Advanced Topics in Computer Science

UNIT 1: MULTI-THREADED CHAT SERVER PROJECT SPECIFICATION

Mr. Fieldman
jfieldman@hw.com

Description

The goal of this project is to create a multi-threaded chat server in Java. Your project should combine concepts learned in the Threading lab with concepts from the Socket-programming example discussed in class. Your server instance should be able to field multiple simultaneous TCP/IP connections from various users, and allow those users to communicate with each other.

There is no pre-coded framework for this project. You are responsible for creating the server from scratch, though you may use any resources you deem appropriate, including source code we have discussed in class.

This is a two-week project. The first week will focus on creating the basic chat room with minimal features. The second week will focus on expanding the chat room with features of your choice.

Participation

You will be in 2-3 student groups. It is expected that everyone in the group will participate in the coding. This can be measured, in part, through your source control history. There may also be write-ups and/or presentations associated with your project. Make sure you understand and can explain how your project works.

Source Control

You are expected to use your GitHub accounts for this project. One user in the group should create a public repository for the project, and then make other members of the team, as well as the teacher, collaborators of the repository. This will create one central repository that all team members will work from. When the project is over, other team members can fork the repository into their own accounts.

All project submissions will be done through GitHub. Make sure all work is committed and pushed to GitHub in a timely manner.

Chat Client

There is a Java chat client available on the Hub. The command-line usage is:

```
java ChatClient <host> <port>
```

Specify **<host>** as **localhost** in order to connect to your own computer.

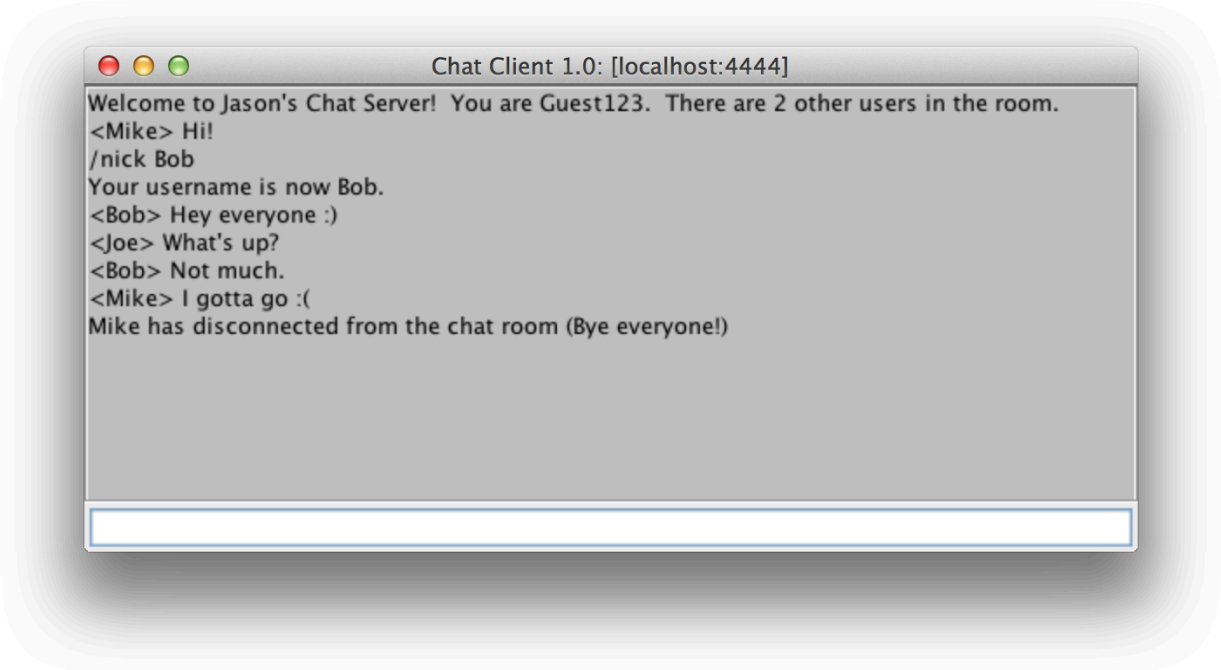
You may modify this chat client as you see fit, but are encouraged to push your modified chat client to your git repository.

Requirements For Week 1

Your server:

- Should not crash.
- Should be able to send input from one user to all other connected users.
- Should be able to assign each connected user a modifiable username.
- Should prepend the appropriate username to outgoing text
- Should understand that any input line beginning with a "/" character is a command, not intended for transmission to other users. Required commands are:
    - `/nick <username>` : used to change usernames:
        - Sending "`/nick Jason`" to the server changes the client name to Jason, if that name is available on the server.
        - Informs all users of the username change.
    - `/disconnect <message>` : gracefully disconnects the user and informs the chatroom. Includes <message> in the disconnect message.
- When a client connects to the chatroom, it is given a guest username and announced to the chatroom.
- When a client disconnects (from /disconnect or connection failure), their departure is announced to the chatroom.

Here is example output from a connected client:



Optional Features for Week 2

You may implement any of the following, or add your own ideas!

- Use the `/me` command for emotes.
  - Example, typing "`/me laughs`" would output "`* Jason laughs`" (assuming the username was Jason).
- Enforce punctuation. Capitalize first letters of sentences; add periods to the end of a line (if not present).
- Use the `/whisper` command to send private messages.
  - Example: "`/whisper Jason Hi, what's up?` " sends the message only to the user Jason, or displays an error if no user by that name exists. The recipient knows who the whisperer is.
- Commands to send messages to the chat room (or whisper) anonymously.
- Create a more complex join handshake so that the client must submit the desired username before being added the chat room.
- Add the ability to create and join "rooms". Users can only see other users in the same room.
- Modify the chat client and server to support colored text.
- Administrative accounts, with powerful admin commands like `/kick` and `/ban` that disconnects users or bans users from reconnecting.