able to unzip your zip file, read the README file, compile the system with Ant, then run it.

Marking: Initially peer reviewed and commented upon. Then graded A-F based on both the software submission (C is satisfactory - basically works and written OK, A-B for progressively better, D-E for worse) and your peer review of another system. Note that concurrency aspects of the system such as correct synchronization will be scrutinized.

The Requirements

This project is similar to the UCL_BANK_DEMO explained in the last lecture – but there are key requirements that are different. You task is to correct, modify and extend the UCL_BANK_DEMO example to create a server for a Stock Broker, together with a command-line interface (**extension: GUI interface**) that can be used by clients of the Stock Broker. The interface should allow clients to:

1. Create new Stock Broker accounts for themselves.
2. Terminate a Stock Broker account.
3. Get a listing of the current prices of different registered stock. (**extension: have a GUI panel of constantly updating stock prices.**)
4. Buy and sell stock – whereupon money is used/deposited in the money section of their account and the stock holding of the account are adjusted accordingly.
5. List everyone's account details in terms of:
    a. Actual money in their account.
    b. Stock units they hold in different companies.
    c. The total value of the account – the sum of the money they hold and the total value of the current stock units they hold. (Essentially a High-Score table used to compete with other stock market 'players')

    (**extension: have a GUI panel of constantly updating information about everyone's account details)**

The following sections give more explicit requirements for these different use cases.

Create a new Stock Broker Account

A new account with the Stock Broker should be created for the client. The client should supply a name for the account and a four-digit PIN number that must be supplied to operate the account. If the account is successfully created, the server should return a unique account number for working with the account.

The account itself must hold information on the current money held by the client and also the number of units of shares held in different companies. At creation, the account contains no money and no shares.

Note that the client does not have any mechanism to either deposit or withdraw money from this account! Essentially the Stock Broker initially loans the client money to buy shares and the money part of the account becomes negative while the 'shares owned' part of the account becomes positive. Hopefully the shares increase in value and when they sell them, the money part of the account becomes positive (i.e. they pay off the loan and make profit). The 'Total Account Value' informs the client whether they have currently made money or lost money by summing both the money component of the account and the current value of all the shares held. Only when the account is terminated does the client get informed whether they owe money to the

Stock Broker or whether they get money from the Stock Broker, as outlined in the next use case.

Terminate an Account

The client needs to supply the account number and PIN number to terminate the account. At this point – all shares held within the account are automatically sold. The server then informs the client what the total monetary value of the account is and the interface informs the client how much money they owe the Stock Broker, or whether the Stock Broker owes them money.

Get a listing of current prices of different registered stock

Any client can get a current listing of registered stock. This listing should include the stock market symbol for the company, the company name and the current price in pence for the company stock. It is essential to include the stock market symbol within this listing since this is used when specifying which stock to buy and sell.

The Stock Broker server should have between 5 and 10 companies registered for the clients to buy and sell shares in. You can select these companies yourself by looking at the URL:
http://uk.finance.yahoo.com/
Note that you should choose companies on the London Stock Exchange so that currency (given in pence) is the same. Use the 'Symbol Lookup' to look up the company symbol for some well-known companies. For instance, 'Glaxo' returns "GlaxoSmithKline Plc" with the company stock symbol "GSK.L".

Once you have chosen between 5 and 10 companies and noted down their company stock symbols, you can use the following URL to obtain stock prices for them:
http://quote.yahoo.com/d/quotes.csv?s=GSK.L+VOD.L+BT.L&f=nl1
The structure of this URL is explained in the Resources section and it is also explained at this URL address:
http://www.gummy-stuff.org/Yahoo-data.htm

This particular example will return a CSV (Comma Separated Values) formatted list containing the company names and current trading prices (in pence) like:

```
"GLAXOSMITHKLINE O",1303.00
"VODAFONE GRP. ORD",141.55
"BT GROUP ORD 5P",142.10
```

Within a browser, this may be opened in spreadsheets like Excel. But within the Stock Broker server you will want to use a URL Java class to read this information using the InputStream, parsing this CSV formatted text into company names and prices.

The Stock Broker server should update the stock prices periodically (say every 5 minutes) so that it's current values are kept up-to-date.

Buy and sell stock

These are the key operations. When buying stock the client needs to specify:

- Account and PIN number.
- What company the client wishes to buy stock in (given as a company symbol).
- How many units of stock the client wishes to buy.

In doing so, the money in the account should be decreased by the number of units bought multiplied by the current stock price (per unit stock in pence) of the company. If successful, that number of units of that particular company stock should be added to the account. The Stock Broker will only loan each account up to 10,000 pounds – if the stock bought would bring the account to below -10,000 pounds then the operation should fail with an exception indicating that there is insufficient funds available.

When selling stock the client needs to specify the account/PIN number, the company they wish to sell (given as a company symbol) and also the number of units of stock they wish to sell. If successful (i.e. if the account currently contains that number of units of that particular company) then the money in the account will be increased by the number of units sold multiplied by the current stock price of that company, while the stock owned in that company within this account will be decreased accordingly.

Note that real Stock Brokers take commissions and charges for buying and selling stock (in addition to taxes) – for simplicity we will not apply any charges to these transactions within this system. Similarly real monetary bank accounts that are negative tend to be charged interest over time. Again, for simplicity, our Stock Broker system will charge 0% interest on accounts in dept.

List everyone's account details

This is similar to listing out account details within the UCL_BANK_DEMO application, except in this case the accounts contain both money and also units of stock owned in different companies. The 'Total value' of the account should also be given in terms of money plus the current value of all the stock owned. Clearly this operation should not list the PIN numbers of the accounts.

The Client

Whereas the client for the UCL_BANK_DEMO takes only a single command at a time, the client for this Stock Broking System should be more interactive (although still command line based for groups of 3, and GUI-based for groups of 6). The client should be started by providing the hostname (or IP address) and port number for the rmiregistry of the Stock Broker server. After starting, the client should take interactive commands, responding by requesting from the user any parameters that may be required by the command, issuing the command to the server via RMI, and finally providing the user with the server's response. An 'exit' command should allow the user to terminate the client.

Implementation Notes

The client/server interaction should be carried out using the RMI framework. The interaction between the server and the Yahoo quote service to obtain stock price information is socket based – although probably best handled using the java.net.URL object since the Yahoo stock quote server is a web server.

You may hard-code selected shares that the Stock Broker server deals with as company stock symbols and descriptions. Choose only 5 to 10 companies so that this is manageable. A class needs to maintain price information on this selection of stocks, regularly updating the prices using the URL given above. Using a separate Thread to do this regular updating seems sensible.

Clearly concurrency between this updating thread and other client-generated threads that query the company stock information needs to be handled correctly.

*Note that multiple clients may manipulate **the same account** at the same time and that concurrency issues (problems which are currently present in the UCL_BANK_DEMO example) need to be dealt with correctly.*

Getting Started

1. Download the UCL_BANK_DEMO from the Moodle site.
2. Everyone should read through the code and get a feel for how it works.
3. Ensure that you can compile the code and also run it.
4. Sort the serialization bug (on the list command) and the synchronization bugs. One of the synchronization bugs results in identical account numbers being created for accounts created by different clients (as demonstrated in the lecture). Another synchronization bug can result in money being lost during deposit or withdraw when multiple clients operate on the same account simultaneously. (It should be noted that multiple clients are allowed to operate on the same Stock Broker account simultaneously – for instance if the account was owned by a company).
5. Ensure these problems are resolved before modifying and extending the system.
6. At this stage you may wish to refactor the code to use class and variable names that are appropriate for this particular problem.
7. You need to organize your project group. Different people within your group may then wish to proceed with different tasks. Alternatively, you may organize yourselves to be responsible for modify different parts of the system such as the client, server and stock information classes. Or you may wish to take on different project roles. However you organize yourselves, the following tasks need to be carried out:
    a. Designing a UML class diagram which describes the overall layout of the system – this will be useful so that everyone can agree on a system design.
    b. Extending the client to be more ‘interactive’ as outlined above. (**extension: or implementing a GUI client for larger groups)**
    c. Adding code that handles the PIN number into both the client and server.

d. Investigating the URL class and adding a class or classes that handle information on company stock and carry out the functionality involved in automatically updating the prices of this stock.

e. Adding in the additional functionality concerned with listing the stock descriptions and prices.

f. Adding in functionality concerned with buying and selling the stock.

g. Adding in functionality concerned with listing out account details for everyone registered with the server.

h. Adding in the functionality concerned with terminating an account.

8. Everyone should be involved in reviewing and documenting the complete source code and locating if any synchronization problems may be present.

Resources

- The COMP2007 Moodle site will contain a form to register your group members, a discussion group to query the requirements, and information about source control systems that you should use to ensure concurrent programming activity by different members of the group stays synchronized!

- http://finance.yahoo.co.uk/ Visit this site to obtain company stock symbols that you may wish to add to your Stock Broker server (London Stock Exchange).

- http://quote.yahoo.com/d/quotes.csv?s=GSK.L+VOD.L+BT.L&f=nl1 Use this URL format to obtain quote information in CSV (comma separated value) format. The '?' symbol is followed by URL parameters given as X=Y which are separated by '&' symbols (X is the parameter name, Y is the value). In this case the first parameter, 's', gives the stock symbols which you request information on, separated by '+' symbols. In this case this gives the current price for GlaxoSmithKline, Vodaphone and Electronic Arts on the London Stock Exchange. The second parameter, 'f', gives the format requested (in this case the letter n is followed by a lowercase L and then the number 1 – this encodes the **N**ame of the company followed by the **L**ast trade price). So this would returns back three lines for each company in turn providing the company name followed by its last trade price as a float values given in pence.

- http://www.gummy-stuff.org/Yahoo-data.htm This provides further details about the Yahoo financial download URL (including all the different information formats provided).