Chris Carter

Jared Heidt

**Clype 1.0 Questions**

**Explain why there should be a separate class to receive data from the server and print it, and the client only gets data from the user and sends it to the server. Also, why is the class called a ‘listener’?**

Sending and receiving data must happen concurrently, therefore objects that send data and objects that receive data must run in separate threads. ClientSideServerListener is needed because data must be able to stream to the client side at any moment, without impeding ClypeClient’s ability to freely send data.

ClientSideServerListener is called a “listener” because it constantly “listens” for incoming data. In other words, the class is always waiting for and echoing incoming data.

**Explain why you need a separate thread for each client, and why you cannot handle all clients in the main server thread. Conceptually, why is the listener class ‘ClientSideServerListener’ different from the class ‘ServerSideClientIO’?**

A separate thread is needed for each client so that every client can send messages at the same time, and not have to wait for other clients to send messages first. Since each client is in a separate thread, all client data can be received and broadcast without the need for “waiting in line”.

ClientSideServerListener is merely a listening class; its job is to listen for data from the server (more abstractly, from any client), and echo that data to the user. ServerSideClientIO is a listening and broadcasting class, since its job is to listen for data from one client and broadcast it to all other clients managed by the server.

**Explain why the broadcast() and remove() methods are synchronized. You may find it easier to answer this question after completing all programming.**

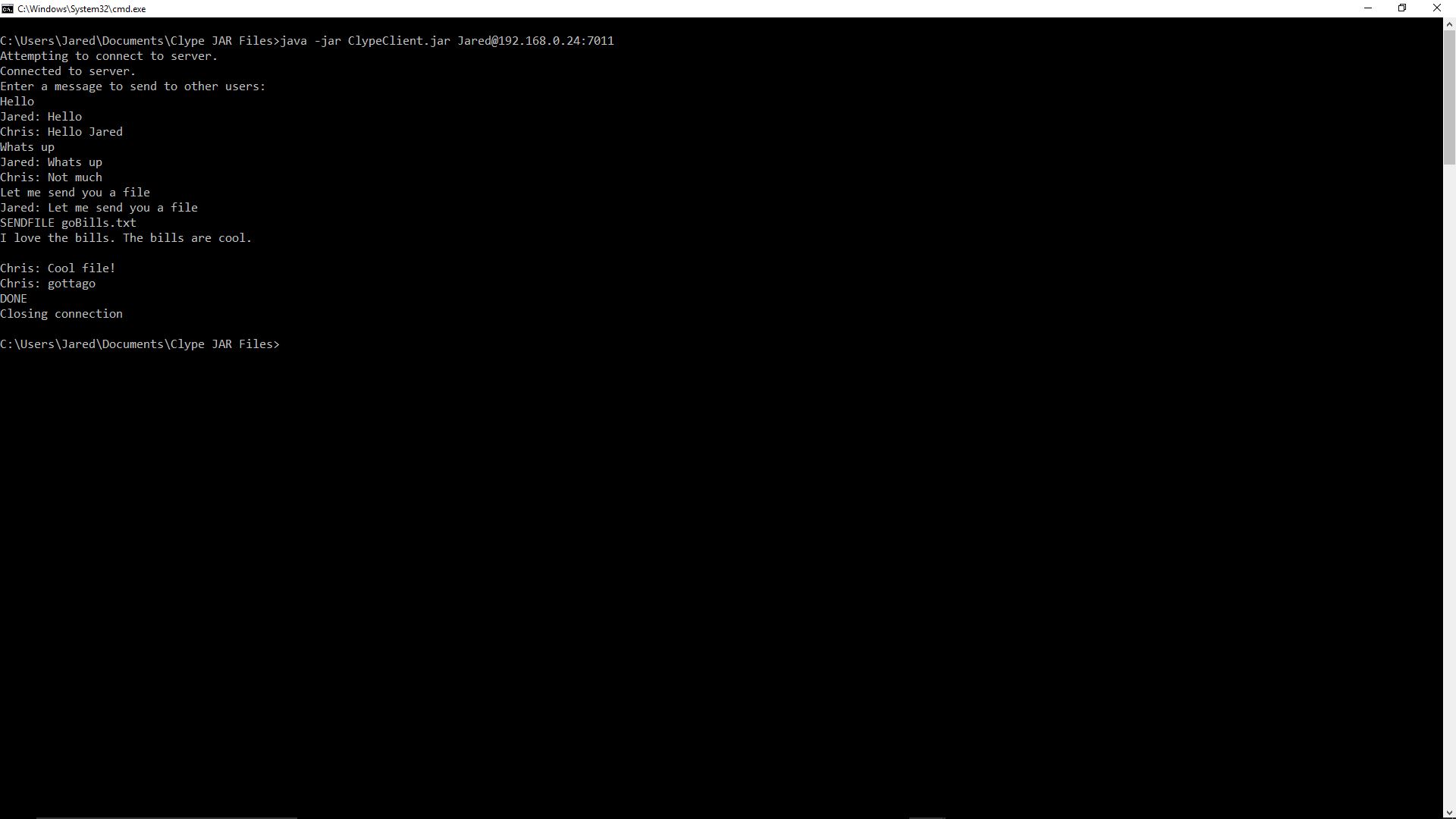
These methods are synchronized so that a user can not be removed while broadcasting and cannot broadcast while being removed. The server should not be able to broadcast to a removed client.

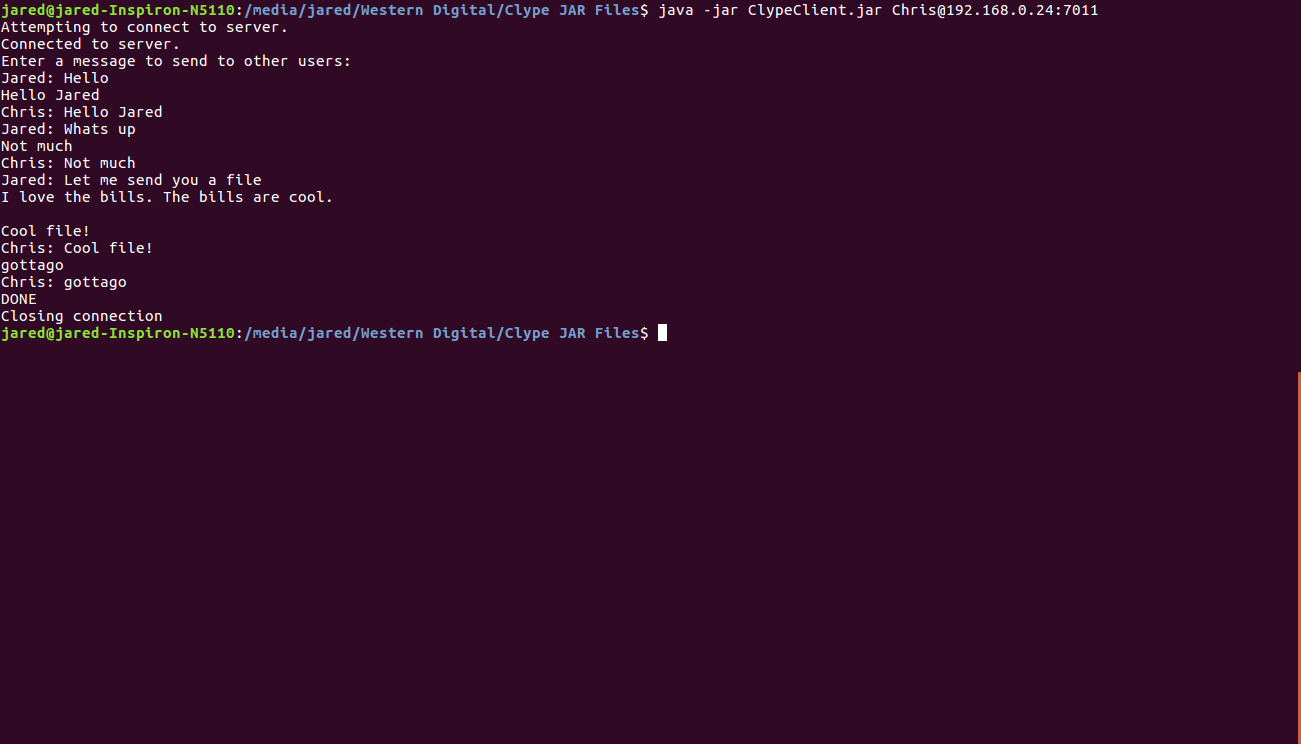
**Discuss all new methods and new code in existing methods that you wrote to handle LISTUSERS.**For LISTUSERS, the client must first send a message to the server with message type LISTUSERS. A ServerSideClientIO object then saves the data**.** The data object is then broadcasted. We created a broadcastToClients method in ServerSideClientIO which checks if the ClypeData type is LIST\_USERS. Depending on whether the type is LIST\_USERS or not, we use one of the two overloaded “broadcast” methods in ClypeServer.

We overloaded the function, “broadcast” in the ClypeServer object to take a ClypeData object *and* a ServerSideClientIO object. This overloaded “broadcast” function sends the ClypeData object *only* to the ServerSideClientIO argument. In this (overloaded broadcast) method, the ClypeData argument is checked if its type is LIST\_USERS. If it's type is LISTUSERS, then the Server retrieves a string containing all of the usernames. The retrieval is done with the ClypeServer method getListUsers(). The ClypeServer then creates a MessageClypeData object with it's message containing all of the usernames and sends this data only to the ServerSideClientIO argument.

**Testing Results**

Client Side - Jared

****

Client Side - Chris

Server Side

