Networks
Project 3

Socket Programming
A Chat Client & Server

1 Chatting

This project involves both *client* and *server* programs that connect via transport layer sockets. Specifically, this pair of programs provides a simple chat service, in which connected clients can send messages that are either broadcast to all users or sent directly to a specific user.

Begin by grabbing the source code:

`https://bit.ly/cosc-283-23f-p3`

The two most important files are the `ChatServer.java` and `ChatClient.java` source code. The best way to see how these programs interact is to use them. So, compile all the code, and then run the server:

```
$ javac *.java
$ java ChatServer
Chat server started on port: 8080
```

At this point, the server is running on your computer, listening at port 8080, waiting for a client to connect. The cursor will just sit there, waiting for something to happen. At this point, you should **open another shell** so that you can separately run the client:

```
$ java ChatClient
Connected to the chat server.
enter your username:
fred
(server):fred has joined the chat!
(server):currently: 1 users in the room
```

Now open a third shell, and run another chat client:

```
$ java ChatClient
Connected to the chat server.
enter your username:
scott
(server):scott has joined the chat!
(server):currently: 2 users in the room
```

You can, of course, create additional shells and connect with additional clients and names. Once you do, you can send messages to every client connected, or you can target a particular client by prefixing the message with their name:

```
[From the "scott" client...]
Here we are!
@fred Just for you.

[...you will see in the "fred" client...]
scott: Here we are!
scott (private): Just for you.

[...and the server will show...]
received: Here we are!
received: [TargetUser=fred] Just for you.
```

2 What to do?

Play around with this code. Open the source, look at it, and see how the server *listens* on a port, waiting for the client to *connect* on that same port. Read the `README.md` file, and notice that you could try adding a feature to this chat code and see if you can make it work.

Is there any networking happening? Notice the use of the hostname `localhost`, which is a special name that always resolves to the IPv4 address `127.0.0.1`. This address creates packets that are then "routed" by the host to itself; that is, the network layer creates the packet, and then it accepts the packet as though it had arrived over the network. This seemingly strange self-addressing is good for debugging the network stack (the layers) and any network applications by keeping the activity local to the programmer's system.

You are, however, welcome to modify the code so that you can run the server on one machine and then connect from another. For example, you all have login access, with your college credentials, to connect to `remus.amherst.edu` and `romulus.amherst.edu`, two Linux servers to which you could connect, upload the code (lookup use of the `scp` or `sftp` commands), and run the server. You should then, from your own computer, be able to modify the client to connect to a port on that server. (**Important warning!** Only one of you can use port 8080 at a time on a given server. You may want to choose another port number, numbered above 1024, for this experiment. Be a little random about it, reducing the odds of choosing the same port number as someone or something else.)

What to submit? Don't! There is nothing to submit. This project is for your own curiosity and exploration; there's nothing you need to submit, and nothing that will be graded.