Distributed Systems - Assignment

Date: 22nd November 2018

Student Name: Lauren Keenan Brennan

Student Number: C15434102

Subject: Distributed Systems

Assignment Title: Assignment

*I declare that this work, which is submitted as part of my coursework, is entirely my own, except where clearly and explicitly stated.*

**How to start the Server and Client**

1. Open 2 separate command lines (you can open more if you want to have more than 1 client running)
2. cd into the assignment folder on both cmd’s
3. On one cmd, type Server.bat and press return
4. On the other cmd, type Client.bat and press return
5. That’s it!

**OR**

1. Open 2 separate command lines (you can open more if you want to have more than 1 client running)
2. cd into the assignment folder on both cmd’s
3. On one cmd, type java -classpath . Server and press return
4. On the other cmd, type java -classpath . Client and press return
5. That’s it!

**PLEASE NOTE –** items are removed from the items.txt file when they are sold, so you will have to write more items onto the file yourself after iterations.

# Architecture & Implementation

This system uses a star pattern architecture – multiple clients connected to a single server. In my Server.java file, I have a server class, a client handler class and a timer class. The server class instantiates the server and waits for clients to connect to it. It also instantiates the Timer and Client classes. The timer class sets up the auction timer and deals with all the timer events. The client handler deals with all the client events like receiving and sending messages to the client.

## Client.java

Client class contains variables for the host address, port number and socket. In the Main() function, it sets host address to localhost and then creates a new socket with the given host address and socket number. It then instantiates the Incoming class and Outgoing class and starts them both.

The Outgoing class handles all messages sent by the client to the server. In the Outgoing() function, the server socket is set to the socket which was created earlier. It also sets up PrintWriter which can write on an output stream to the Server and sets up a stream for keyboard entry. When the Outgoing class is first run, it sends the user a message telling them to enter 0 to close the connection if they want to. In this class, there are 2 variables – “tosend” and “sending”. “tosend” is a String variable but I need to send an Integer variable to the Server, so “tosend” is turned into an integer and stored in “sending” to do this. The “sending” variable is then sent to the server

The Incoming class handles all messages sent from the server to the client. In the Incoming() function, the server socket is set to the socket which was created earlier, and a Scanner is set up which will read from an InputStream stream from the Server. When the Incoming class is running, it will read the messages sent from the server and print out the messages out with line break

If the user sends 0 to the server, it will close the connection to the server.

## Server.java

The Server class contains variables for the server socket, port number and client socket, variables for file reading and writing, an ArrayList to hold client Socket addresses and a client count, a variable to hold the current time value used for the auction timer and a string to hold the value of the current auction item. In the main() function, all the variables are initialised and the first item up for auction is read from the items file. It then sets up the server socket on port 1234 and adds 0 to the bidding file using the addToFile() function which will create the file and set the current highest bid to $0 to begin with. It then instantiates the timer class and starts it. Then the Server waits for clients, accepts them and adds them to the Client ArrayList. A thread is then created to handle communication with this client and pass the constructor for this thread a reference to the relevant socket and starts the client handler. When a new client is added to the ArrayList, the client count is incremented.

The clientMessage() function used to send a message to all connected clients. It takes in the client socket address, an integer value and a message. It sets the client socket variable to the socket address which was passed in when the function was called and the sets up a PrintWriter to write and send messages to clients. It sets up the PrintWriter on the clients Socket and then sends them the message which was passed in when the function was called.

The addToFile() function is used to add new highest bids to the bidding file. It is a simple process which was taken from the Writer.java file we used in Week 3.

The readFromFile() function is used to read the highest bid from the bidding file. It reads the file one line at a time and sets the highest bid to the value on the last line of the file.

The ReadItems() function is used to read auction items from the items file. It sets the current auction item to the value on the last line of the file. Reading from the last line of the file means that if an item is not auctioned off, the item will remain on the last line of the file, so it will automatically be auctioned off again.

The RemoveItems() function is used to remove an auction item from a file when it has been auctioned off. It has a temporary string used to hold item values as the function reads lines from the items file until the end of file is reached. If the value of the current line is NOT the same as the current item up for action, it will be added to the temporary string with a line break. The items file is then closed and a new PrintWriter is instantiated to write to the items file. The function then copies the temporary string into the items file without the item that had been sold.

The Timer class which will handle the auction timer. It first sets the time to the current time in milliseconds and then it runs. While the timer hasn't reached 60 seconds, it will continuously update the current time value to be able to compare the current time with the timers start time. It will also send a message to the clients letting the know when 30 seconds have passed in the auction. When the timer finishes it will begin the auction for that item again if it wasn’t bid on. If the item was sold, it will remove the item from the items file and reset the current item to the new item. It will then let clients know that the item was sold and what the new item up for auction is. It then resets the server time value and resets the highest bid to 0.

The ClientHandler class handles all the clients. It first sets up a reference to associated socket and then lets the client know of the current item up for auction and ask how much they would like to bid. It then takes in the client’s bid value and convert it from a String into an Integer. The current highest bid value is gotten, and the current highest bid and the clients bid are compared. If the client’s bid wasn't higher than the current bid, it informs them of the current highest bid. If the client made the new highest bid, it will add the new highest bid to bidding file and inform all clients of the new highest bid. It will also reset the timer value. The ClientHandlers continuously run while the client hasn't sent 0 to close the connection. If the client sends 0, it sends the client the exit code, so they know to close the connection and removes the client from the array list and decreases the client count. Finally, it closes the connection.