Network Programming Assignment.
Student: G00411344

The application consists of a server called ChatServer and a client (ChatClient) for communication between multiple users. The application utilises a Thread pool to manage multiple client connections concurrently and supports basic message broadcasting to all connected clients.
It allows users to join, send messages, and gracefully exit the chat by entering “\q”.
There is an error message and stack trace if unable to connect to the server.

The ChatServer listens on a specified port (5566) for incoming connections and maintains a list of connected clients.
It uses a fixed-size thread pool (ExecutorService) for handling concurrent client connections.
The Client Handling accepts incoming client connections using a ServerSocket.
It creates a new ClientHandler thread for each connected client and adds the new client handler to the list of connected clients.
It then submits the client handler thread to the thread pool for concurrent execution.
The ClientHandler Thread is an inner class representing a thread for handling an individual client. It manages input and output streams for communication with the client.
Communication Handling continuously reads messages from clients and broadcasts them to others.
Closes client sockets and removes handlers when clients leave, broadcasting a departure message.

The ChatClient defines the server's IP address ("localhost") and port (5566). It then creates a socket to connect to the chat server.
It sets up input streams to read from the console and the server and an output stream to send messages to the server.
The Message Handling Thread starts a separate thread to listen for incoming messages from the server. This continuously reads and displays messages from the server in the main thread.
It accepts user input from the console and sends messages to the server.
Breaks the loop if the user enters the termination command ("\q").

How To Run The Application:

Open a terminal or command prompt and navigate to the folder named ChatAppAssignment.
Compile both ChatClient.java and ChatServer.java.
In the same terminal, run the ChatServer class.
You should see a message indicating that the server has started.
Ensure that the server is running before starting any clients.
You can now open new terminals for each client you want to run.
In each terminal, run the ChatClient class.
Follow the instructions on the client's terminal. You can run multiple instances of the client in different terminals to simulate multiple users.
To quit the application, type \q.

References:

Socket Programming Examples, John French, Online Class Tutorial, Week 8; Network Technologies.
Threads, John Healy, Weeks 11 & 12; Software Design & Data Structures.
The Knock Knock Server, Oracle, Java Tutorials. (Online) Available at:
https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html.
Class Thread, Oracle, Java Documentation. (Online) Available at:
https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html.
Multi-threaded chat Application, GeeksForGeeks. (Online) Available at:
https://www.geeksforgeeks.org/multi-threaded-chat-application-set-1/.