Implemented functionality

Function	C#	Java
Client establishes a connection with the server	yes	yes
Client is assigned a unique ID when joining the market	yes	yes
Client displays up-to-date information about the market state	yes	yes
Client allows passing the stock to another player	yes	yes
Server manages multiple client connections	yes	yes
Server accepts new connections while traders are exchanging stock among themselves	yes	yes
Server correctly handles clients leaving the market	yes	yes
Client is compatible with the server in the other language	yes	yes
Additional tasks:		
Client GUI	no	no
Server GUI	no	no
Server restarts are correctly implemented	no	no
Unit tests	no	no

https://moodle.essex.ac.uk/mod/page/view.php?id=459287

Protocol

Firstly, the server is started. After the server is started, it waits for client connections. When the client program is run, the client will be automatically connected to the server with the same port. Unique ID is incremented with each connection. When the server picks up the client, it creates a new thread and a ClientHandler class with the port. In the ClientHandler class, there is a hashmap for the traders in the market. Scanner and the Writer objects are created. The ClientHandler calls the bank class to create a trader with the uniqueID and a stock value. After that, the writer object sends a "success" message to the socket. ClientHandler enters a while loop which is where the handling of the clients is done by responding to their requests. In this while loop, balance, transfer and id can be requested, which this class will respond accordingly. Exceptions are checked for unusual inputs.

When the ClientProgram is running, a Client object is created. Client ID is provided by the server and set in the client object. The user is then asked to enter a Trader ID to transfer the stock to or “-1” to print the list of currently active traders.

When the user enters a valid Trader ID to send the stock to, the transfer method (Transfer())

is called in the client class. This class uses the writer object to write to the server which the server responds with a result. Before the transfer has happened, the bank checks if the user sending the stock actually has the stock. After the transfer, the stock is removed from the sender and received by the receiver. The updated traders with the stocks can be seen with the “-1” command entered by the user. If the result is valid, the transfer happens. If not, for example not a number, “Unknown command.” is printed to the user. If a TraderID that is not in the market is input, “Trader not in market.” Will be printed to the user.

The user can type “-1” to see the list of Traders with who has the stock. When the user types “-1”, client.getInfo() method is called and that method uses the writer object to request information from the server. The server responds with the list of traders using the information from the bank with the method getCurrentTraders().

A client can also type “exit” to leave the market. The Quit method is called in the client class. This class uses the writer object to write to the server so the user can leave the market. When a client disconnects, its socket is closed and trader is removed from the bank traders list. When a connected user types “-1” again, they would see that that trader has disconnected.

Both C# and Java work with eachother meaning that, C# Server with Java Client is working as well as Java Server with C# Client.

Client Threads

A thread is created for each client in the Server class. A client thread is terminated when that specific running program is closed. Each client thread is syncronized (locked) in the bank transfer so that everything goes smoothly in the market.

Server Threads

A single server thread is created when the Server class is run. The server thread is terminated when the server closes. Each client are syncronized (locked) in the bank transfer so that all the users are at the same position in the market.

Project review

For me, the server-side seemed easier to program than the client-side. I had less problem and bugs on the server-side. Establishing a connection with the server and managing multiple connections were the easiest. Passing the stock to another player was a bit harder but still manageable. Up to date information about the market and assigning unique ID was the hardest part in my opinion. I am most proud of my abilities of be able to program this kind of an application. My project management went well. I planned before I started and it worked out well in the end. Next time, I would draw diagrams instead of just researching.