Making Improvements (2.3) to ChatSystem

Serialized Objects

I went about this a little back-to-front as certain items are more easily done if we have a serializable set of data. Note for brevity I have removed the automatic client to save duplicate of updates.

So; first thing created the ChatMessage class which contains two strings, a key and a value and implementes serializable.

Then in the client I replaced the buffered reader and writer with ObjectInputStream and ObjectOutputStream (oins, oouts).

This was then replaced throughout the code so now on send a new ChatMessage was created (with the key "message" and the value of the message) and sent, likewise reception was of a ChatMessage and used the value as the text.

I then made the same changes in the server (inside the ClientConnection class), updating everything to move around those classes rather than strings.

Note the first update was serializing objects only in the ClientConnection i.e. the wider server just dealt in strings (though this would change).

Also note a little "gotcha" – the order of the DataInput/OutputStreams needs to be different in the client and server or it'll hang (the input stream waiting for an output stream on the other end).

Once I'd done this messages were passed serialised so we could move on.

Client ID

The client ID can be set and managed by the server (and is useful for the echo).

Added an integer id to ClientConnection in server, a nextid field in Server and when a client connected set it to nextid and incremented it.

Even if a client leaves then everyone gets a unique ID (to max int).

I had ClientConnection take an int in the constructor to set and added a getter (it shouldn't change once created so no setter).

Added output in the server to show client number to test.

Echo

Now we have an ID in the server we can just make it so the client updates it's GUI when a message is written (using Write).

On the server side we update the Send() to also take the ID of the originating client. When we loop through clients, if the ID matches… don't send.

Control Codes

Now the class ChatMessage is serialized we can use the key to send other types of data such as commands around.

Here we will implement the quit key which will shut down the server or kill a client if received.

This requires making the overall Server class aware of ChatMessage rather than just receiving strings but easily enough done.

Firstly the client is updated so that it will generate a quit message when the command /quit is entered (we could expand this so / issues a command for example but as there is only one command we look for /quit specifically) – this then generates a ChatMessage() with the key "quit".

Also when the client receives a message with the key quit, it exits.

Then updated the Server to pass around ChatMessage as discussed (rather than strings). In send if it's a quit we sent to all (messages don't go to origin) and in both client and server once a quit is processed System.exit() is called.