Rapport #1

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

TP1
Chat Messaging Application

INF3405 - Reseaux Informatiques

Summer 2023
Department of Computer Engineering
Ecole Polytechnique de Montreal

Last update: 25 mai 2023

Alexandre Nguyen	1631518
Louis-Antoine Martel-Marquis	2222436

1 Introduction

The goal of this task was to build a functional chat messaging application. The primary requirements of the application were as follows :

— Multiple clients must be able connect to a server
— The server must be able to authenticate an user
— The server must be able to store usernames and passwords
— Users must be able to communicate in real time.
— The server must be able to store and broadcast messages written by all users

2 Summary of the Solution

The solution consists of the following classes

— Server
— Client
— ClientHandler
— PassWordAuthentificationProtocol
— ReadThread
— WriteThread

2.1 Server

The role of the Server class is to create a ServerSocket object and bind it to a port and an IP address. The input port and IP address are assumed to be constant and are known by the user in advance. Everytime an user connects to the server, a ClientHandler object is create to manage the new client. The main components of the Server class are :

— An attribute set of Clienthandler objects
— A attribute set of userNumber integers
— A method communicateBetweenClients to broadcast an user's message to all other users connected in the server save themselves.
— A method EliminateClient to remove a client from the server once they are disconnected.

2.2 Client

The purpose Client class is to connect to the server via a socket as well as initiating two simultaneous threads : one to read inputs and one to write outputs. The main featuress of the Client class are :

— A socket linking it to the server
— Starting a Readthread object which runs an input reader until the socket is closed
— Starting a Writethread object which runs an output reader until the user is disconnected.

2.3 Readthread

The purpose of the Readthread class is to run an input reader until the socket is closed. As mentionned previously, this object runs simultaneously with the Writethread object.

2.4 Writethread

The purpose of the Writethread class is to run an output reader until the user writes disconnect in the console. As mentionned previously, this object runs simultaneously with the Readthread object.

2.5 ClientHandler

The clienthandler determines what needs to be done when a client connects to the server. There is only one clienthandler per client. It has its own input reader and output writer to help it manage its client. The main features of the Clienthandler class are :
- Calling a Password Authentification protocol to authenticate the user by the means of their username and password.
- Calling the communicateBetweenClients method define previously
- An input reader to read the messages that were written by the user.
- Methods saveClientInputTXT and printLastMessages to save an user's input in a text file and to provide the last fifteen messages recorded by the server.

2.6 PassWordAuthentificationProtocol

The PassWordAuthentificationProtocol manages the state of the interaction between the server and the client. The user's input on the console serves as the function's input parameter. The user's input is processed by the class and may go through up to 4 consecutive states :
- WAITING the initial state of the protocol
- AUTHENTICATINGUSERNAME where the user is prompted to insert their username
- AUTHENTICATINGPASSWORD where the user is prompted to insert their password. A username/password verification is done at this step.
- AUTHENTICATED. Once the user has passed the username/password check, this state allows the user to broadcast their messages to all other users in the chat application.

3 Difficulties Encountered

The team encoutered issues while trying to implement multithreading or the ability for multiple users to interact with one another in the application. The team had trouble grasping the fact that the program had to allow reading and writing to/from the console simulateneously. Online documentation and examples had to be consulted to fully understand that. In addition, the team attempted to implement the Password Authentification protocol in the Client class. This was unsuccessful for reasons still unclear to the team. The Password Authentification protocol ended up being incorporated in the Clienthandler class as shown in the code. One more bug encountered by the team was how to print the list of 15 past messages. The team confirmed that the Server class can correctly output the 15 lines. However, upon receiving the input from the Server, the Client class failed to print the full list of 15 messages. The team tried many attempts to solve this issue and was unsuccessful.

4 Criticism and Improvements

The following improvements could be brought to this activity :

— The code as-is in the pdf provided by the course cannot be easily copied and pasted into Java. This is an error prone process since formatting must be redone and students may forget to copy and paste some lines.

— Diagrams detailing the interaction between the client, clienthandler and server class work would also have been helpful.

— Links to multi threading in a server (how to connect multiple users) could be provided in the future for students. That information was not readily available in the documentation provided.

— The mandatory rule of 3 students per team could be improved. The class saw a fair amount of students leaving the course. Perhaps the teacher or the teaching assistant may help with the pairings to cover these situations.

5 Conclusion

In conclusion, the team was able to successfully build a personalized chat server application which allows multiple users to chat in a public space. This allowed the team to understand the basics of building a client class, a server class, a clienthandler class as well as an user interaction protocol.