Design Document - Brian Scott - G00411391

Command Line Chat Application:

ChatServer:

Creates a server socket which will wait for a client to connect.
Listens on port '4321' as default.
Once a client connects, the server will create a socket connection and then:

- Creates a thread to 'listen' for input from the client connection
    - When input is detected, it is output to the server's console interface
- Creates a thread to allow input from the server user
    - When a new line of input is submitted using the enter key, the message is sent through the socket to the client
        - The client also uses a similar listener to handle this message and display it to the client user

The chat session may be terminated when either user (Server or Client) enters the message '\q'.

- At this point, the connection will be terminated and the program will end.

ChatClient:

The chat client runs very similarly to the chat server, with two distinctions:

1) A server must be running for the client to connect to at runtime.
    - If a server is not running and listening on port 4321, the client will return an error and terminate.
2) The client may accept a command line argument which specifies the host to connect to.
    - If no argument is provided, the default 'localhost' is used.

The client also uses two threads for managing:

- Listening for new data from the server
- Accepting data from the user and sending it to the server.

Similarly, the client can terminate the connection by entering the message '\q'.

Application implementation:

The application has defined the classes 'ChatServer' and 'ChatClient', making it easily reusable.
The main method of the application simply invokes the appropriate class for its purposes.

Running the application

1. Run the ChatServer first, no arguments
2. Run the ChatClient - If 'localhost' is appropriate for your configuration, then no arguments are needed. Otherwise, provide the host as an argument to the program on the command line.

External sources used:
Reviewed the usability of a Multicast socket, but opted against it:
https://docs.oracle.com/javase/7/docs/api/java/net/MulticastSocket.html

Reviewed previously build group chat submissions for ideas on how best to implement:
https://www.geeksforgeeks.org/a-group-chat-application-in-java/

ATU Lessons:
https://web.microsoftstream.com/video/fcf51e22-fa7f-4cb9-838a-4e008120d23d?referrer=https:%2F%2Fvlegalwaymayo.atu.ie%2F

I had some trouble building the packages on command line form Eclipse, so I saught help here:
https://www.freecodecamp.org/news/how-to-execute-and-run-java-code/
https://stackoverflow.com/questions/50882074/how-to-fix-java-lang-noclassdeffounderror-in-command-prompt
https://www.examtray.com/java/how-compile-and-run-java-programs-cmd-or-eclipse
https://www.guru99.com/java-packages.html
https://www.eclipse.org/forums/index.php/t/803689/

Research into the setSoTimeout functionality:
https://stackoverflow.com/questions/12820874/what-is-the-functionality-of-setsotimeout-and-how-it-works