Design Document

Jason Yang, Jenny Lin, Sayeed Tasnim

Server

Class: Server
- Attributes:
  - UserListHandler allUsers: UserListHandler object containing all users connected to the server
  - ChatHandler allChats: ChatHandler object containing all chatrooms in the server
- Methods:
  - serve(): Runs the server while listening to connections and creates ClientHandlers for those connections

Class UserListHandler
- Attributes:
  - ArrayList<ClientHandler> users: All clients in this user list
- Methods:
  - removeUser(ClientHandler user): Removes user from users and informs all remaining ClientHandlers in users of the change by adding a message to their outputQueue
  - addUser(ClientHandler user): Adds user to users and informs all ClientHandlers in users of the change by adding a message to their outputQueue

Class ChatHandler
- Attributes:
  - ArrayList<Conversation> chats: All chatrooms in this user list
- Methods:
  - removeRoom(Conversation chatroom): Removes user from users and informs all remaining ClientHandlers in users of the change by adding a message to their outputQueue
  - addRoom(Conversation chatroom): Adds user to users and informs all ClientHandlers in users of the change by adding a message to their outputQueue

Class: ClientHandler
- Attributes:
  - ChatHandler myChats: ChatHandler containing all chatrooms client is a member of
  - Queue outputQueue: Queue containing commands to be sent to the client from an appropriate Conversation
- Methods:
  - handleConnection(Socket socket): Makes bufferedReader around the socket. Parses input from the bufferedReader and sends commands

        through the socket back to the client
        - parseCommand(String command): parses a command to figure out what action to perform
        - joinChat(Conversation Chatroom): calls addRoom(Chatroom) in myChats, and calls Chatroom.users.addUser(this)
        - leaveChat(Conversation Chatroom): calls removeRoom(Chatroom) in myChats, and calls Chatroom.users.removeUser(this)
        - makeChat(String chatName): creates a new Conversation object with the given chatName and adds it to both the server's ChatHandler and this ClientHandler's ChatHandler; this ClientHandler is the first ClientHandler in the Conversation's UserListHandler
        - sayMessage(Conversation Chatroom, String Message): consumes a String from the inputQueue and sends the Message to the Chatroom's queue
        - hearCommand(String Command): consumes a Command from the outputQueue and sends it back to the Client

Class: Conversation (mutable & threaded)

- Attributes:
    - final String Name: the name of the chatroom, set by client
    - UserListHandler users: the UserListHandler of all the clients in that particular chatroom
    - Queue Messages: Queue containing messages that were said in the chatroom
- Methods:
    - relayMessage(String message): consume a message from Messages and send that message to the outputQueue of each user in the users

<u>ClientHandler</u>

This object will handle in general the IO to and from the respective client. Upon construction, the client will require a username from the client and will be checked against the list of users in the userlist object. if the username is unique, the clienthandler will add the instance to the userlist object. if the username is not unique, the socket will be disconnected.

The client handler will have an blocking 'output' queue to handle sending messages back to the client. It will also have a list of all of the chatroom objects that the client is currently associated with.

This class will also be threaded such that each instance will be its own thing. in this thread will be a loop that will check if the output queue is empty and if not, relay the queue back to the client. The client will also check if the buffered reader is empty and if not, start parsing the strings in the reader. These strings will be of the form [header] [message]. The header will contain information of what the clienthandler should do (make a message to room x, disconnect, etc).

If the command is to connect to a chatroom, the client handler will use the Conversation object's UserListHandler to add this instance to the UserListHandler object. It will also append the Conversation object to this instance's ChatHandler, representing the rooms it is connected to.

If the command is to send a message, the clienthandler will parse and append the message to the chatroom object's output queue.

If the command is to disconnect the user, the handler will iterate through all of the rooms that the client is connected to and disconnect the client before disconnecting the client from the server.

<u>UserListHandler</u>
This object will hold a list of all of the connected users on the server. It will have provisions to add a user or remove a user from the list where this object will inform all of the users on this list of the change by appending a command to each users output queue.

<u>ChatHandler</u>
Much like the UserListHandler object, this object will hold a list of all of the chatrooms currently created. if a room is created or destroyed, this class will access the main UserListHandler object and inform all members of the change.

Client

Class Client

- Attributes:
  - String userName
  - GUI listOfUsers
  - GUI listOfChatrooms
  - ArrayList<GUI> chats
- Methods:
  - addNewConversation(String conversationName)
  - updateListOfUsers()
  - updateListOfUsersInConversation(String conversationName)
  - addMessage(String chatroom, String user, String message)
  - checkName(String name)

Clients (people) will connect first to a server at a specified IP address and port, and be required to provide a username - Once a username is given, the gui checks to make sure that the username has no spaces. Once that check passes, a connection is made to the server, and it's checked to see if the username is taken. This will be done on the server-side with a synchronized method that goes through the list of all current users. If the username is taken, a message is returned to the client telling them to try something else, and then they are disconnected from the server. The client then keeps submitting possible usernames until they choose one that is not taken.

Client-Server Protocol

From client to server:

- connect [username]
  - Sent when a client attempts to connect with a username
- disconnect [username]
  - Sent when a client with the specified username attempts to disconnect

- make [chatroom name]
  - Sent when a client attempts to make the specified chatroom
- join [chatroom name]
  - Sent when a client attempts to join the specified chatroom
- exit [chatroom name]
  - Sent when a client attempts to exist the specified chatroom

- message [chatroom name] [message]
  - Sent when a user attempts to send the particular message to the specified chatroom.

From server to client:

- connectedServer
  - Sent when the server successfully connects the user
- invalidUsername
  - Sent when the user submits a username that is already taken
- disconnectedServer
  - Sent when the server is about to disconnect the user
- connectedRoom [chatroom name]
  - Sent when the server successfully connects the user to the specified chatroom
- disconnectedRoom [chatroom name]
  - Sent when the server successfully disconnects the user to the specified chatroom
- serverUserList [username list]
  - Sent when the server updates the list of users on the entire server
- chatUserList [username list]
  - Sent when the chatroom updates the list of users in the chat
- message [chatroom name] [message]

  Sent when the chatroom sends a message to the users

Before a client attempts to connect, the client will need to provide a username first. This will establish the connection and send the string “connect [username]”. After the client side checks to make sure that the username contains valid characters only (no spaces), the server will check the available list of usernames and if the username has not been taken yet, the server will construct a ClientHandler for the particular client and add the client to the UserListHandler. The server will send a “connectedServer” command to let the client know that the client is connected and then display the UserListHandler and the ChatHandler to the client as a list of current users and chats by sending a “serverUserList” command. Otherwise, if the username is taken, the server will send a ‘serverUserList’ message with a list of users in the chat displayed in a separate window and an ‘invalid’ message to the user. This will state to the user that the username is already taken and that the user should select a different chat name.

Whenever the list of all the server users is updated, the server sends a chatUserList command to each of the clients.

When a user requests to make or join a chatroom, the client will send the respective command “make” or “join” to the server. The GUI will create a new window for the specific chat and the user will be able to send messages through a text box. The server will send a “connectedRoom” command to let the client know that the client is connected to the chatroom.

When a user sends messages, the client sends a “message” command to the server. When a server receives the “message” command, it will process it and place the message into the proper conversation queue. The conversation queue will consume the messages and the server will send “message” commands to each of the clients in the particular chat room.

When the list of users in the particular chat room changes, the server will send a “chatUserList” command to each of the clients in the particular chat room. This will update each client and GUI with a new list of users in the chatroom.

When a user disconnects from a chatroom, the client will send an “exit” command to the server. The server will process the disconnection and relay a “disconnectedRoom” command back to the client if the client is still connected to the server.

When the user disconnects from the server, a “disconnect” command is sent to the server. The server will then remove the user from all chatrooms and from the list of all users on the server and return a “disconnectedServer” command. All open GUI chat windows will be closed except for the original username prompt window. The user will need to login again in order to reconnect to the chatroom.

Conversation

The Conversation will be a class that serves as one ‘room’ of people chatting. It will contain a list of all the connected clients in that room (subscriber) and upon receiving a message from the server with text, update all of the connected clients of the change (publisher). People are free to come and go from this object (subscribe/unsubscribe). As long as one person remains in the room, the Conversation will still be valid and upon becoming invalid, the

Conversation will deregister itself from the server and all connected clients will be notified of this change.

Conversation classes will contain an instance of UserList that is a subset of the UserList that the server has. The Conversation will need at least one client in the list of ClientHandlers for the Conversation to hold. Otherwise, the server will delete the Conversation by calling the respective command in the ChatHandler. The Conversation will have a blocking queue of the messages to be passed to the members of this conversation. This class will be threaded such that the thread checks if anything is in the queue and if so, alert all parties. This class will aslo have provisions to add or remove a client from this object, updating the respective lists and informing all members of this change.