Homework 4: Chatty Chat Chat

(a.k.a. ChattyMcChatFace)

COSC150

Assigned: 10/18/2018
Due: 11/01/2018 at 11:59pm ET

This assignment is worth 180 points.

Description

In an attempt to become the next dotcom quadrillionaire[1], you are inventing an awesome Java application that allows multiple users to do group messaging. (We're pretty sure this hasn't been invented yet.[2])

Your program, which you must call ChattyChatChat[3], will consists of two parts: the ChattyChatChat server and the ChattyChatChat client.

There is one instance of the ChattyChatChat server. It will run on a particular port and receive TCP ("Stream") connections from one or more ChattyChatChat clients. If a message arrives from a ChattyChatChat client, the ChattyChatChat server should relay that message to all of the other ChattyChatChat clients. (E.g., if Alice, Bob, and Charlie are using the service and Alice sends a message, the server should relay that message to Bob and Charlie.)

Users participate in the group discussion by running the ChattyChatChat client. The client connects to a ChattyChatChat server. When the user types a message, the ChattyChatChat client sends that message to the ChattyChatChat server, which in turn will relay it to the other clients. When a message is received by the ChattyChatChat client (from the server), the client prints out the message.

---

[1] Please remember your kind COSC150 instructor.
[2] It definitely has. That was sarcasm.
[3] Sorry.

The Gritty Details

Command-line Usage and Basic Operation

The ChattyChatChat server, **which must be called ChattyChatChatServer.java**, takes a single command-line argument: the port to listen on. For example, you would invoke it from the command-line as follows

```
java ChattyChatChatServer 9876
```

if you want the server to listen for client connections on TCP port 9876.

Important note about port numbers: In previous classes (e.g., COSC 051) you may have used the server at cs-class.uis.georgetown.edu to run/test code. If you would like to do so for testing this project, please use the following rule to decide on a server port:

- Take the numerical suffix of your netID (mine is re268 => 268)
- If you are in the Monday/Wednesday section, add 20000 to this number
- If you are in the Tuesday/Thursday section, add 10000 to this number

These rules ensure that no two students are using the same port on cs-class (which would lead to bad things happening for all involved).

Instructions for connecting to cs-class.uis.georgetown.edu are below; note that this does not apply if you are working on your local machine.

The ChattyChatChat client, **which must be called** ChattyChatChatClient.java, takes in two arguments: the hostname where the server is running and the port on which the server is listening. For example, you would invoke it from the command-line as follows

```
java ChattyChatChatClient cs-class.uis.georgetown.edu 9876
```

if you want the client to connect to the ChattyChatChatServer instance running on cs-class.uis.georgetown.edu that's listening on port 9876.

If you want to run your server on your local machine, you can instead use the hostname 'localhost' which resolves to the IP 127.0.0.1 and references your personal machine.

Note that ChattyChatChatServer must be started before any ChattyChatChatClient can join the group chat.

Chat Commands

Normally, when a client types a message, that message is sent to the other clients and printed to standard output (e.g., via System.out.println()).

The ChattyChatChat service supports additional *chat commands* that cause some other action to take place. The commands, which are typed by users into the ChattyChatChat client, are as follows:

- `/nick name`
  Sets my nickname to be `name`. E.g., "`/nick Cosc150Guru`". The nick command may be used more than once, which updates the client's nickname. Nicknames must be single words and cannot contain spaces.

- `/dm name msg`
  Send the message `msg` directly to the user with nickname `name`; if no connected client has that nickname, then ignore the message. No client who is not named "`name`" should receive the message.

  If multiple clients have the same nickname, then they should receive the message.

  Example:
  `/dm Cosc150Guru Hi there, how's it going?`

- `/quit`
  Exit the client. Quit the program.

Other Requirements

- You should not use Java RMI for this assignment. You should use Java sockets.
- You may use threads. In fact, you probably should
- You should not assume an "order" of who types what when. As soon as a user presses [ENTER], that message should immediately appear at all other clients. That is, if Alice, Bob, and Charlie are using ChattyChatChat, when Alice types something and presses [ENTER] -- regardless of who sent the last message -- then that message should appear at Bob's and Charlie's client. Threads will be helpful here.
- Here's a non-requirement: you can invent your own protocol, and it's not necessary that your ChattyChatChatClient work with other students' ChattyChatChatServers, or vice versa. Clearly, your ChattyChatChatClient must work with your own ChattyChatChatServer, and vice versa.

Advice

You might find the following to be helpful:

- Java String's indexOf method
- Java String's split method and this discussion about splitting on space
- Oracle's Java client/server example

Testing

You can develop your code on your local machine and use Eclipse or any other IDE.

If you like, you may also run your code on cs-class.uis.georgetown.edu. You should be able to log into cs-class.uis.georgetown.edu using SSH and your NetID and NetID password. I have requested access for all enrolled students on cs-class; there may be a slight delay while the accounts are created. If you have trouble accessing cs-class.uis.georgetown.edu on/after October 25, please post a private note on Piazza.

Note that you do not need to use cs-class.uis.georgetown.edu.

As with previous homeworks, this homework will be at least partially autograded.

OK, I've just finished my amazing ChattyChatChat program. How do I turn it in?

What to turn in:

You will turn in a single zip file that contains ChattyChatChatClient.java, ChattyChatChatServer.java, and any other supporting files. Your .zip file should not have any directories; that is, everything should be in the "root" of the .zip file. Hence, you should **not** use the package statement.

If you have trouble creating .zip files, please post a note on Piazza.

Once you have created a .zip file that contains your code (again, in the top (and only) directory of the zip file), upload it to Autolab at https://autolab.georgetown.edu/. In Autolab, look for the "ChattyChatChat" assignment.

Where to go for help:

For questions about this assignment, please post to Piazza at https://piazza.com/georgetown/fall2018/cosc150/home.

Grading Rubric

This rubric may be updated closer to the submission deadline.

Description	Points awarded
Something submitted	10
It compiles!	30
Server binds and accepts at least a single connection	15
Server binds and accepts multiple client connections	25
Client can connect to server	20
Client’s typed message appears at at least one other client	25
Client’s typed message appears at all other clients	20
/quit command works	5
/dm command works if no user has nickname	5
/dm command works if 1 other user has nickname	10
/dm command works if multiple other users have nickname	15

I reserve the right to spot-check your submitted source files for good programming practices, naming conventions, comments, etc.

This check will not affect your numerical grade on this assignment, but may lead to me cornering you after lecture to ask overly-obtuse questions about, e.g., the meaning of a strangely-named variable.