COMP3331 ASSIGNMENT

INTRODUCTION

In this assignment, I completed all tasks to the best of my ability within the time frame given. I was unable to successfully implement the `/p2pvideo` functionality, however, I have left my attempt in the `Client.java` file and commented it out to demonstrate that there was a great effort put in towards this task.

Compilation Instructions (JDK 11):

```
`javac Server.java, Client.java, ClientThread.java, User.java, Group.java`
```

Server.java should be in the same directory as ClientThread.java and User.java and Group.java

DESIGN

The files included in submission are Server.java, Client.java, User.java, Group.java, and ClientThread.java.

The main Server is contained within the Server.java file. To handle interactions between the server and client, I created the ClientThread.java class which extends Thread. This ClientThread class works on the server side to interpret information sent by the client and act appropriately.

The Server file is setup with many static attributes and methods so that they can be accessed and used by the ClientThread file. The main function first checks that the user has provided the correct args and once such is verified, the ClientThread.java run() method is invoked to handle interactions.

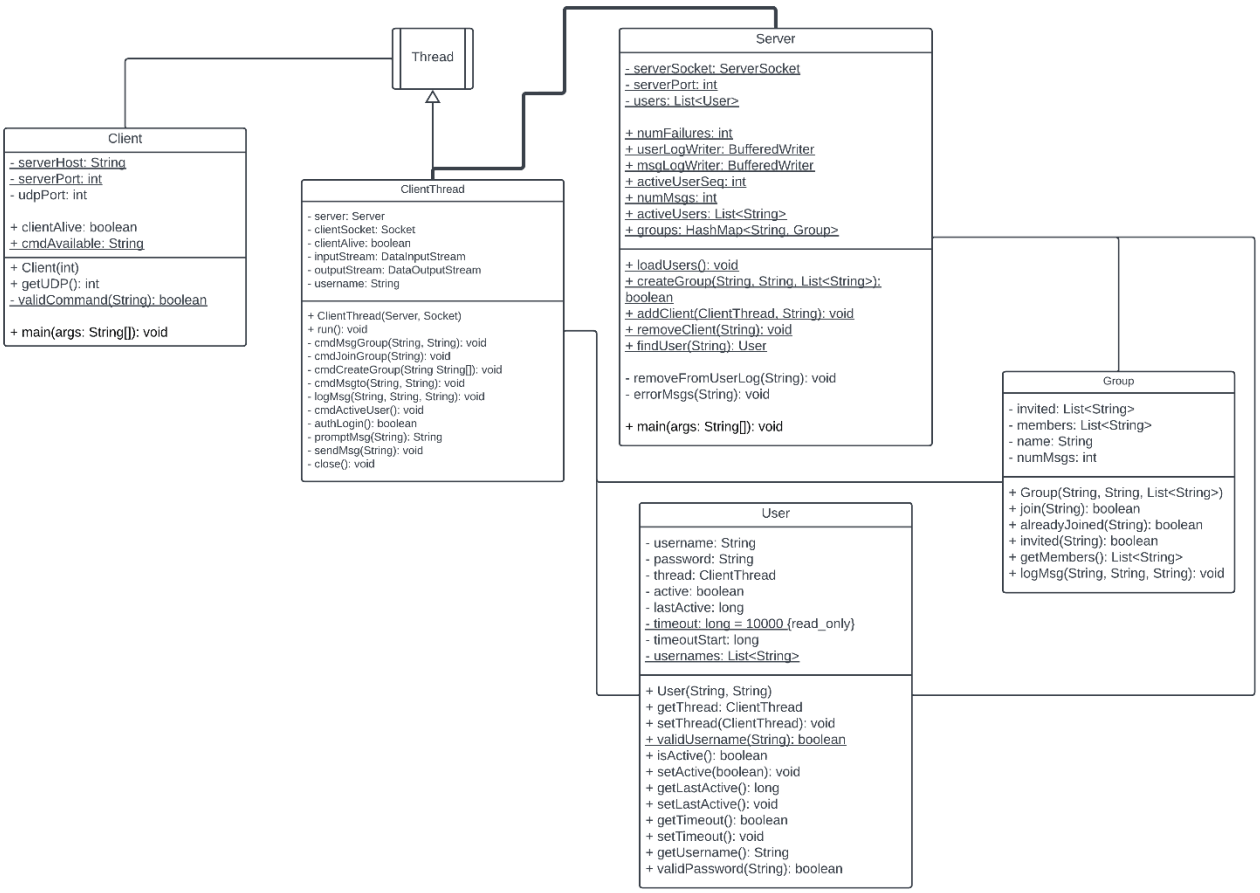
The run() method in ClientThread.java handles every interaction between the server and client. It first ensures that the client is authenticated properly and logs in. Tasks such as these are delegated to helper functions within the ClientThread.java class. . Originally this class existed within the Server.java file, however, I extracted it to its own file when refactoring for readability when navigating the codebase.

These interactions are only possible with how I set up the Client.java file. This file handles all of the client-side interactions and then conveys information from the server to the client and from the client to the server. The first aspect of the main function is error checking that the user provided the corrected args but then I create a loop that is only broken once the user has logged in (if the user has been blocked the program sends a message, then closes). This allows commands to only be accepted from the user once they have successfully logged in. Then, I set up 2 threads to

simultaneously listen for user input and server input. This allows messages from other users to appear in the client's terminal whilst still allowing that client to use commands.

To help with the server-side actions, I also created the User.java and Group.java classes. These are helper classes that I created for carrying out functions related to user and group commands.

Below is a UML diagram of the program design. (If difficult to read, I have also included a pdf of it in my submission)



To complete this program in Java, I conducted self-study into additional multi-threading methods, including the usage of synchronized methods. The following resources influenced my codebase and there exists adapted code of such in my implementation.

- Oracle Java Docs on synchronised methods, concurrency, threads
- Geeksforgeeks website
    - Synchronisation in Java
    - Creating asynchronous multi-threaded chat application
- Stackoverflow questions
    - Java P2P using UDP socket
    - Static Method Multi-threading
- YouTube Video: DEVDAYO "JAVA NETWORK : UDP - P2P Chat (Datagram Socket)"
- YouTube Video: Prototype Project "Peer to Peer (P2P) Chat w/ JAVA"

Some behaviours in my code that I was not sure about and wanted to explain, include:

1. Authentication: invalid username. My program does not block the IP of the client and send them into a "timeout" after consecutive failures if just the username is invalid. I understand that the spec says
   "*On entering invalid credentials (username or password), the user is prompted to retry. After several consecutive failed attempts, the user is blocked for a duration of 10 seconds (number is an integer command line argument supplied to the server and the valid value of number should be between 1 and 5) and cannot login during this 10 second duration (even from another IP address).*"
   However upon reading forum post #343, I interpreted the block/timeout as only occurring for consecutive incorrect passwords (thus, allowing me to put logic for this in a separate User class)
2. Messagelog.txt and groupname_messagelog.txt are not cleared upon members logging out or the server closing.
3. Users can log into the same account simultaneously since this behaviour was undefined in the spec

The error messages sent by my program were just formatted as UTF strings and did not follow a specific convention. Within the program, I tried to ensure that my error messages and server messages were all consistent and meaningful.

REFLECTION

This task was quite challenging when starting since it felt like a mountain to climb, however, by breaking it down to each of its commands, I was able to start and complete what I could. I am disappointed that I was not able to fulfill the p2pvideo functionality and that is the first thing I would work on if given more time.

Other future improvements to the program:

- Solving concurrency issues: The assignment specifically did not ask for cases where the same user tries to log on from multiple end points but this could be a future path to tackle.
- Encryption and security. Currently the authentication phase is extremely lax and all credentials are stored as plain text. In future, maybe creating a sign up functionality and hashing the information would be an improvement.
- Further customisation of group chat functions to edit name and invite more users
- User connections: allow sending friend requests to users
- Persistence: users can access their previous chat histories. This could perhaps be done in my current program by modifying the User class to include a chat history functionality
- Scalability: design the server to handle increased loads and influxes of users