**Client Design:**

**Definition:**

The client is a human user in the chat system. Each client may communicate with any number of other clients in the chat system by sending streams of text messages to the server based on a defined protocol (see “client\_server\_protocol”).

**Client Class:**

**Fields:**

String myUsername

Socket mySocket

PrintWriter myOut

HashMap<String, User> userNameToUser

When a client logs into the chat system, we add the client’s username to this set. This datatype prevents duplicate entries of the same username (which should not occur assuming the server design is correct). This set contains the usernames of all clients logged into the chat system except “me.”

HashMap<String, Conversation> conversationNameToConversation

When a new conversation is made, we add the conversation to the map. The conversation’s name is used as the key. A new window (JFrame) is opened for each new conversation that the client creates or enters. Closing the conversation window is tantamount to exiting the conversation.

HashSet<String> myInvites

When a client receives an invite to a conversation from another client, we add the conversation name to the set. This datatype allows us to group all invites to the same conversation together into one invitation. When the client responds to an invitation to a conversation, we remove the conversation name from this set.

**Methods:**

login()

Open a connection with the server and keep the program running (call handleConnection)

handleConnection(Socket socket)

Keeps the program running, listening for messages from the server.

Adds messages from the server onto a queue?

handleRequest(ServerMessage serverMessage)

Handle individual requests from the server.