CE303 Assignment

The assignment is to build a socket-based client-server system to trade a single stock in a simple stock market. Each client application is a trader. Once a trader obtains the stock, it prompts the user whom to give it to. The clients can connect and disconnect from the server at any time.

You will need to implement two versions of each of the server and the client applications: in C# and in Java. All these applications have to work with each other seamlessly.

The rules of the market

At any point in time, exactly one trader has the single stock. This trader needs to decide whom to give the stock to. (They are allowed to give the stock to themselves.) Once the decision is made, they give the stock to the corresponding trader. For simplicity, we do not take into account buying / selling the stock; we simply give it to another trader.

New traders can join the market at any time. The number of traders in the market is unlimited. Every trader joining the market has to be assigned a unique ID that will not change until they leave the market and will not be reused after they leave the market. All the traders including the one with the stock will immediately learn about new traders and, hence, the current stock owner can decide to give the stock to the trader who has just joined the market.

Any trader can leave the market at any time (the client application can be closed/killed). If the trader with the stock leaves the market, the server gives the stock to one of the remaining traders.

If there are no traders in the market, the server waits until someone joins the market in which case the first trader to connect receives the stock.

Client User Interface

Each client has to display their own ID, information about all the traders currently in the market and the ID of the trader currently having the stock.

Server User Interface

The server has to display main events:

- Trader joining the market (display the ID of the new trader and the list of traders currently in the market).
- Trader leaving the market (display the ID of that trader and the list of traders currently in the market).
- The stock being automatically given to one of the traders after the trader with the stock left the market (display the ID of the new trader with the stock).
- Trader giving the stock to another trader (display the IDs of both traders).

Implementation

Please note that the core requirement of the assignment is to implement the client-server architecture using sockets. The server and each trader have to run in separate processes (not thread!). **Submissions with a single process running all the traders will lose a lot of points.** Also, the emphasis should be on reliability. For example, if two traders leave at the same time, or a client process is killed (not closed "correctly"), this should be handled correctly.

There has to be server-side validation of data. For example, if trader *A* gives the stock to trader *B* but trader *B* leaves just before that, the server should prevent this, e.g., by returning the stock to *A*.

By default, only command line interfaces are expected. However, you can score additional points for graphical user interfaces, see below.

Additional Tasks

To score extra marks, you can implement the following functionality. These extra points are added to the standard grading of the corresponding implementation attribute (not the overall mark). For details see the Grading section.

1. (Up to 15 extra points) If the server is closed or killed, the clients will (i) start a new server application, (ii) reconnect to the new server to continue the market. For the users, this server change has to be transparent; it should not affect the state of the market.
2. (Up to 5 extra points) Implement graphical user interfaces for the client and/or server. Pay attention to usability rather than design.
3. (Up to 5 extra points) Unit tests are not required in this assignment but you can score extra points if you have them and describe them in the report.

Submission

The submission should include the code, a README file explaining how to run the Java and C# code, and a report (PDF or Microsoft Word document) of **no more than 1000 words** with the following sections.

Implemented functionality

At the beginning of your report, fill in the table provided in the "Implemented functionality table.docx" document on Moodle.

If your implementation of a function is incomplete, describe what you have done or what is missing in a few words. Otherwise state "yes" or "no".

Protocol

This section should include a detailed description of the application-level protocol. Explain it from the client's point of view: what data does the client send to the server and what data does it expect from the server, and in which order is the data exchanged. Be as specific

as possible. **This section, together with the assignment brief, should be sufficient for an independent developer to implement a client and a server compatible with your software.**

Client Threads

This section should tell what threads are running in the client process. Briefly explain the purpose of each thread and when it is created and terminated.

Server Threads

This section should tell what threads are running in the server process. Briefly explain the purpose of each thread and when it is created and terminated.

Project review

A review of your project – how did it go? Which parts were easy, which parts were challenging? Are there any features you are particularly proud of? Are there any problems with the quality of the program? How was your project management? Is there anything you would do differently next time?

Grading

Attribute	**Weight**
Implementation in the first language	60%
Implementation in the second language	15%
Report	25%

Which of C# and Java is the first language is decided in your favour; if the mark for the C# implementation is higher than the mark for the Java implementation, C# will be assumed your first language. Otherwise Java will be assumed your first language.

Grading Table for the Implementation Attributes (marked separately for C# and Java implementations)

80-100: The implementation exceeds expectations.

70-79: The applications implement all the required functionality, function smoothly and are reliable.

60-69: The applications implement all the required functionality, however there might be some minor issues with their reliability, or there could be insignificant issues with the required functionality.

50-59: The application lacks some of the required functionality but an attempt to guarantee reliability is evident from the code.

40-49: The application implements the basic functionality; reliability is not properly considered.

0-39: The application does not function in a reasonable way, or it does not meet most of the requirements.

Also, extra points will be added to the implementation mark for the additional functionality described in the Additional Tasks section. As a result, the mark for each implementation attribute can exceed 100. The overall assignment mark will be capped at 100.

Grading Table for the Report Attribute

80-100: The report exceeds expectations.

70-79: The report gives sufficient details in every section.

60-69: The report includes all the required discussions but may lack some details.

50-59: The report may lack some significant details.

40-49: The report includes few details about the protocol and/or threads.

0-39: The report is lacking useful information in most of the sections.

Academic Integrity

You are reminded that this work is for credit towards the composite mark in CE303, and that the work you submit must therefore be your own. Any material you make use of, whether it is from textbooks, the web or any other source (other than the given program code or CE303 lecture / lab worksheets) must be acknowledged in a comment in the program, with the extent of the reference clearly indicated.