

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



LÝ TUẤN AN – 52000620

BÁO CÁO CUỐI KỲ
LẬP TRÌNH MẠNG CĂN BẢN

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



LÝ TUẤN AN – 52000620

BÁO CÁO CUỐI KỲ
LẬP TRÌNH MẠNG CĂN BẢN

Người hướng dẫn
TS. Bùi Quy Anh

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Em xin chân thành cảm ơn Thầy Bùi Quy Anh đã hỗ trợ cho em trong suốt quá trình học tập. Nhờ có những kiến thức của thầy, em mới có đủ kiến thức để hoàn thành bài báo cáo này.

Em xin chân thành cảm ơn Thầy.

TP. Hồ Chí Minh, ngày 19 tháng 05 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)



Lý Tuấn An

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH

TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS. Bùi Quy Anh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 19 tháng 05 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)



Lý Tuấn An

TÓM TẮT

Báo cáo này trình bày việc phát triển các ứng dụng giao tiếp giữa server và client sử dụng Java Swing để tạo giao diện đồ họa (JFrame) và các giao thức TCP và UDP để truyền dữ liệu. Ứng dụng này minh họa cách thức thiết kế và triển khai một hệ thống client-server cơ bản với hai giao thức mạng phổ biến. Các nội dung chính bao gồm:

- Giới thiệu tổng quan về Java Swing và các giao thức TCP, UDP.
- Thiết kế giao diện người dùng cho server và client sử dụng JFrame.
- Triển khai các chức năng chính như kết nối, gửi và nhận tin nhắn giữa server và client.
- Phân tích và so sánh hai giao thức TCP và UDP trong bối cảnh ứng dụng thực tế.

Báo cáo cũng cung cấp mã nguồn minh họa, các sơ đồ mô tả luồng dữ liệu và chi tiết kỹ thuật về cách xây dựng một hệ thống giao tiếp hiệu quả và thân thiện với người dùng. Thông qua việc thực hiện dự án này, người đọc sẽ hiểu rõ hơn về lập trình giao diện trong Java, cũng như cách sử dụng TCP và UDP để truyền thông tin trong các ứng dụng mạng.

ABSTRACT

This report presents the development of an application for server-client communication using Java Swing for creating graphical interfaces (JFrame) and TCP and UDP protocols for data transmission. The application demonstrates the design and implementation of a basic client-server system utilizing these popular network protocols. The main contents include:

- An overview of Java Swing and the TCP and UDP protocols.
- Designing the user interface for the server and client using JFrame.
- Implementing key functions such as connecting, sending, and receiving messages between the server and client.
- Analyzing and comparing the TCP and UDP protocols in the context of practical applications.

The report also provides illustrative source code, flow diagrams, and technical details on building an effective and user-friendly communication system. Through this project, readers will gain a deeper understanding of GUI programming in Java and how to use TCP and UDP for information exchange in networked applications.

MỤC LỤC

DANH MỤC CÁC CHỮ VIẾT TẮT.....	vii
CHƯƠNG 1. TỔNG QUAN BÁO CÁO	1
1.1 Giới thiệu.....	1
1.2 Mục tiêu	1
1.3 Các chức năng chính của ứng dụng	1
1.4 Phương pháp triển khai	2
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	3
2.1 Java Swing	3
2.2 Giao thức TCP (Transmission Control Protocol)	3
2.3 Giao thức UDP (User Datagram Protocol)	4
2.4 Mô hình Client-Server.....	4
2.5 Mã hóa và Bảo mật Dữ liệu	4
2.5.1 AES (<i>Advanced Encryption Standard</i>).....	4
2.5.2 Hashing	5
CHƯƠNG 3. THIẾT KẾ	7
3.1 Mô hình đề xuất	7
3.1.1 Mô hình hệ thống tổng quan	7
3.1.2 Xây dựng chương trình	17
3.1.3 Test Demo	91
CHƯƠNG 4. KẾT LUẬN.....	118
4.1 Những kết quả đạt được	118
4.2 Những hạn chế và hướng phát triển	118

TÀI LIỆU THAM KHẢO120

DANH MỤC CÁC CHỮ VIẾT TẮT

TCP Transmission Control Protocol

UDP User Datagram Protocol

CHƯƠNG 1. TỔNG QUAN BÁO CÁO

1.1 Giới thiệu

Báo cáo này tập trung vào việc phát triển một ứng dụng giao tiếp giữa server và client sử dụng Java Swing để tạo giao diện đồ họa (JFrame) và hai giao thức mạng phổ biến là TCP và UDP để truyền dữ liệu. Ứng dụng được xây dựng nhằm mục đích cung cấp một hệ thống tương tác giữa người dùng thông qua các chức năng khác nhau, từ chat room đến truyền file và các ứng dụng chuyển đổi đơn vị.

1.2 Mục tiêu

Mục tiêu chính của báo cáo này là trình bày chi tiết về quá trình thiết kế, triển khai và thử nghiệm một ứng dụng đa năng, hỗ trợ nhiều loại hình giao tiếp và chức năng khác nhau giữa server và client. Ứng dụng này không chỉ dừng lại ở việc truyền thông đơn thuần mà còn mở rộng ra nhiều lĩnh vực ứng dụng khác, giúp người dùng có được trải nghiệm toàn diện và phong phú.

1.3 Các chức năng chính của ứng dụng

Ứng dụng bao gồm các menu cho phép lựa chọn các ứng dụng khác nhau sử dụng cả hai phương thức TCP và UDP. Cụ thể, các chức năng chính bao gồm:

- **Ứng dụng Chat Room (TCP và UDP):**

Hai phiên bản chat room sử dụng giao thức TCP và UDP để kết nối người dùng và cho phép họ trao đổi tin nhắn trong thời gian thực.

- **Các ứng dụng TCP:**

- Chat và Truyền File: Cho phép người dùng không chỉ trò chuyện mà còn có thể truyền file giữa các máy.
- Audio Streaming: Truyền âm thanh liên tục từ client đến server hoặc ngược lại, phục vụ các ứng dụng nghe nhạc trực tuyến.

- Mã Hóa File AES: Ứng dụng mã hóa file sử dụng thuật toán AES, đảm bảo an toàn cho dữ liệu.
 - Giải Mã File AES: Cho phép giải mã các file đã được mã hóa bằng ứng dụng mã hóa AES.
 - Text Hashing: Ứng dụng tạo mã băm cho các đoạn văn bản, phục vụ việc xác thực dữ liệu.
- Các ứng dụng UDP:
 - Tic Tac Toe Game: Trò chơi cờ điện Tic Tac Toe với hai người chơi kết nối qua giao thức UDP.
 - Chuyển Đổi Độ Dài: Ứng dụng chuyển đổi các đơn vị đo độ dài khác nhau.
 - Chuyển Đổi Cân Nặng: Cho phép người dùng chuyển đổi giữa các đơn vị đo cân nặng.
 - Chuyển Đổi Tiền Tệ: Hỗ trợ chuyển đổi giữa các loại tiền tệ theo tỷ giá hối đoái.
 - Chuyển Đổi Nhiệt Độ: Ứng dụng chuyển đổi nhiệt độ giữa các đơn vị khác nhau như Celsius, Fahrenheit và Kelvin.

1.4 Phương pháp triển khai

Trong quá trình triển khai, ứng dụng được xây dựng trên nền tảng Java với sử dụng thư viện Swing để tạo giao diện đồ họa. Giao thức TCP được sử dụng cho các ứng dụng yêu cầu độ tin cậy cao như truyền file và chat, trong khi đó UDP được chọn cho các ứng dụng yêu cầu tốc độ và hiệu quả cao như trò chơi và các phép chuyển đổi đơn vị.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Java Swing

Java Swing là một phần của Java Foundation Classes (JFC) – một tập hợp các thư viện cung cấp khả năng tạo giao diện người dùng đồ họa (GUI) trong Java. Swing cho phép tạo ra các ứng dụng với giao diện phong phú, tương tác và có thể chạy trên nhiều nền tảng khác nhau.

- JFrame: Là cửa sổ chính của một ứng dụng Swing. JFrame cung cấp khung chứa các thành phần GUI khác như JButton, JTextField, JLabel, v.v.
- JPanel: Là một container cơ bản để tổ chức các thành phần GUI. JPanel có thể chứa các thành phần khác và hỗ trợ layout management.
- Event Handling: Xử lý sự kiện trong Swing, bao gồm các sự kiện như nhấn nút, nhập dữ liệu, và tương tác với người dùng.

2.2 Giao thức TCP (Transmission Control Protocol)

TCP là một giao thức truyền thông hướng kết nối, đáng tin cậy, và được sử dụng rộng rãi trong mạng Internet. TCP đảm bảo rằng dữ liệu được truyền đi chính xác và theo đúng thứ tự.

- Kết nối hướng kết nối (Connection-oriented): Trước khi truyền dữ liệu, TCP thiết lập một kết nối ổn định giữa client và server. Quá trình này bao gồm ba bước: thiết lập kết nối, truyền dữ liệu và đóng kết nối.
- Độ tin cậy (Reliability): TCP đảm bảo rằng dữ liệu được truyền đi chính xác và theo đúng thứ tự bằng cách sử dụng cơ chế kiểm tra lỗi và xác nhận. Nếu một gói tin bị mất hoặc bị hỏng, TCP sẽ yêu cầu gửi lại gói tin đó.
- Điều khiển luồng (Flow Control): TCP quản lý tốc độ truyền dữ liệu giữa server và client để tránh tình trạng nghẽn mạng. Quá trình này bao gồm việc sử dụng cửa sổ trượt để điều chỉnh lưu lượng dữ liệu.

- Điều khiển tắc nghẽn (Congestion Control): TCP điều chỉnh lưu lượng truyền tải dựa trên tình trạng mạng để tránh tắc nghẽn. Nó sẽ giảm tốc độ truyền dữ liệu nếu cảm nhận được dấu hiệu của tắc nghẽn trên mạng.

2.3 Giao thức UDP (User Datagram Protocol)

UDP là một giao thức truyền thông không hướng kết nối, không đảm bảo độ tin cậy nhưng có tốc độ nhanh hơn TCP. UDP thường được sử dụng trong các ứng dụng yêu cầu truyền dữ liệu nhanh và thời gian thực.

- Không kết nối (Connectionless): UDP không thiết lập kết nối trước khi truyền dữ liệu. Điều này làm giảm độ trễ khi thiết lập kết nối, nhưng cũng có nghĩa là không có cơ chế tự động để đảm bảo gói tin được gửi đến đúng đích.
- Không đảm bảo (Unreliable): UDP không đảm bảo dữ liệu được truyền đi chính xác hoặc theo đúng thứ tự. Nó chỉ làm việc như một ống dẫn, truyền gói tin từ nguồn đến đích mà không có bất kỳ kiểm tra lỗi nào.
- Hiệu suất cao (High Performance): UDP có độ trễ thấp và tốc độ truyền dữ liệu nhanh, làm cho nó thích hợp cho các ứng dụng nhu streaming video, game trực tuyến, nơi mà độ trễ thấp và tốc độ truyền dữ liệu là quan trọng hơn độ tin cậy.

2.4 Mô hình Client-Server

Mô hình client-server là một kiến trúc mạng trong đó client yêu cầu dịch vụ và server cung cấp dịch vụ. Mô hình này thường được sử dụng trong các ứng dụng mạng như web, email, và truyền file.

- Client: Máy tính hoặc thiết bị yêu cầu và sử dụng dịch vụ từ server.
- Server: Máy tính hoặc hệ thống cung cấp dịch vụ và xử lý yêu cầu từ client.
- Giao tiếp Client-Server: Các client và server giao tiếp với nhau qua mạng, sử dụng các giao thức truyền thông như TCP hoặc UDP.

2.5 Mã hóa và Bảo mật Dữ liệu

2.5.1 AES (Advanced Encryption Standard)

Là một tiêu chuẩn mã hóa đối xứng được sử dụng rộng rãi, AES cung cấp một mức độ bảo mật cao bằng cách mã hóa dữ liệu với các khóa có độ dài 128, 192 hoặc 256 bits.

Độ dài khóa càng dài thì mức độ bảo mật càng cao, vì nó tăng số lượng các khóa có thể có mà một kẻ tấn công cần thử để giải mã dữ liệu.

2.5.1.1 Quá Trình Mã Hóa

Quá trình mã hóa bao gồm chia dữ liệu văn bản thô thành các khối có kích thước cố định và áp dụng một loạt các biến đổi toán học.

Mỗi khối được xử lý bằng một khóa và đầu ra của một khối trở thành đầu vào cho khối tiếp theo.

Thuật toán mã hóa lặp lại qua nhiều vòng (10, 12 hoặc 14 vòng tùy thuộc vào độ dài khóa), kết hợp các bit dữ liệu và khóa để tạo ra văn bản mã hóa.

2.5.1.2 Quá Trình Giải Mã

Quá trình giải mã thực chất là quá trình nghịch đảo của quá trình mã hóa.

Dữ liệu mã hóa được chia thành các khối và được xử lý bằng thuật toán giải mã, sử dụng cùng một khóa nhưng theo thứ tự ngược lại.

Mỗi khối được giải mã, và đầu ra được XOR với dữ liệu mã hóa của khối trước đó để khôi phục lại văn bản thô ban đầu.

2.5.2 Hashing

Hashing là một hàm một chiều chuyển đổi dữ liệu đầu vào có bất kỳ kích thước nào thành một chuỗi ký tự có độ dài cố định, thường là một giá trị hash.

Hashing thường được sử dụng để xác minh tính toàn vẹn của dữ liệu, vì bất kỳ thay đổi nào trong dữ liệu đầu vào cũng sẽ tạo ra một giá trị hash hoàn toàn khác biệt

Các Loại Mã Hóa Băm Java Hỗ Trợ:

- MD5 (Message Digest Algorithm 5)

MD5 tạo ra một giá trị băm 128-bit từ dữ liệu đầu vào.

Đây là một thuật toán băm phổ biến nhưng không còn được khuyến nghị cho các ứng dụng mới do vấn đề về bảo mật.

- SHA-1 (Secure Hash Algorithm 1)

SHA-1 tạo ra một giá trị băm 160-bit từ dữ liệu đầu vào.

Tuy nhiên, SHA-1 cũng đã bị coi là không an toàn do các lỗ hổng bảo mật.

- SHA-256 (Secure Hash Algorithm 256-bit)

SHA-256 tạo ra một giá trị băm 256-bit từ dữ liệu đầu vào.

Đây là một trong những thuật toán băm mạnh mẽ và được sử dụng rộng rãi trong các ứng dụng an ninh.

- SHA-384 (Secure Hash Algorithm 384-bit):

SHA-384 tạo ra một giá trị băm 384-bit từ dữ liệu đầu vào.

Nó cung cấp một mức độ bảo mật cao hơn so với SHA-256 do kích thước đầu ra lớn hơn.

- SHA-512 (Secure Hash Algorithm 512-bit):

SHA-512 tạo ra một giá trị băm 512-bit từ dữ liệu đầu vào.

Nó cung cấp một mức độ bảo mật cao nhất trong các biến thể SHA-2.

- SHA-224 (Secure Hash Algorithm 224-bit):

SHA-224 tạo ra một giá trị băm 224-bit từ dữ liệu đầu vào.

Nó là một biến thể của SHA-256 và cung cấp một kích thước đầu ra nhỏ hơn.

- SHA-512/224 và SHA-512/256:

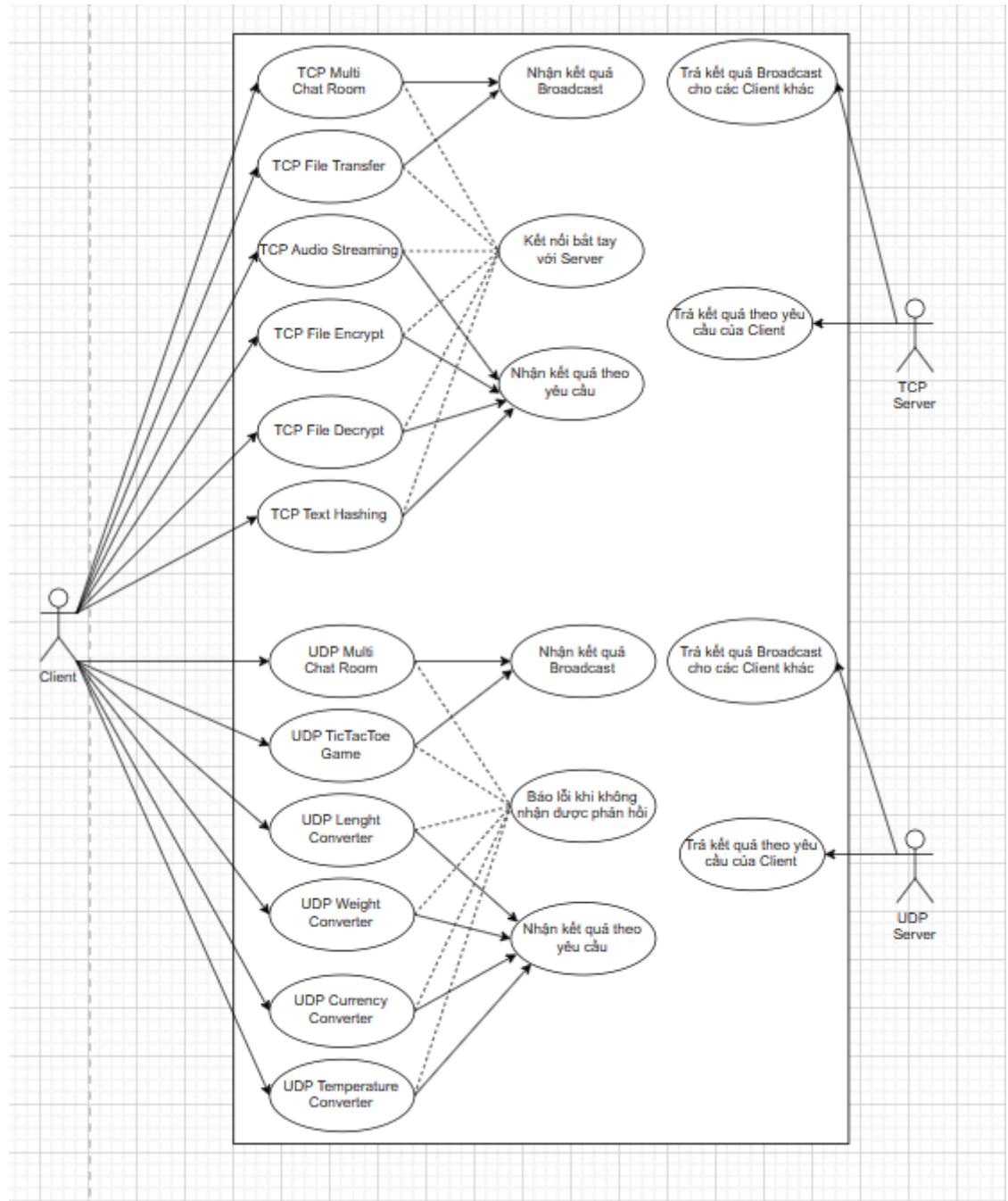
Đây là hai biến thể khác của SHA-512, tạo ra các giá trị băm có kích thước nhỏ hơn lần lượt là 224-bit và 256-bit.

CHƯƠNG 3. THIẾT KẾ

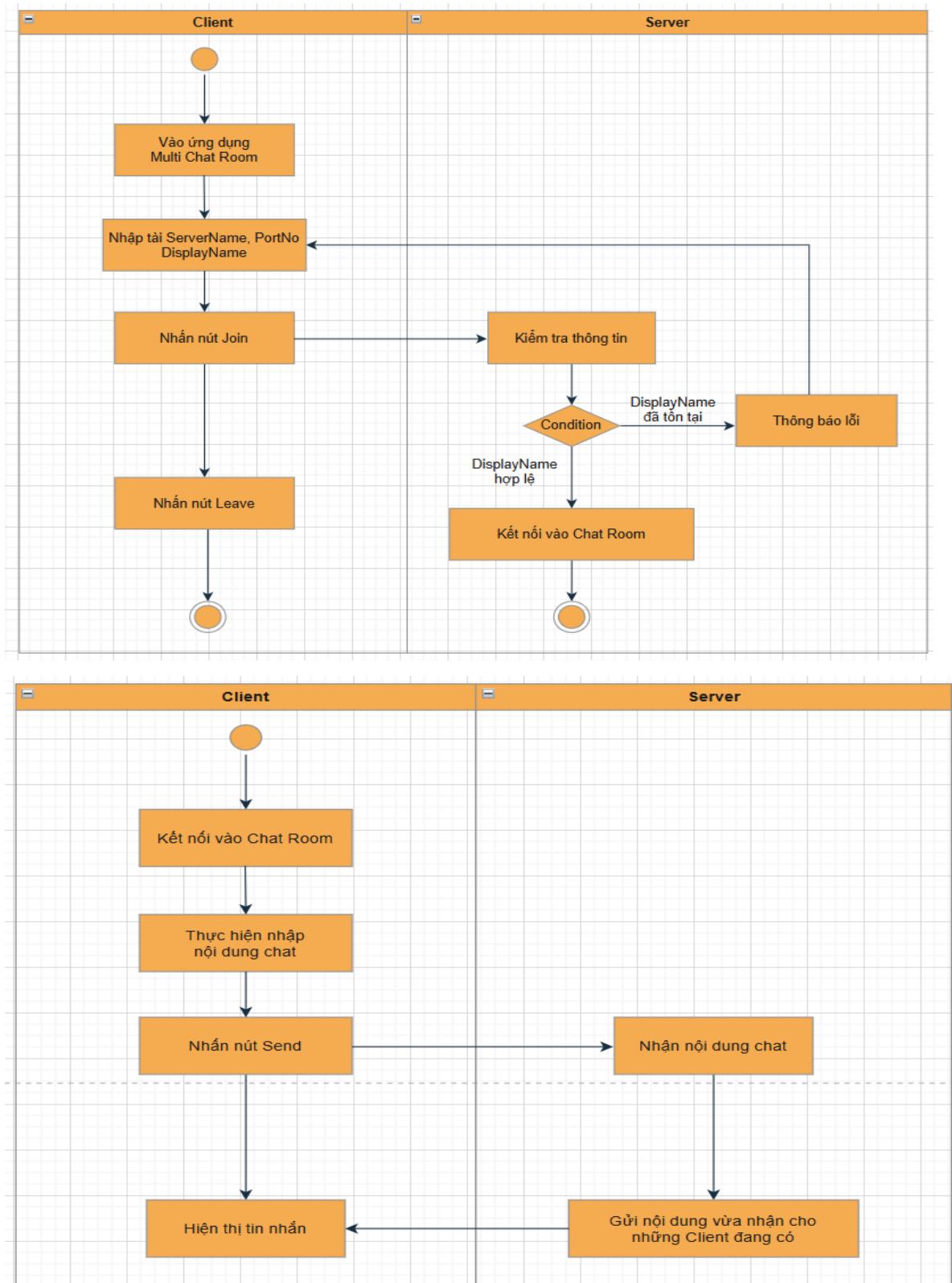
3.1 Mô hình đề xuất

3.1.1 Mô hình hệ thống tổng quan

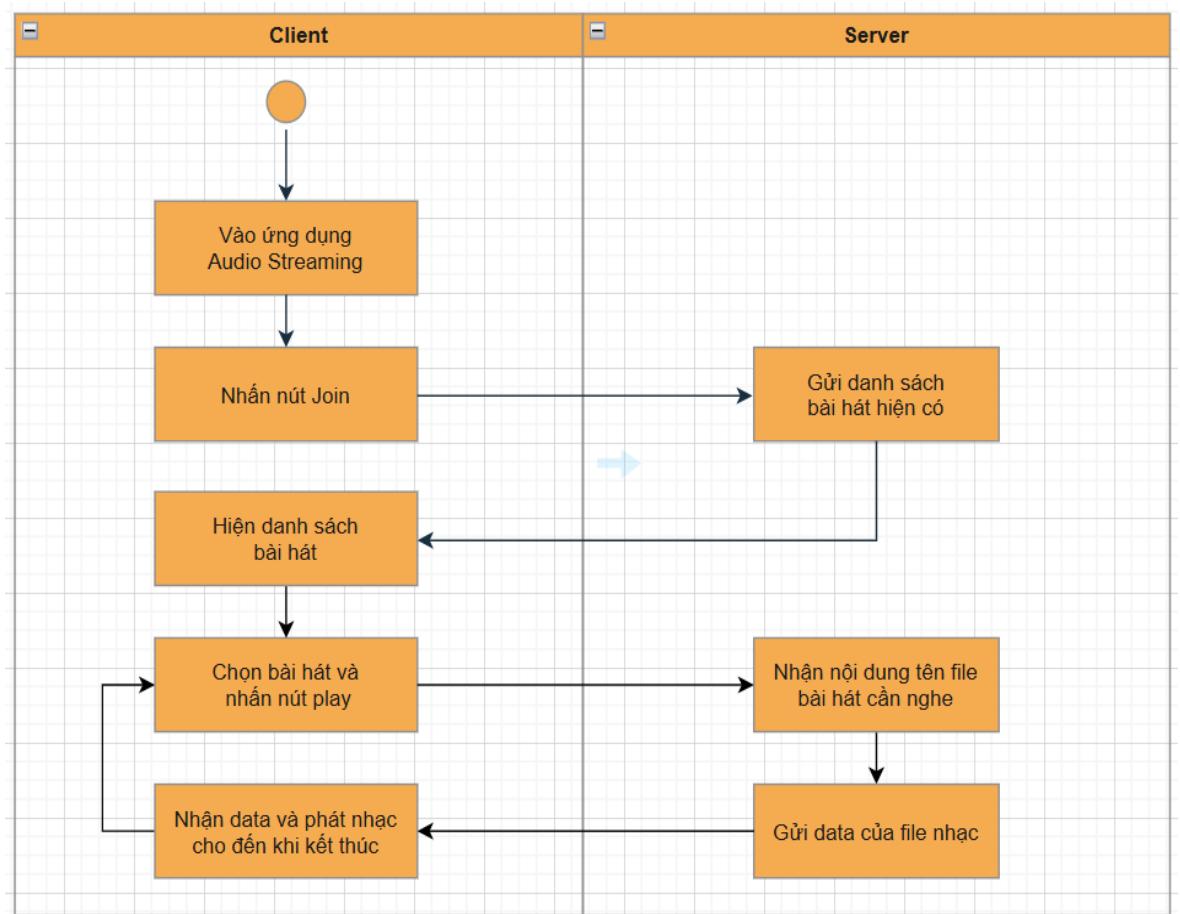
3.1.1.1 Biểu đồ tổng quát của đề tài



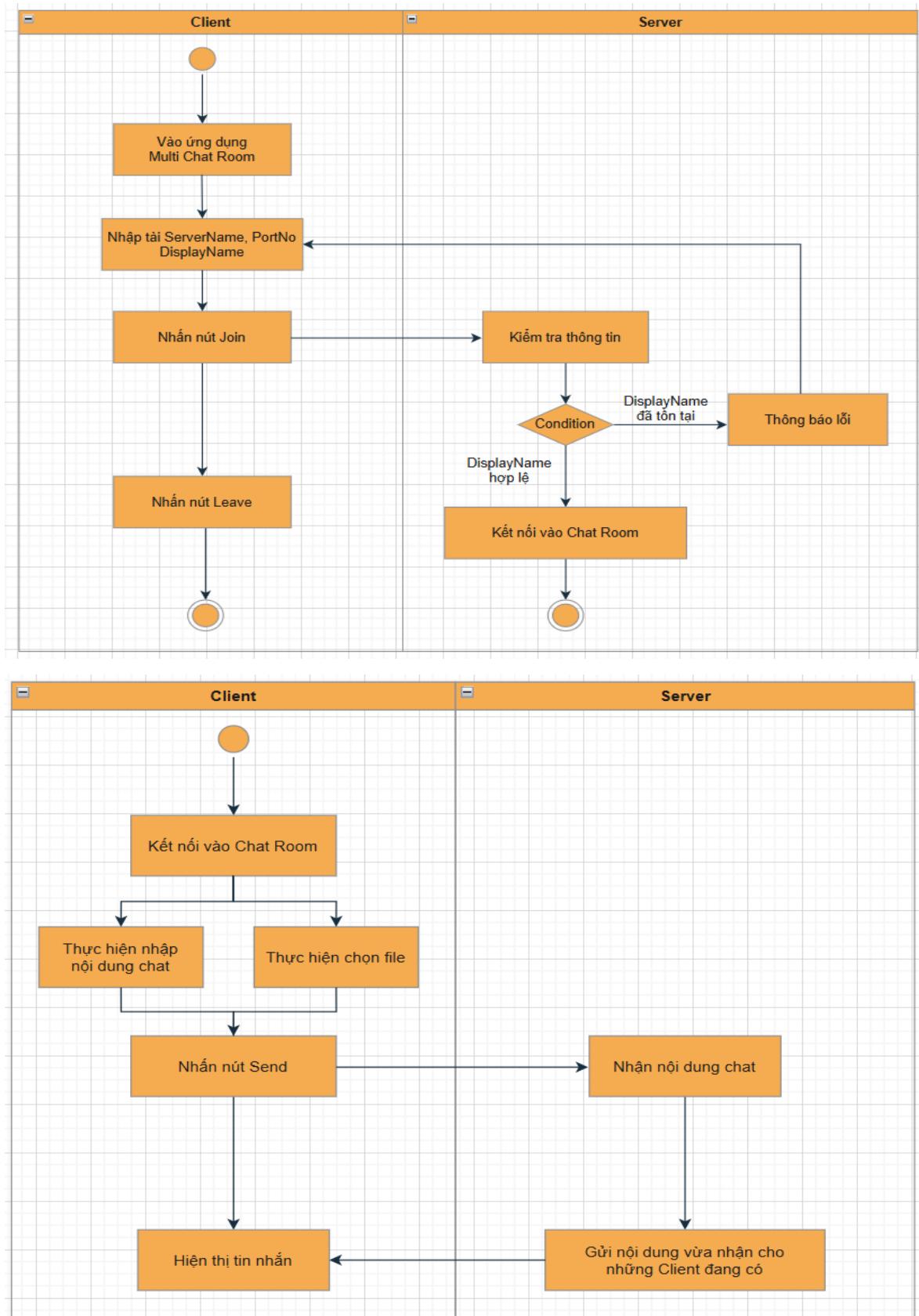
3.1.1.2 Biểu đồ hoạt động của từng chức năng



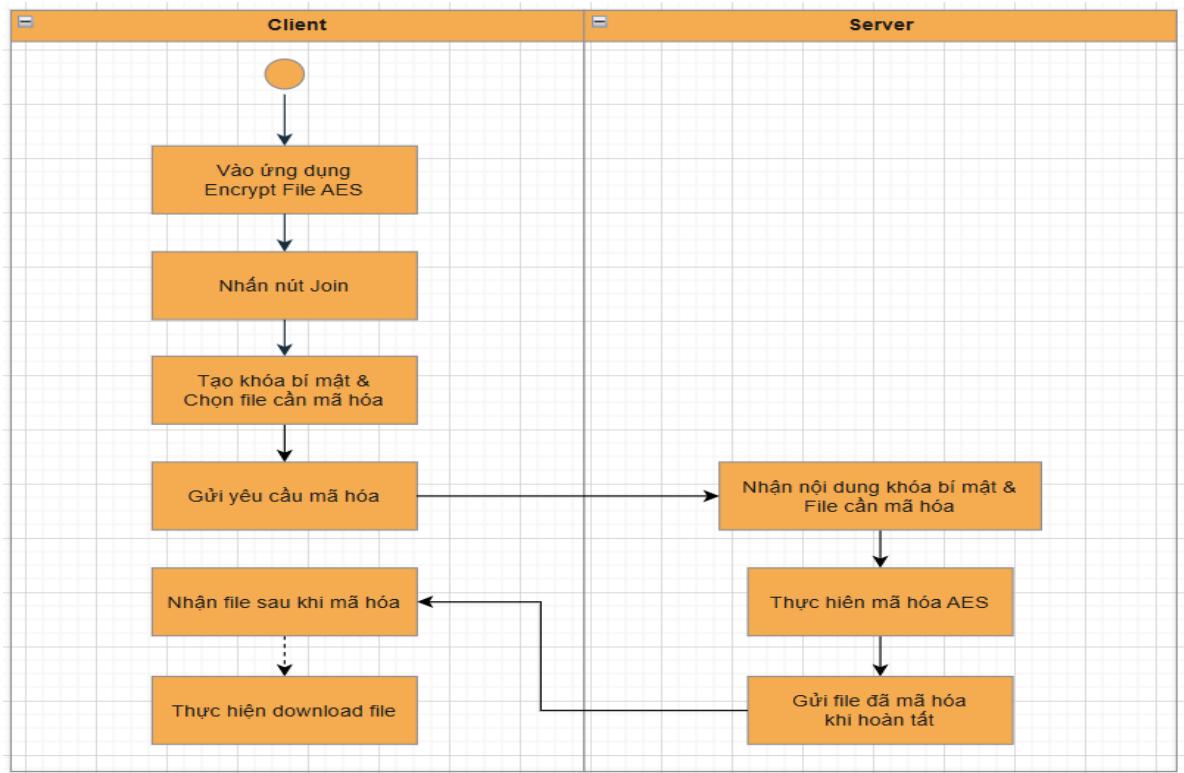
TCP – Chương trình Multi Chat Room (Chat nhiều người)



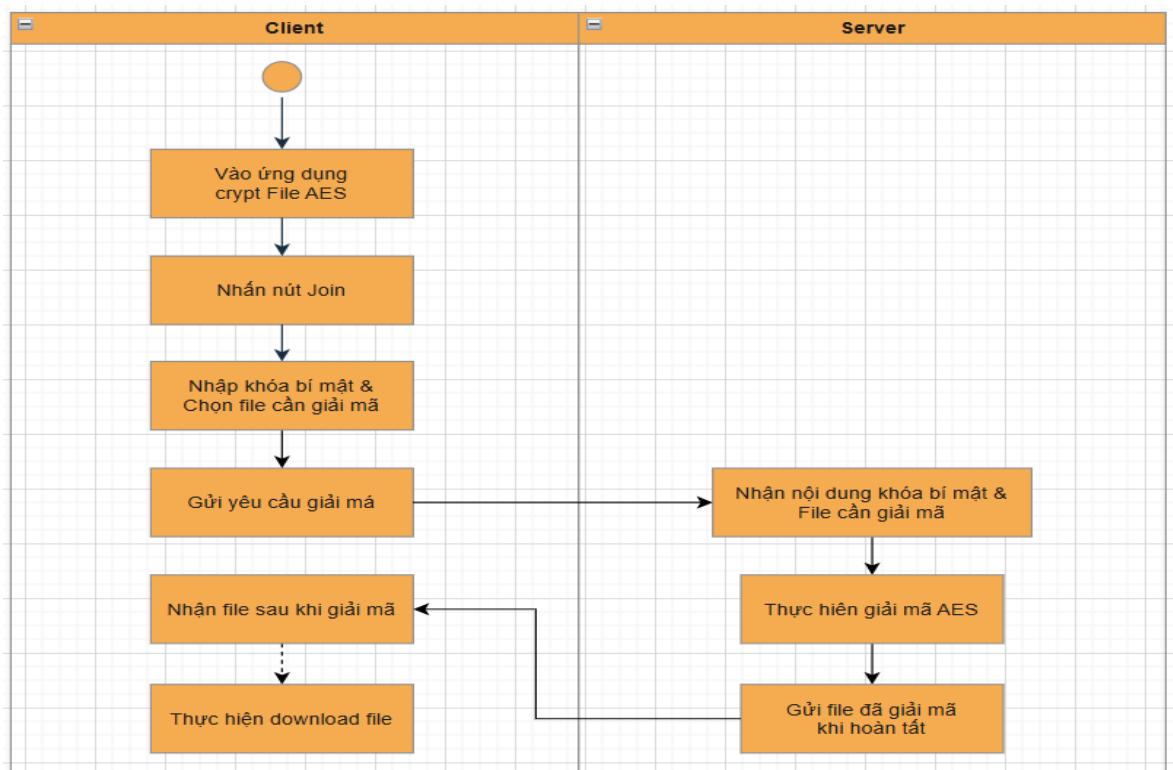
TCP – Chương trình Audio Streaming (Phát thanh, phát nhạc)



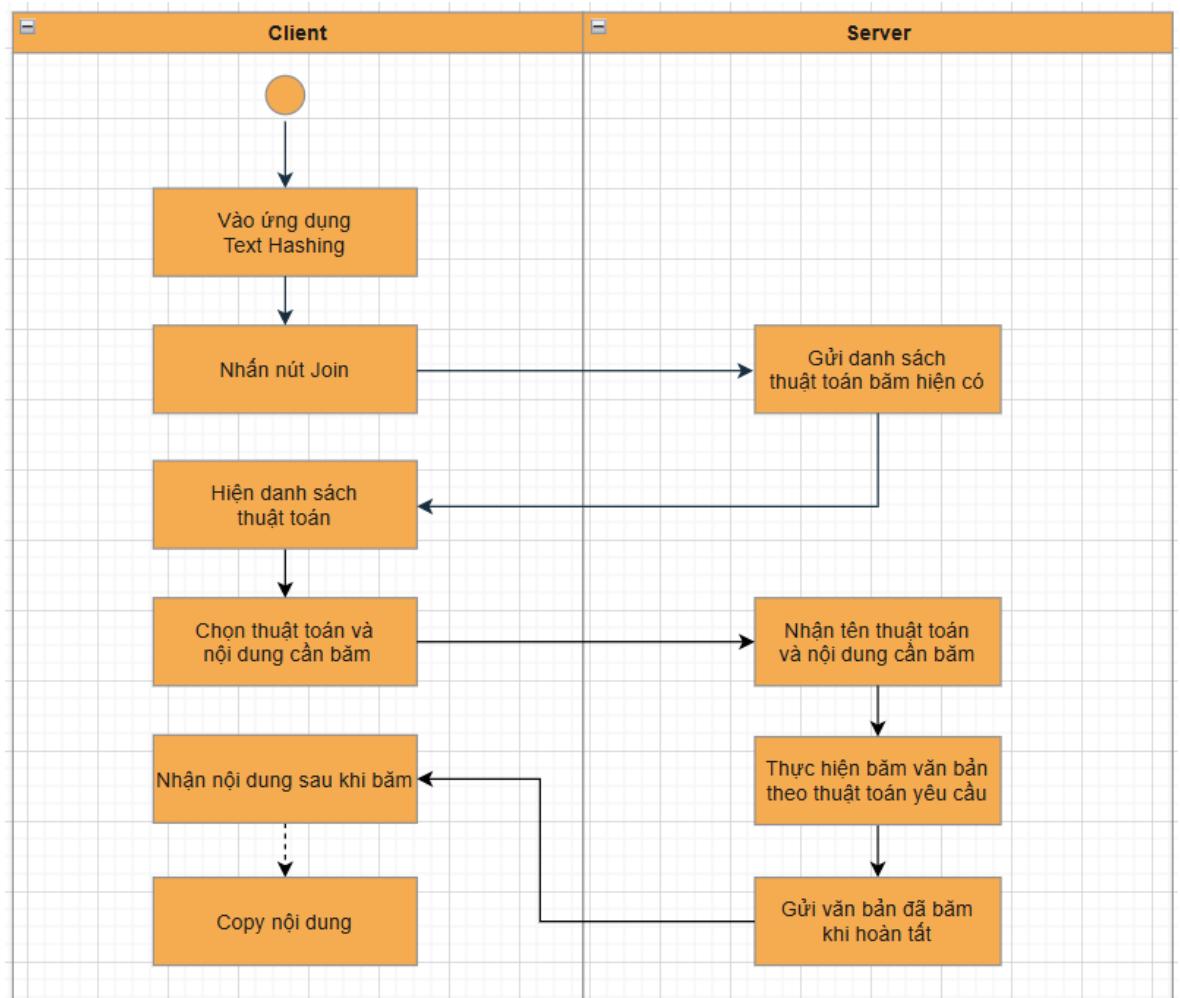
TCP – Chương trình File Transfer (Truyền file)



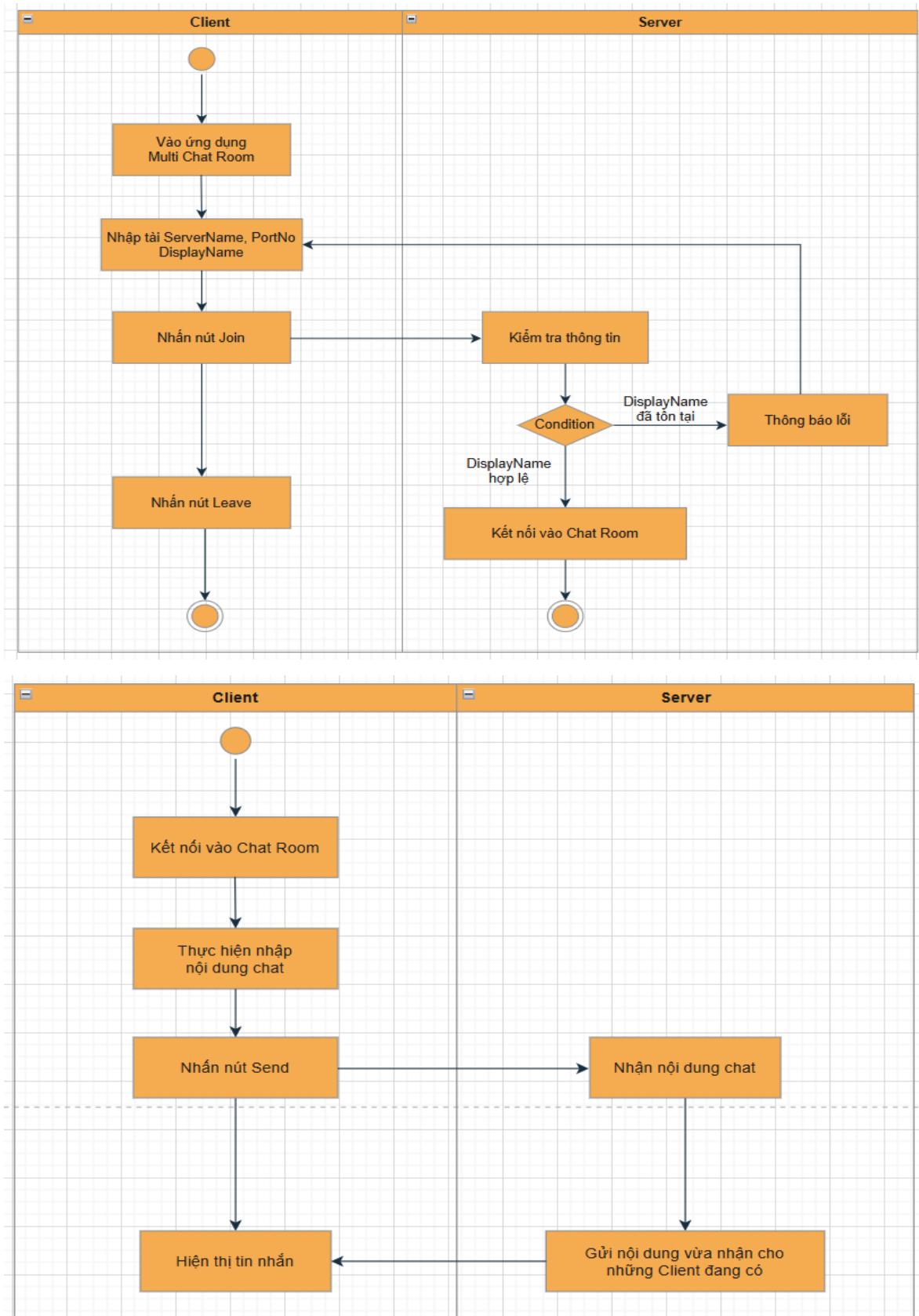
TCP – Chương trình File Encrypt AES (Mã hóa file)



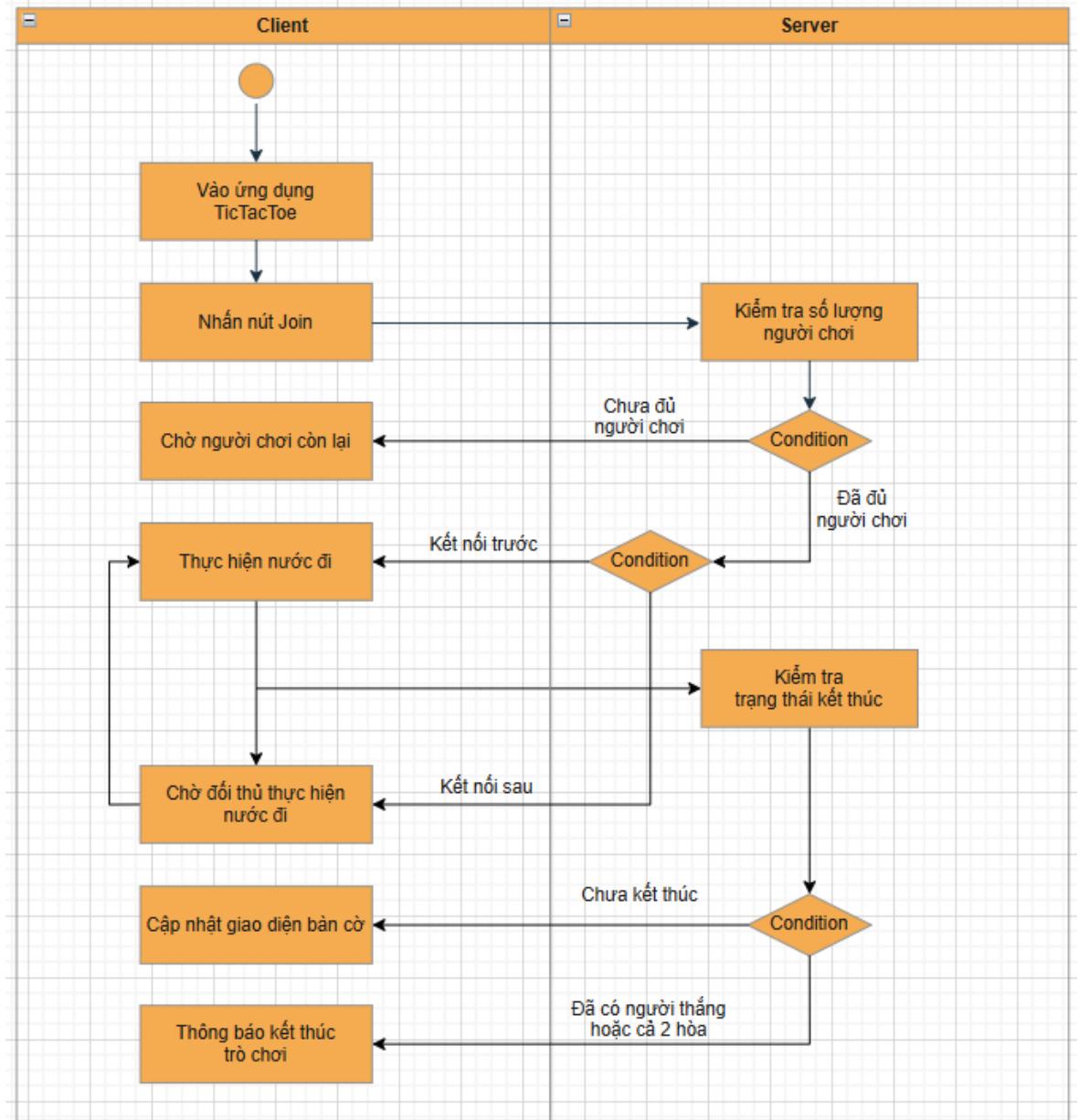
TCP – Chương trình File Decrypt AES (Giải mã file)



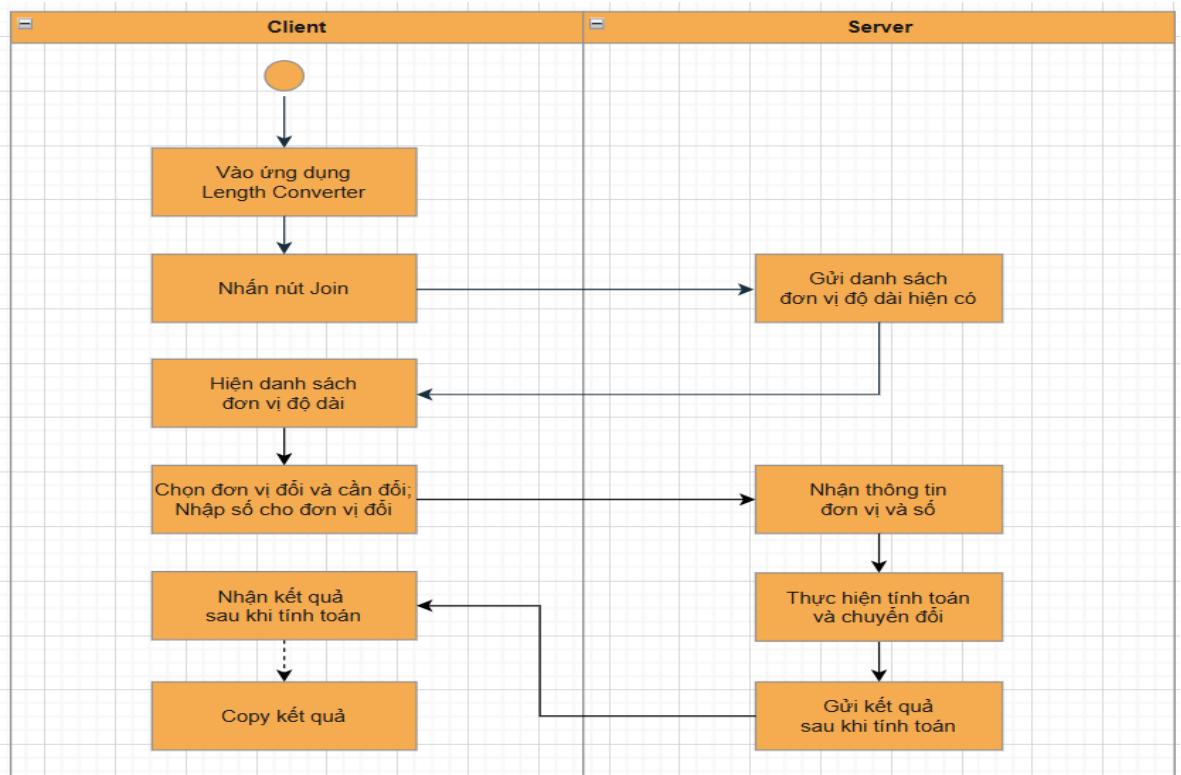
TCP – Chương trình Text Hashing (Mã hóa băm văn bản)



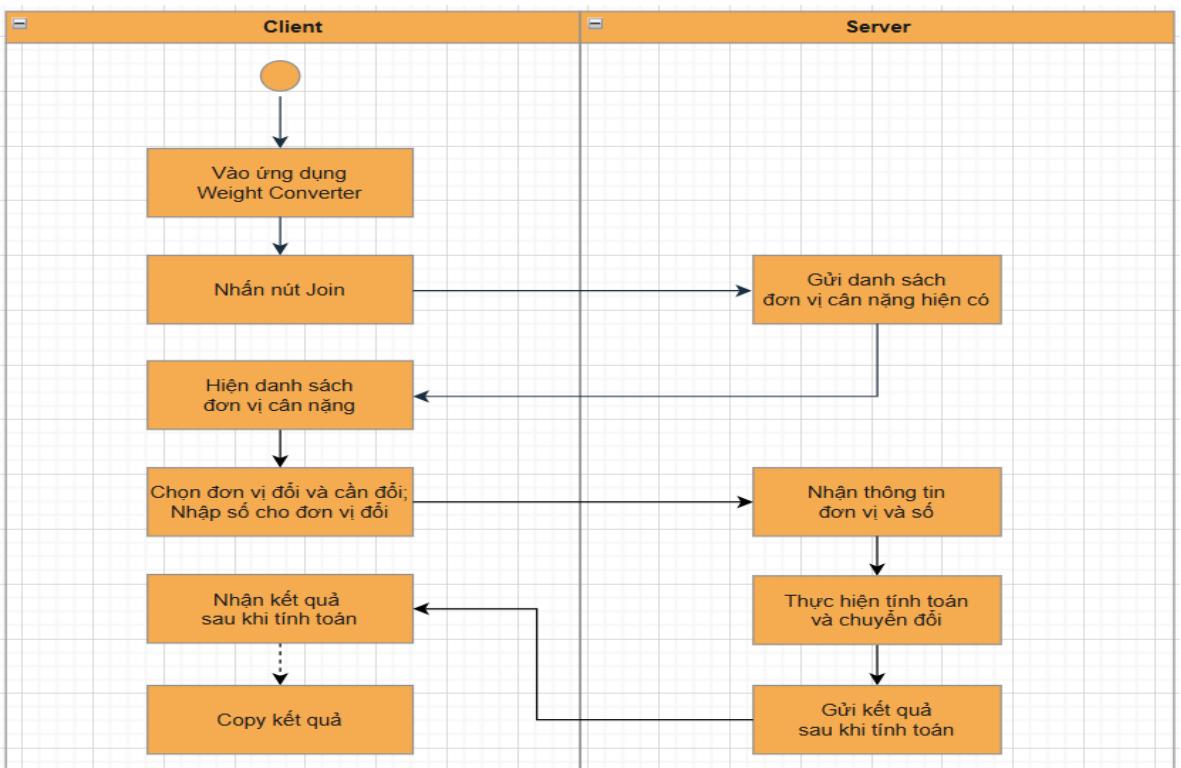
UDP – Chương trình Multi Chat Room (Chat nhiều người)



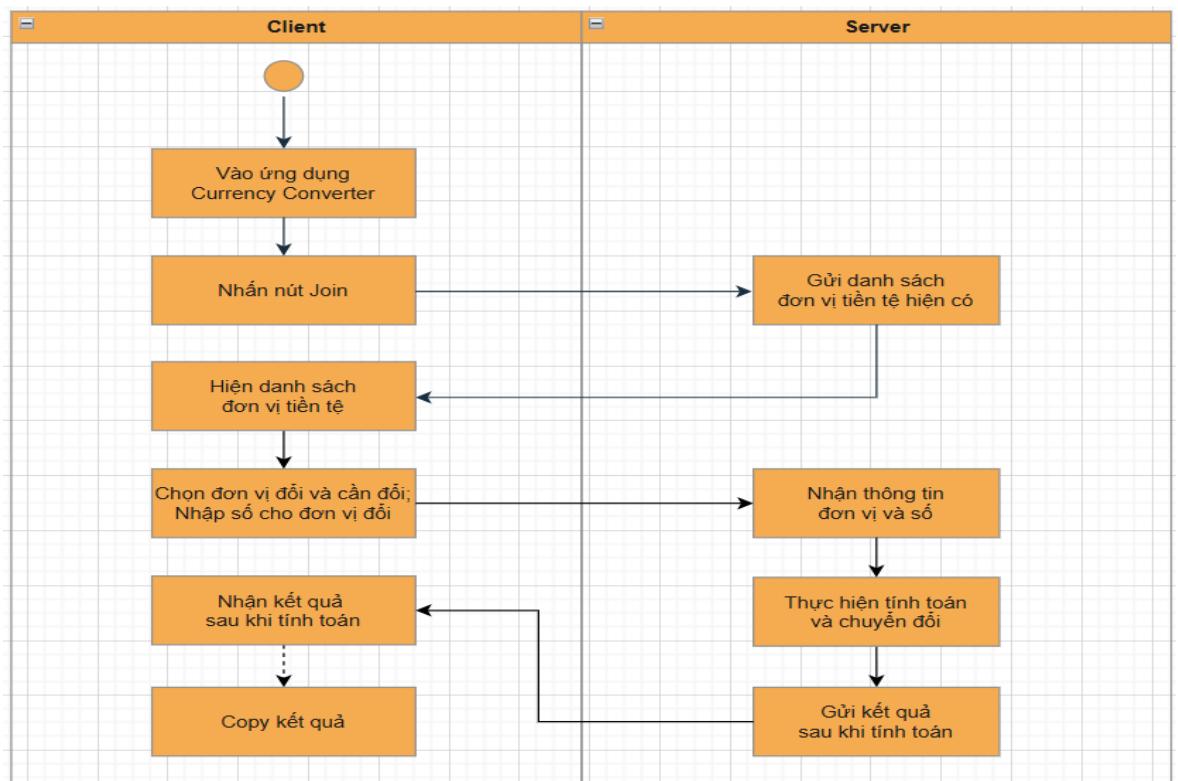
UDP – Chương trình TicTacToe Game (Trò chơi XO)



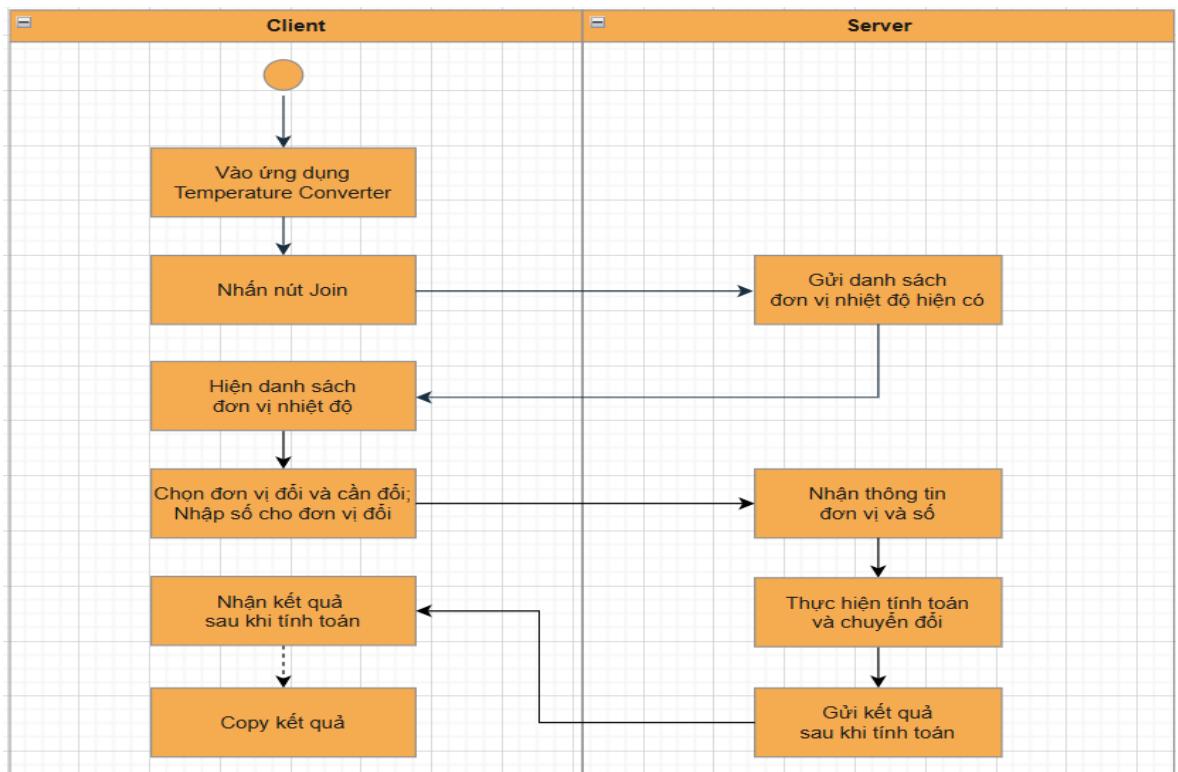
UDP – Chương trình Length Converter (Chuyển đổi độ dài)



UDP – Chương trình Weight Converter (Chuyển đổi cân nặng)



UDP – Chương trình Currency Converter (Chuyển đổi tiền tệ)



UDP – Chương trình Temperature Converter (Chuyển đổi nhiệt độ)

3.1.2 Xây dựng chương trình

3.1.2.1 TCP – Chương trình Multi Chat Room (Chat nhiều người)

- Server

```

1 package main.TCP.ChatRoom;
2
3 import java.io.*;
4 import java.net.*;
5 import java.util.*;
6 import javax.swing.*;
7
8 public class TCPChatServer extends javax.swing.JFrame {
9
10    private ServerSocket serverSocket;
11    private boolean connected = false;
12    private final Set<TCPClientInfo> connectedClients = new HashSet<>();
13
14    public TCPChatServer() {
15        initComponents();
16    }

```

Khai báo biến:

- + ServerSocket serverSocket: Đôi tượng server socket để chấp nhận kết nối từ client.
- + boolean connected: Biến trạng thái để kiểm tra xem server đang hoạt động hay không.
- + Set<TCPClientInfo> connectedClients: Danh sách các client đang kết nối, sử dụng HashSet để lưu trữ thông tin client.

Phương thức TCPChatServer():

Phương thức này chỉ gọi initComponents() để khởi tạo giao diện người dùng.

```

101   private void btnHandleActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnHandleActionPerformed
102       if (!connected) {
103           startServer();
104       } else {
105           stopServer();
106       }
107   }

```

Phương thức btnHandleActionPerformed():

- + Được gọi khi người dùng nhấn vào nút "Start/Stop".
- + Nếu máy chủ chưa được kết nối (connected == false), phương thức startServer() được gọi để khởi động máy chủ. Ngược lại, stopServer() được gọi để dừng máy chủ.

```

109 private void startServer() {
110     String port = textFieldPort.getText().trim();
111
112     try {
113         int Iport = Integer.parseInt(port);
114
115         if (Iport < 1 || Iport > 65535) {
116             throw new IllegalArgumentException("Port number must be between 1 and 65535");
117         }
118
119         serverSocket = new ServerSocket(Iport);
120
121         connected = true;
122         Thread serverThread = new Thread(() -> {
123             while (!serverSocket.isClosed()) {
124                 try {
125                     Socket clientSocket = serverSocket.accept();
126
127                     Thread clientHandlerThread = new Thread(() -> {
128                         try {
129                             DataInputStream dataIn = new DataInputStream(clientSocket.getInputStream());
130                             DataOutputStream dataOut = new DataOutputStream(clientSocket.getOutputStream());
131
132                             // Đọc tên người dùng từ client
133                             String username = dataIn.readUTF();
134
135                             // Kiểm tra xem tên người dùng đã tồn tại hay chưa
136                             boolean usernameExists = connectedClients.stream()
137                                 .anyMatch(client -> client.getUsername().equals(username));
138
139                             if (usernameExists) {
140                                 dataOut.writeUTF("USERNAME_EXISTS");
141                                 return; // Kết thúc luồng nếu tên người dùng đã tồn tại
142                             } else {
143                                 dataOut.writeUTF("USERNAME_ACCEPTED");
144
145                                 // Gửi danh sách các tên người dùng hiện tại cho client mới
146                                 StringBuilder usersList = new StringBuilder();
147                                 if (!connectedClients.isEmpty()) {
148                                     synchronized (connectedClients) {
149                                         for (TCPclientInfo client : connectedClients) {
150                                             usersList.append(client.getUsername()).append(", ");
151                                         }
152                                         usersList.setLength(usersList.length() - 2);
153                                         usersList.append(" & You are already in the chat room");
154                                     } else {
155                                         usersList.append("You are the only one in the chat room");
156                                     }
157
158                                 dataOut.writeUTF(usersList.toString());
159
160                                 // Tạo một ClientInfo mới và thêm vào danh sách connectedClients
161                                 TCPClientInfo client = new TCPClientInfo(username, dataOut, clientSocket);
162                                 synchronized (connectedClients) {
163                                     connectedClients.add(client);
164                                 }
165
166                                 // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
167                                 onlineUsersTextArea.append(username + " has connected to the chat room\n");
168
169                                 // Thông báo client mới đã tham gia cho các client trước đó
170                                 broadcastMessage(username, "has joined the chat room");
171
172                                 while (connected) {
173                                     String msgIn = dataIn.readUTF();
174
175                                     if ("LEAVE_CHAT_ROOM".equals(msgIn)) {
176                                         // Ngắt kết nối khi client thoát
177                                         synchronized (connectedClients) {
178                                             connectedClients.remove(client);
179                                         }
180                                         clientSocket.close();
181
182                                         // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
183                                         onlineUsersTextArea.append(client.getUsername() + " has disconnected from the chat room\n");
184
185                                         // Thông báo client đã rời khỏi phòng cho các client khác
186                                         broadcastMessage(client.getUsername(), "has left the chat room");
187                                         break;
188                                     }
189
190                                     broadcastMessage(username, msgIn);
191
192                                 }
193
194                                 } catch (IOException e) {
195                                     // Xử lý lỗi khi giao tiếp với client
196                                     if (connected) {
197                                         onlineUsersTextArea.append("An error occurred while receiving the message\n");
198                                     }
199                                 }
200
201                             });
202
203                             clientHandlerThread.start();
204                         } catch (IOException e) {
205                             if (!serverSocket.isClosed()) {
206                                 JOptionPane.showMessageDialog(null, "Server connection error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
207                             }
208                         }
209                     });
210
211                     serverThread.start();
212
213                     btnHandle.setText("Stop");
214                     textFieldPort.setEnabled(false);
215                     onlineUsersTextArea.setText("Server was started ...\\n\\n");
216
217                     } catch (NumberFormatException ex) {
218                         JOptionPane.showMessageDialog(null, "Malformed port: " + port, "Invalid Port Number", JOptionPane.WARNING_MESSAGE);
219                     } catch (IllegalArgumentException ex) {
220                         JOptionPane.showMessageDialog(null, ex.getMessage(), "Invalid Port Number", JOptionPane.WARNING_MESSAGE);
221                     } catch (IOException ex) {
222                         JOptionPane.showMessageDialog(null, ex.getMessage(), "Server Connection Error", JOptionPane.ERROR_MESSAGE);
223                     }
224                 }
225             }

```

Phương thức startServer():

- + Phương thức này được gọi khi người dùng muốn bắt đầu máy chủ.
- + Đầu tiên, nó kiểm tra cổng mà người dùng đã nhập. Nếu cổng không hợp lệ, sẽ hiển thị một hộp thoại cảnh báo.
- + Sau đó, nó tạo một ServerSocket trên cổng đã chọn và bắt đầu lắng nghe các kết nối từ client.
- + Một luồng mới được tạo ra cho mỗi client mới, cho phép xử lý đa luồng.
- + Mỗi khi một client kết nối thành công, một luồng mới được tạo để xử lý kết nối với client đó.

```

227  private void stopServer() {
228      try {
229          connected = false;
230
231          // Gửi yêu cầu ngắt kết nối đến tất cả các client
232          synchronized (connectedClients) {
233              for (TCPClientInfo client : connectedClients) {
234                  client.getDataOut().writeUTF("SERVER_SHUTDOWN");
235              }
236          }
237
238          // Đóng tất cả các kết nối client
239          synchronized (connectedClients) {
240              for (TCPClientInfo client : connectedClients) {
241                  client.getSocket().close();
242              }
243              connectedClients.clear();
244          }
245
246          // Đóng ServerSocket
247          if (serverSocket != null && !serverSocket.isClosed()) {
248              serverSocket.close();
249          }
250
251          btnHandle.setText("Start");
252          textFieldPort.setEnabled(true);
253          onlineUsersTextArea.append("Server has been stopped.\n");
254
255      } catch (IOException e) {
256          JOptionPane.showMessageDialog(null, "Error while stopping the server: " + e.getMessage(), "Server Error", JOptionPane.ERROR_MESSAGE);
257      }
258  }

```

Phương thức stopServer():

- + Phương thức này được gọi khi người dùng muốn dừng máy chủ.
- + Trước tiên, nó gửi một tin nhắn tới tất cả các client đang kết nối để thông báo rằng máy chủ sẽ tắt.
- + Tiếp theo, nó đóng tất cả các kết nối client và đóng ServerSocket.

```

260     private void broadcastMessage(String username, String message) {
261         synchronized (connectedClients) {
262             for (TCPClientInfo client : connectedClients) {
263                 if (!client.getUsername().equals(username)) {
264                     try {
265                         client.getDataOut().writeUTF("\n" + username + ": " + message);
266                     } catch (IOException e) {
267                         JOptionPane.showMessageDialog(null, "Error broadcasting message: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
268                     }
269                 }
270             }
271         }
272     }

```

Phương thức broadcastMessage():

- + Phương thức này gửi một tin nhắn từ một người dùng đến tất cả các client khác.
- + Nó lặp qua tất cả các client trong connectedClients và gửi tin nhắn đến từng client.

```

274     private void formWindowClosing(java.awt.event.WindowEvent evt) { //GEN-FIRST:event_formWindowClosing
275         if (serverSocket != null && !serverSocket.isClosed()) {
276             stopServer();
277         }
278     }

```

Phương thức formWindowClosing():

- + Phương thức này được gọi khi cửa sổ đang được đóng.
- + Nếu máy chủ đang chạy, nó sẽ gọi stopServer() để đảm bảo rằng tất cả các kết nối client được đóng trước khi đóng cửa sổ.

- **Client**

```

1 package main.TCP.ChatRoom;
2
3 import java.io.*;
4 import java.net.*;
5 import javax.swing.*;
6 import javax.swing.event.*;
7 import javax.swing.text.*;
8
9 public class TCPChatClient extends javax.swing.JFrame {
10
11     private Socket socket;
12     private DataInputStream dataIn;
13     private DataOutputStream dataOut;
14     private boolean connected = false;
15
16     public TCPChatClient() {
17         initComponents();
18         DocumentListener validateInput = new DocumentListener() {
19
20             @Override
21             public void insertUpdate(DocumentEvent e) {
22                 validateTextField();
23             }
24
25             @Override
26             public void removeUpdate(DocumentEvent e) {
27                 validateTextField();
28             }
29
30             @Override
31             public void changedUpdate(DocumentEvent e) {
32                 validateTextField();
33             }
34
35             protected void validateTextField() {
36                 String inputServerName = tfdServerName.getText().trim();
37                 String inputPort = tfdPort.getText().trim();
38                 String inputName = tfdName.getText().trim();
39
40                 if (!inputServerName.equals("") && !inputPort.equals("") && !inputName.equals("")) {
41                     btnHandle.setEnabled(true);
42                 } else {
43                     btnHandle.setEnabled(false);
44                 }
45             }
46         };
47
48         DocumentListener validateMsg = new DocumentListener() {
49
50             @Override
51             public void insertUpdate(DocumentEvent e) {
52                 validateTextField();
53             }
54
55             @Override
56             public void removeUpdate(DocumentEvent e) {
57                 validateTextField();
58             }
59
60             @Override
61             public void changedUpdate(DocumentEvent e) {
62                 validateTextField();
63             }
64
65             protected void validateTextField() {
66                 String inputMsg = msgArea.getText().trim();
67                 if (!inputMsg.equals("")) {
68                     btnSend.setEnabled(true);
69                 } else {
70                     btnSend.setEnabled(false);
71                 }
72             }
73         };
74
75         tfdServerName.getDocument().addDocumentListener(validateInput);
76         tfdPort.getDocument().addDocumentListener(validateInput);
77         tfdName.getDocument().addDocumentListener(validateInput);
78         msgArea.getDocument().addDocumentListener(validateMsg);
79     }

```

Khai báo biến:

- + Socket socket: Đối tượng socket để kết nối với server.
- + DataInputStream dataIn: Đối tượng để đọc dữ liệu từ server.
- + DataOutputStream dataOut: Đối tượng để gửi dữ liệu đến server.
- + boolean connected: Biến trạng thái để kiểm tra xem client đang kết nối hay không.

Phương thức TCPChatClient():

- + Phương thức này khởi tạo giao diện
- + Thiết lập thêm các DocumentListener để kiểm tra và xác thực đầu vào từ người dùng.

```

214  private void btnHandleActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnHandleActionPerformed
215      if (!connected) {
216          joinChatRoom();
217      } else {
218          leaveChatRoom();
219      }
220  }

```

Phương thức btnHandleActionPerformed():

- + Xử lý sự kiện khi người dùng nhấn vào nút "Join/Leave".
- + Nếu máy khách chưa kết nối (connected == false), phương thức joinChatRoom() được gọi để tham gia vào phòng chat. Ngược lại, leaveChatRoom() được gọi để rời khỏi phòng chat.

```

222 private void joinChatRoom() {
223     String SERVER = tfdServerName.getText().trim();
224     String inputPORT = tfdPort.getText().trim();
225     String USERNAME = tfdName.getText().trim();
226
227     try {
228         int PORT = Integer.parseInt(inputPORT);
229
230         if (PORT < 1 || PORT > 65535) {
231             throw new IllegalArgumentException("Port number must be between 1 and 65535");
232         }
233
234         socket = new Socket(SERVER, PORT);
235         dataIn = new DataInputStream(socket.getInputStream());
236         dataOut = new DataOutputStream(socket.getOutputStream());
237
238         // Gửi tên người dùng đến server để đăng ký
239         dataOut.writeUTF(USERNAME);
240         String response = dataIn.readUTF();
241
242         if ("USERNAME_EXISTS".equals(response)) {
243             JOptionPane.showMessageDialog(null, "Username already taken. Please choose another name.", "Username Error", JOptionPane.ERROR_MESSAGE);
244             socket.close();
245             return;
246         } else if ("USERNAME_ACCEPTED".equals(response)) {
247             appendToPane(chatDataPane, "Connected to the server as \\" + USERNAME + "\\n", StyleConstants.ALIGN_LEFT);
248         }
249
250         // Tạo luồng để nhận tin nhắn từ server
251         connected = true;
252         Thread clientThread = new Thread(() -> {
253             try {
254                 while (connected) {
255                     String msgIn = dataIn.readUTF();
256
257                     // Kiểm tra nếu server gửi yêu cầu ngắt kết nối
258                     if ("SERVER_SHUTDOWN".equals(msgIn)) {
259                         connected = false;
260                         appendToPane(chatDataPane, "Server has been shut down. You have been disconnected.\n", StyleConstants.ALIGN_LEFT);
261                         toggleInput(connected);
262                         socket.close();
263                         break;
264                     }
265
266                     appendToPane(chatDataPane, msgIn + "\n", StyleConstants.ALIGN_LEFT);
267                 }
268             } catch (IOException e) {
269                 // Xử lý lỗi khi giao tiếp với server
270                 if (connected) {
271                     appendToPane(chatDataPane, "\nAn error occurred while receiving the message! Please try connecting again.\n", StyleConstants.ALIGN_LEFT);
272                     leaveChatRoom();
273                 }
274             }
275         });
276         clientThread.start();
277         toggleInput(connected);
278         this.setTitle(USERNAME);
279     }
280     } catch (NumberFormatException ex) {
281         JOptionPane.showMessageDialog(null, "Malformed port: " + inputPORT, "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
282     } catch (IllegalArgumentException ex) {
283         JOptionPane.showMessageDialog(null, ex.getMessage(), "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
284     } catch (UnknownHostException ex) {
285         JOptionPane.showMessageDialog(null, "Unknown server: " + SERVER, "Unknown Host", JOptionPane.ERROR_MESSAGE);
286     } catch (IOException ex) {
287         JOptionPane.showMessageDialog(null, "No corresponding server found", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
288     }
289 }
290 }
```

Phương thức joinChatRoom():

- + Phương thức này được gọi khi người dùng muốn tham gia vào phòng chat.
- + Nó xác thực thông tin máy chủ và tên người dùng.
- + Sau đó, nó tạo một Socket để kết nối với máy chủ và mở các luồng dữ liệu vào/ra.
- + Nếu tên người dùng đã tồn tại, nó hiển thị một thông báo lỗi và đóng kết nối. Nếu không, nó tham gia vào phòng chat và bắt đầu lắng nghe tin nhắn từ máy chủ.

```

292  private void leaveChatRoom() {
293      try {
294          // Gửi tin nhắn rời phòng chat tới server
295          dataOut.writeUTF("LEAVE_CHAT_ROOM");
296          connected = false; // Dừng luồng nhận tin nhắn
297          chatDataPane.setText("");
298          JOptionPane.showMessageDialog(null, "You have successfully left the chat room.", "Successfully", JOptionPane.INFORMATION_MESSAGE);
299          socket.close(); // Đóng socket đến server
300      } catch (IOException e) {
301          e.printStackTrace();
302      }
303
304      toggleInput(connected);
305  }

```

Phương thức leaveChatRoom():

- + Phương thức này được gọi khi người dùng muốn rời khỏi phòng chat.
- + Nó gửi một tin nhắn tới máy chủ để thông báo rằng người dùng muốn rời khỏi phòng chat.
- + Sau đó, nó đóng các luồng dữ liệu và kết nối socket.

```

307  private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnSendActionPerformed
308      String inputMsg = msgArea.getText().trim();
309
310      try {
311          dataOut.writeUTF(inputMsg);
312          appendToPane(chatDataPane, inputMsg + "\n", StyleConstants.ALIGN_RIGHT);
313      } catch (IOException ex) {
314          JOptionPane.showMessageDialog(null, "An error occurred while sending the message! Please try connecting again.", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
315      }
316
317      msgArea.setText("");
318  }

```

Phương thức btnSendActionPerformed():

- + Xử lý sự kiện khi người dùng nhấn nút "Send".
- + Nó gửi tin nhắn mà người dùng nhập vào cho máy chủ và hiển thị nó trên giao diện.

```

320  private void formWindowClosing(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_formWindowClosing
321      if (socket != null && !socket.isClosed()) {
322          leaveChatRoom();
323      }
324  }

```

Phương thức formWindowClosing()

- + Phương thức này được gọi khi cửa sổ đang được đóng.
- + Nếu máy khách đang kết nối, nó sẽ gọi leaveChatRoom() để đảm bảo rằng kết nối được đóng trước khi đóng cửa sổ.

```

326  private void toggleInput(boolean connected) {
327      btnHandle.setText(connected ? "Leave" : "Join");
328      tfdServerName.setEnabled(!connected);
329      tfdPort.setEnabled(!connected);
330      tfdName.setEnabled(!connected);
331      msgArea.setText("");
332      msgArea.setEnabled(connected);
333  }

335  private void appendToPane(JTextPane tp, String msg, int alignment) {
336      StyledDocument doc = tp.getStyledDocument();
337      SimpleAttributeSet attr = new SimpleAttributeSet();
338      StyleConstants.setAlignment(attr, alignment);
339
340      try {
341          doc.insertString(doc.getLength(), msg, attr);
342          doc.setParagraphAttributes(doc.getLength() - msg.length(), msg.length(), attr, false);
343      } catch (BadLocationException e) {
344          e.printStackTrace();
345      }
346  }

```

Các phương thức hỗ trợ khác:

- + *toggleInput(boolean connected)*: Cập nhật trạng thái các thành phần giao diện dựa trên trạng thái kết nối.
- + *appendToPane(JTextPane tp, String msg, int alignment)*: Thêm tin nhắn vào JTextPane với căn lề chỉ định. Nó cũng định dạng văn bản để hiển thị tin nhắn của máy khách và máy chủ theo hai bên của giao diện.

- **TCPClientInfo Class**

```
1 package main.TCP.ChatRoom;
2
3 import java.io.DataOutputStream;
4 import java.net.Socket;
5
6 public class TCPClientInfo {
7
8     private String username;
9     private DataOutputStream dataOut;
10    private Socket socket;
11
12    public TCPClientInfo(String username, DataOutputStream dataOut, Socket socket) {
13        this.username = username;
14        this.dataOut = dataOut;
15        this.socket = socket;
16    }
17
18    public String getUsername() {
19        return username;
20    }
21
22    public DataOutputStream getDataOut() {
23        return dataOut;
24    }
25
26    public Socket getSocket() {
27        return socket;
28    }
29 }
30
```

3.1.2.2 TCP – Chương trình Audio Streaming (Phát thanh, phát nhạc)

- **Server**

```

1 package main.TCP.AudioStreaming;
2
3 import java.io.*;
4 import java.net.*;
5 import java.nio.file.*;
6 import java.util.ArrayList;
7 import java.util.Collections;
8 import java.util.stream.Stream;
9 import javax.sound.sampled.*;
10 import javax.swing.*;
11
12 public class TCPAudioStreamingServer extends javax.swing.JFrame {
13
14     private ServerSocket serverSocket;
15     private Socket clientSocket;
16     private DataInputStream dataIn;
17     private DataOutputStream dataOut;
18     private boolean connected = false;
19     private final String MUSIC_FOLDER_PATH = "/resource/music/";
20
21     public TCPAudioStreamingServer() {
22         initComponents();
23     }

```

Khai báo biến:

- + ServerSocket serverSocket: Đối tượng server socket để chấp nhận kết nối từ client.
- + Socket clientSocket: Đối tượng socket để giao tiếp với client.
- + DataInputStream dataIn: Để đọc dữ liệu từ client.
- + DataOutputStream dataOut: Để gửi dữ liệu đến client.
- + boolean connected: Biến trạng thái để kiểm tra kết nối.
- + final String MUSIC_FOLDER_PATH: Đường dẫn tới thư mục chứa nhạc.

Phương thức TCPAudioStreamingServer():

- + Khởi tạo giao diện người dùng bằng cách gọi phương thức initComponents().

```
108 private void btnHandleActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnHandleActionPerformed
109     if (!connected) {
110         startServer();
111     } else {
112         stopServer();
113     }
114 }
```

Phương thức btnHandleActionPerformed():

- + Xử lý sự kiện khi người dùng nhấp vào nút "Start/Stop".
- + Gọi startServer() nếu chưa kết nối, hoặc stopServer() nếu đã kết nối.

```

116 private void startServer() {
117     String port = textFieldPort.getText().trim();
118
119     try {
120         int Iport = Integer.parseInt(port);
121
122         if (Iport < 1 || Iport > 65535) {
123             throw new IllegalArgumentException("Port number must be between 1 and 65535");
124         }
125
126         serverSocket = new ServerSocket(Iport);
127
128         Thread serverThread = new Thread(() -> {
129             while (!serverSocket.isClosed()) {
130                 try {
131                     clientSocket = serverSocket.accept();
132
133                     Thread clientHandlerThread = new Thread(() -> {
134                         try {
135                             dataIn = new DataInputStream(clientSocket.getInputStream());
136                             dataOut = new DataOutputStream(clientSocket.getOutputStream());
137
138                             // Đọc yêu cầu từ client
139                             String request = dataIn.readUTF();
140
141                             if ("GET_SONG_LIST".equals(request)) {
142                                 onlineUsersTextArea.append("A new client has requested to send a playlist\n");
143
144                                 // Gửi danh sách bài hát về cho client
145                                 ArrayList<String> songList = getSonglistFromResources();
146                                 StringBuilder songListBuilder = new StringBuilder();
147                                 for (String song : songList) {
148                                     songListBuilder.append(song).append("\n");
149                                 }
150                                 dataOut.writeUTF(songListBuilder.toString());
151                             }
152
153                             while (connected) {
154                                 request = dataIn.readUTF();
155
156                                 if ("LEAVE_CHAT_ROOM".equals(request)) {
157                                     // Ngắt kết nối khi client thoát
158                                     clientSocket.close();
159                                     // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
160                                     onlineUsersTextArea.append("Client has disconnected to the audio streaming room\n");
161                                     break;
162                                 }
163
164                                 // Phát nhạc tương ứng với yêu cầu từ client
165                                 String songName = request.substring("PLAY_SONG|".length());
166                                 File file = new File(MUSIC_FOLDER_PATH + songName);
167                                 if (!file.exists()) {
168                                     dataOut.writeUTF("File not found");
169                                     continue;
170                                 }
171
172                                 // Gửi thông báo tìm thấy file và dữ liệu của file cho client
173                                 try (AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(file)) {
174                                     AudioFormat format = audioInputStream.getFormat();
175                                     System.out.println(format);
176
177                                     byte[] buffer = new byte[4096];
178                                     int bytesRead;
179                                     while ((bytesRead = audioInputStream.read(buffer)) != -1) {
180                                         dataOut.write(buffer, 0, bytesRead);
181                                     }
182                                 } catch (UnsupportedAudioFileException ex) {
183                                     ex.printStackTrace();
184                                 }
185                             }
186
187                         } catch (IOException e) {
188                             // Xử lý lỗi khi giao tiếp với client
189                             if (connected) {
190                                 onlineUsersTextArea.append("An error occurred while receiving the message\n");
191                             }
192                         }
193                     });
194
195                     connected = true;
196                     clientHandlerThread.start();
197                 } catch (IOException e) {
198                     if (!serverSocket.isClosed()) {
199                         JOptionPane.showMessageDialog(null, "Server connection error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
200                     }
201                 }
202             }
203         });
204
205         serverThread.start();
206
207         btnHandle.setText("Stop");
208         textFieldPort.setEnabled(false);
209         onlineUsersTextArea.setText("Server was started ...\\n\\n");
210
211     } catch (NumberFormatException ex) {
212         JOptionPane.showMessageDialog(null, "Malformed port: " + port, "Invalid Port Number", JOptionPane.WARNING_MESSAGE);
213     } catch (IllegalArgumentException ex) {
214         JOptionPane.showMessageDialog(null, ex.getMessage(), "Invalid Port Number", JOptionPane.WARNING_MESSAGE);
215     } catch (IOException ex) {
216         JOptionPane.showMessageDialog(null, ex.getMessage(), "Server Connection Error", JOptionPane.ERROR_MESSAGE);
217     }
218 }

```

Phương thức startServer():

- + Kiểm tra và lấy số cổng từ giao diện người dùng.
 - + Khởi tạo server socket và bắt đầu chấp nhận kết nối từ client trong một luồng mới.
 - + Khi client kết nối, một luồng mới được tạo ra để xử lý yêu cầu từ client.
- Xử lý các yêu cầu từ client, bao gồm yêu cầu danh sách bài hát và yêu cầu phát nhạc.

```

220  private ArrayList<String> getSongListFromResources() {
221      ArrayList<String> songList = new ArrayList<>();
222      try {
223          // Lấy đường dẫn tới thư mục hoặc JAR file chứa class này
224          String jarDirPath = new File(TCPAudioStreamingServer.class.getProtectionDomain().getCodeSource().getLocation().toURI()).getPath();
225          File jarDir = new File(jarDirPath);
226          File musicFolder;
227
228          if (jarDir.isDirectory()) {
229              // Nếu chạy từ IDE (thư mục)
230              musicFolder = new File(jarDir, MUSIC_FOLDER_PATH);
231              if (musicFolder.exists() && musicFolder.isDirectory()) {
232                  File[] musicFiles = musicFolder.listFiles();
233                  if (musicFiles != null) {
234                      for (File musicFile : musicFiles) {
235                          if (musicFile.isFile()) {
236                              songList.add(musicFile.getName());
237                          }
238                      }
239                  }
240              }
241          } else {
242              // Nếu chạy từ JAR (tệp nén)
243              URI jarURI = new URI("jar:file:" + jarDirPath);
244              try (FileSystem fileSystem = FileSystems.newFileSystem(jarURI, Collections.emptyMap())) {
245                  Path musicFolderPath = fileSystem.getPath(MUSIC_FOLDER_PATH);
246                  try (Stream<Path> paths = Files.walk(musicFolderPath, 1)) {
247                      paths.filter(Files::isRegularFile).forEach(path -> songList.add(path.getFileName().toString()));
248                  }
249              }
250          }
251      } catch (Exception e) {
252          e.printStackTrace();
253      }
254      return songList;
255  }

```

Phương thức getSongListFromResources():

- + Lấy danh sách các bài hát từ thư mục resource/music/ cho dù server đang chạy từ IDE hay từ tệp JAR.
- + Nếu chạy từ IDE, sử dụng File để lấy danh sách bài hát.
- + Nếu chạy từ JAR, sử dụng FileSystem để duyệt qua các tệp trong JAR.

```

257 private void stopServer() {
258     connected = false;
259
260     // Đóng ServerSocket
261     if (serverSocket != null && !serverSocket.isClosed()) {
262         try {
263             serverSocket.close();
264         } catch (IOException e) {
265             JOptionPane.showMessageDialog(null, "Error while stopping the server: " + e.getMessage(), "Server Error", JOptionPane.ERROR_MESSAGE);
266         }
267     }
268
269     btnHandle.setText("Start");
270     textFieldPort.setEnabled(true);
271     onlineUsersTextArea.append("Server has been stopped...\n");
272 }

```

Phương thức startServer():

- + Đặt connected thành false để dừng việc chấp nhận các kết nối mới.
- + Đóng ServerSocket nếu nó đang mở.
- + Cập nhật trạng thái giao diện và hiển thị thông báo rằng máy chủ đã dừng.

```

274 private void formWindowClosing(java.awt.event.WindowEvent evt) { //GEN-FIRST:event_formWindowClosing
275     if (serverSocket != null && !serverSocket.isClosed()) {
276         stopServer();
277     }
278 }

```

Phương thức stopServer():

- + Gọi khi cửa sổ đang được đóng.
- + Nếu máy chủ đang kết nối, gọi stopServer() để đảm bảo kết nối được đóng trước khi đóng cửa sổ.

- **Client**

```

1 package main.TCP.AudioStreaming;
2
3 import java.io.*;
4 import java.net.*;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7 import javax.sound.sampled.*;
8 import javax.swing.*;
9 import javax.swing.event.*;
10
11 public class TCPAudioStreamingClient extends javax.swing.JFrame {
12
13     private Socket socket;
14     private DataInputStream dataIn;
15     private DataOutputStream dataOut;
16     private boolean connected = false;
17     private boolean isPlaying = false;
18     private String selectedSong = null;
19     DefaultListModel<String> listModel;
20
21     public TCPAudioStreamingClient() {
22         initComponents();
23         DocumentListener validateInput = new DocumentListener() {
24
25             @Override
26             public void insertUpdate(DocumentEvent e) {
27                 validateTextField();
28             }
29
30             @Override
31             public void removeUpdate(DocumentEvent e) {
32                 validateTextField();
33             }
34
35             @Override
36             public void changedUpdate(DocumentEvent e) {
37                 validateTextField();
38             }
39
40             protected void validateTextField() {
41                 String inputServerName = tfdServerName.getText().trim();
42                 String inputPort = tfdPort.getText().trim();
43
44                 if (!inputServerName.equals("") && !inputPort.equals("")) {
45                     btnHandle.setEnabled(true);
46                 } else {
47                     btnHandle.setEnabled(false);
48                 }
49             }
50         };
51
52         tfdServerName.getDocument().addDocumentListener(validateInput);
53         tfdPort.getDocument().addDocumentListener(validateInput);
54         listModel = new DefaultListModel<>();
55         listSongsReceived.setModel(listModel);
56
57         lblNameSong.setVisible(false);
58         btnPrevious.setVisible(false);
59         btnPause.setVisible(false);
60         btnPlay.setVisible(false);
61         btnStop.setVisible(false);
62         btnNext.setVisible(false);
63     }

```

Khai báo biến:

- + Socket socket: Đối tượng socket để kết nối với server.
 - + DataInputStream dataIn: Đối tượng để đọc dữ liệu từ server.
 - + DataOutputStream dataOut: Đối tượng để gửi dữ liệu đến server.
 - + boolean connected: Biến trạng thái để kiểm tra xem client đang kết nối hay không.
 - + boolean isPlaying: Biến trạng thái để kiểm tra xem client đang phát nhạc hay không.
 - + String selectedSong: Biến lưu tên bài hát được chọn.
- DefaultListModel<String> listModel: Model để quản lý danh sách bài hát nhận được từ server.

Phương thức TCPAudioStreamingClient():

- + Khởi tạo giao diện và thêm các DocumentListener để kiểm tra và xác thực đầu vào từ người dùng, khởi tạo model danh sách bài hát.

```

218 private void btnHandleActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnHandleActionPerformed
219     if (!connected) {
220         startClient();
221     } else {
222         leaveAudioStreamRoom();
223     }
224 }
```

Phương thức btnHandleActionPerformed():

- + Xử lý sự kiện khi người dùng nhấn vào nút "Start/Leave".
- + Gọi startClient() nếu chưa kết nối, hoặc leaveAudioStreamRoom() nếu đã kết nối.

```

226 private void startClient() {
227     String SERVER = tfdServerName.getText().trim();
228     String inputPORT = tfdPort.getText().trim();
229
230     try {
231         int PORT = Integer.parseInt(inputPORT);
232
233         if (PORT < 1 || PORT > 65535) {
234             throw new IllegalArgumentException("Port number must be between 1 and 65535");
235         }
236
237         socket = new Socket(SERVER, PORT);
238         dataIn = new DataInputStream(socket.getInputStream());
239         dataOut = new DataOutputStream(socket.getOutputStream());
240
241         // Gửi yêu cầu lấy danh sách bài hát
242         dataOut.writeUTF("GET_SONG_LIST");
243
244         // Nhận danh sách bài hát từ server
245         String songListStr = dataIn.readUTF();
246         String[] songList = songListStr.split("\n");
247         for (String song : songList) {
248             listModel.addElement(song);
249         }
250
251         connected = true;
252         playSongThread();
253         toggleInput();
254
255     } catch (NumberFormatException ex) {
256         JOptionPane.showMessageDialog(null, "Malformed port: " + inputPORT, "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
257     } catch (IllegalArgumentException ex) {
258         JOptionPane.showMessageDialog(null, ex.getMessage(), "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
259     } catch (UnknownHostException ex) {
260         JOptionPane.showMessageDialog(null, "Unknown server: " + SERVER, "Unknown Host", JOptionPane.ERROR_MESSAGE);
261     } catch (IOException ex) {
262         JOptionPane.showMessageDialog(null, "No corresponding server found", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
263     }
264 }

```

Phương thức startClient():

- + Lấy thông tin máy chủ và cổng từ các trường nhập liệu.
- + Tạo Socket để kết nối với máy chủ.
- + Gửi yêu cầu "GET_SONG_LIST" để nhận danh sách bài hát từ máy chủ.
- + Nhận danh sách bài hát và cập nhật listModel.
- + Đánh dấu trạng thái connected là true và gọi playSongThread() để bắt đầu phát nhạc.
- + Gọi toggleInput() để cập nhật trạng thái giao diện.

```

266 private void playSongThread() {
267     // Tạo luồng để nhận dữ liệu và phát nhạc từ server
268     Thread audioStreamingThread = new Thread(() -> {
269         try {
270             // Tạo buffer để lưu dữ liệu âm thanh từ server
271             byte[] buffer = new byte[4096]; // hoặc kích thước buffer phù hợp với nhu cầu của bạn
272
273             // Mở SourceDataLine để phát âm thanh
274             AudioFormat audioFormat = new AudioFormat(AudioFormat.Encoding.PCM_SIGNED, 44100.0f, 16, 2, 4, 44100.0f, false);
275             DataLine.Info info = new DataLine.Info(SourceDataLine.class, audioFormat);
276             SourceDataLine sourceDataLine = (SourceDataLine) AudioSystem.getLine(info);
277             sourceDataLine.open(audioFormat);
278             sourceDataLine.start();
279
280             // Đọc dữ liệu từ server và ghi vào SourceDataLine để phát nhạc
281             while (connected) {
282                 int bytesRead = dataIn.read(buffer);
283                 if (bytesRead == -1) {
284                     break; // Kết thúc luồng nếu không còn dữ liệu nữa
285                 }
286                 sourceDataLine.write(buffer, 0, bytesRead); // Ghi dữ liệu vào SourceDataLine
287             }
288
289             // Khi kết thúc, đóng SourceDataLine
290             sourceDataLine.drain();
291             sourceDataLine.close();
292         } catch (IOException ex) {
293             // Xử lý các ngoại lệ
294         } catch (LineUnavailableException ex) {
295             Logger.getLogger(TCPAudioStreamingClient.class.getName()).log(Level.SEVERE, null, ex);
296         }
297     });
298     // Khởi chạy luồng mới để nhận dữ liệu và phát nhạc
299     audioStreamingThread.start();
300 }

```

Phương thức playSongThread():

- + Tạo một luồng để nhận dữ liệu âm thanh từ máy chủ và phát nhạc.
- + Sử dụng SourceDataLine để phát âm thanh từ dữ liệu nhận được.
- + Đọc dữ liệu từ máy chủ và ghi vào SourceDataLine để phát nhạc.

Đóng SourceDataLine khi kết thúc.

```

302 private void leaveAudioStreamRoom() {
303     connected = false;
304     toggleInput();
305     listModel.clear();
306     try {
307         // Gửi tin nhắn rời phòng chat tới server
308         dataOut.writeUTF("LEAVE_CHAT_ROOM");
309         // Dừng luồng nhận tin nhắn
310         JOptionPane.showMessageDialog(null, "You have successfully left the chat room.", "Successfully", JOptionPane.INFORMATION_MESSAGE);
311         socket.close(); // Đóng socket đến server
312     } catch (IOException e) {
313         e.printStackTrace();
314     }
315 }

```

Phương thức leaveAudioStreamRoom():

- + Gửi yêu cầu rời phòng stream đến server và ngắt kết nối.
- + Cập nhật giao diện người dùng sau khi ngắt kết nối.

```

317  private void formWindowClosing(java.awt.event.WindowEvent evt) { //GEN-FIRST:event_formWindowClosing
318      if (socket != null && !socket.isClosed()) {
319          leaveAudioStreamRoom();
320      }
321  }

```

Phương thức formWindowClosing():

- + Gọi khi cửa sổ đang được đóng.
- + Nếu máy khách đang kết nối, gọi leaveAudioStreamRoom() để đảm bảo kết nối được đóng trước khi đóng cửa sổ.

```

323  private void listSongsReceivedValueChanged(javax.swing.event.ListSelectionEvent evt) { //GEN-FIRST:event_listSongsReceivedValueChanged
324      if (!evt.getValueIsAdjusting()) {
325          String selection = listSongsReceived.getSelectedValue();
326          if (selection != null && !selection.isEmpty()) {
327              selectedSong = selection;
328              lblNameSong.setText(selectedSong);
329              isPlaying = true;
330              toggleAudio();
331          }
332          listSongsReceived.clearSelection();
333      }
334  }

```

Phương thức listSongsReceivedValueChanged():

- + Xử lý sự kiện khi người dùng chọn một bài hát từ danh sách.
- + Cập nhật selectedSong và hiển thị tên bài hát trên giao diện.
- + Gọi toggleAudio() để hiển thị các nút điều khiển âm thanh.

```

336  private void btnPlayActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnPlayActionPerformed
337      // Gửi yêu cầu lấy dữ liệu bài hát để phát trực tiếp
338      if (selectedSong != null) {
339          try {
340              dataOut.writeUTF("PLAY_SONG|" + selectedSong);
341              btnPlay.setEnabled(false);
342          } catch (IOException ex) {
343              Logger.getLogger(TCPAudioStreamingClient.class.getName()).log(Level.SEVERE, null, ex);
344          }
345      }
346  }

```

Phương thức btnPlayActionPerformed():

- + Xử lý sự kiện khi người dùng nhấn nút "Play".
- + Gửi yêu cầu "PLAY_SONG" cùng tên bài hát đến máy chủ để phát nhạc.

```

348 private void toggleAudio() {
349     lblNameSong.setVisible(isPlaying);
350     btnPrevious.setVisible(isPlaying);
351     btnPause.setVisible(isPlaying);
352     btnPlay.setVisible(isPlaying);
353     btnStop.setVisible(isPlaying);
354     btnNext.setVisible(isPlaying);
355 }
356
357 private void toggleInput() {
358     btnHandle.setText(connected ? "Leave" : "Start");
359     tfdServerName.setEnabled(!connected);
360     tfdPort.setEnabled(!connected);
361 }

```

Các phương thức hỗ trợ khác:

- + *toggleAudio()*: Hiển thị hoặc ẩn các nút điều khiển âm thanh dựa trên trạng thái phát nhạc.
- + *toggleInput()*: Cập nhật trạng thái các thành phần giao diện dựa trên trạng thái kết nối.

3.1.2.3 TCP – Chương trình File Transfer (Truyền file)

- **Server**

Tương tự với Chương trình Multi Chat Room, nhưng bổ sung thêm tính năng gửi kèm file.

```

110  private void startServer() {
111      ...
112
113      while (connected) {
114          try {
115              // Đọc loại dữ liệu đầu tiên từ client
116              String dataType = dataIn.readUTF();
117
118              // Kiểm tra loại dữ liệu để xác định xem client gửi tin nhắn hay file
119              if (dataType.equals("MESSAGE")) {
120                  // Đọc tin nhắn từ client
121                  String msgIn = dataIn.readUTF();
122
123                  if ("LEAVE_CHAT_ROOM".equals(msgIn)) {
124                      // Ngắt kết nối khi client thoát
125                      synchronized (connectedClients) {
126                          connectedClients.remove(client);
127                      }
128                      clientSocket.close();
129
130                      // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
131                      onlineUsersTextArea.append(client.getUsername() + " has disconnected from the chat room\n");
132
133                      // Thông báo client đã rời khỏi phòng cho các client khác
134                      broadcastMessage(client.getUsername(), "has left the chat room");
135                      break;
136                  }
137
138                  // Chuyển tiếp tin nhắn cho tất cả các client khác
139                  broadcastMessage(username, msgIn);
140              } else if (dataType.equals("FILE")) {
141                  // Đọc tên file từ client
142                  String fileName = dataIn.readUTF();
143                  // Đọc kích thước file từ client
144                  long fileSize = dataIn.readLong();
145
146                  // Nhận dữ liệu file từ client
147                  byte[] fileData = new byte[(int) fileSize];
148                  dataIn.readFully(fileData);
149
150                  // Chuyển tiếp file cho tất cả các client khác
151                  broadcastFile(username, fileName, fileData);
152              }
153          } catch (IOException e) {
154              // Xử lý lỗi khi giao tiếp với client
155              if (connected) {
156                  onlineUsersTextArea.append("An error occurred while receiving the message\n");
157              }
158          }
159      }
160  }
161
162  ...
163 }

```

Phương thức startServer():

- + Lấy số cổng từ trường nhập liệu và kiểm tra xem cổng có hợp lệ không (1-65535).
- + Tạo một ServerSocket để lắng nghe các kết nối đến từ khách hàng.
- + Tạo một luồng để chấp nhận các kết nối từ nhiều khách hàng đồng thời.
- + Tạo một luồng riêng cho mỗi khách hàng để xử lý yêu cầu.
- + Sử dụng DataInputStream và DataOutputStream để giao tiếp với khách hàng.
- + Đọc tên người dùng từ khách hàng và kiểm tra xem tên người dùng có tồn tại không.

- + Nếu tên người dùng tồn tại, gửi thông báo lỗi. Nếu không, chấp nhận và thêm vào danh sách khách hàng.
- + Gửi danh sách người dùng hiện tại cho khách hàng mới.
- + Thêm khách hàng mới vào danh sách và thông báo cho các khách hàng khác.
- + Đọc loại dữ liệu từ khách hàng (tin nhắn hoặc file).
- + Chuyển tiếp tin nhắn hoặc file cho tất cả khách hàng khác.
- + Xử lý việc khách hàng rời phòng chat.
- + Xử lý các lỗi xảy ra khi giao tiếp với khách hàng và cập nhật giao diện.

```

304  private void broadcastFile(String fileOwner, String fileName, byte[] fileData) {
305      synchronized (connectedClients) {
306          for (TCPClientInfo client : connectedClients) {
307              if (!client.getUsername().equals(fileOwner)) {
308                  try {
309                      // Gửi loại dữ liệu là FILE
310                      client.getDataOut().writeUTF("FILE");
311                      // Gửi tên file
312                      client.getDataOut().writeUTF(fileName);
313                      // Gửi kích thước file
314                      client.getDataOut().writeLong(fileData.length);
315                      // Gửi dữ liệu file
316                      client.getDataOut().write(fileData);
317                      // Gửi tên người gửi
318                      client.getDataOut().writeUTF(fileOwner);
319                  } catch (IOException e) {
320                      JOptionPane.showMessageDialog(null, "Error broadcasting file: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
321                  }
322              }
323          }
324      }
325  }

```

Phương thức broadcastFile():

- + Phương thức được đồng bộ hóa trên danh sách connectedClients để đảm bảo tính toàn vẹn dữ liệu khi truy cập từ nhiều luồng.
- + Duyệt qua tất cả các khách hàng trong danh sách connectedClients.
- + Bỏ qua khách hàng là chủ sở hữu file (người gửi file).
- + Gửi loại dữ liệu là "FILE".
- + Gửi tên file và kích thước file.
- + Gửi dữ liệu file.
- + Gửi tên người gửi file.
- + Nếu có lỗi khi gửi file, hiển thị thông báo lỗi qua JOptionPane.

- **Client**

Tương tự với Chương trình Multi Chat Room, nhưng bổ sung thêm tính năng gửi nhận và download file.

```

1 package main.TCP.FileTrans;
2
3 import java.io.*;
4 import java.net.*;
5 import java.nio.file.*;
6 import java.util.ArrayList;
7 import javax.swing.*;
8 import javax.swing.event.*;
9 import javax.swing.text.*;
10
11 public class TCPFileTransClient extends javax.swing.JFrame {
12
13     private Socket socket;
14     private DataInputStream dataIn;
15     private DataOutputStream dataOut;
16     private File selectedFile;
17     private boolean connected = false;
18     DefaultListModel<String> listModel;
19     private ArrayList<FileInfo> fileList = new ArrayList<>();

```

Khai báo biến:

- + socket, dataIn, dataOut: Các biến dùng để kết nối và trao đổi dữ liệu với máy chủ.
- + selectedFile: Tập tin được chọn để gửi.
- + connected: Biến trạng thái để kiểm tra xem có đang kết nối tới máy chủ hay không.
- + listModel: Mô hình danh sách để hiển thị các tập tin nhận được.
- + fileList: Danh sách lưu trữ thông tin các tập tin nhận được.

```

251 private void joinChatRoom() {
252     ...
253
254     // Tạo luồng để nhận tin nhắn từ server
255     connected = true;
256     Thread clientThread = new Thread(() -> {
257         try {
258             while (connected) {
259                 // Đọc loại dữ liệu từ máy chủ
260                 String dataType = dataIn.readUTF();
261
262                 // Nếu dữ liệu là tin nhắn
263                 if ("MESSAGE".equals(dataType)) {
264                     String msgIn = dataIn.readUTF();
265                     // Hiển thị tin nhắn
266                     appendToPane(chatDataPane, msgIn + "\n", StyleConstants.ALIGN_LEFT);
267                 } // Nếu dữ liệu là file
268                 else if ("FILE".equals(dataType)) {
269                     // Đọc tên file và kích thước file
270                     String fileName = dataIn.readUTF();
271                     long fileSize = dataIn.readLong();
272
273                     // Đọc dữ liệu file
274                     byte[] fileData = new byte[(int) fileSize];
275                     dataIn.readFully(fileData);
276
277                     // Đọc tên người gửi
278                     String fileOwner = dataIn.readUTF();
279
280                     fileList.add(new FileInfo(fileOwner, fileName, fileData));
281
282                     appendToPane(chatDataPane, "\n" + fileOwner + ": attached file - " + fileName + "\n", StyleConstants.ALIGN_LEFT);
283
284                     // Thêm tên file vào JList
285                     listModel.addElement(fileOwner + ": " + fileName);
286
287                 } else if ("SERVER_SHUTDOWN".equals(dataType)) {
288                     connected = false;
289                     appendToPane(chatDataPane, "Server has been shut down. You have been disconnected.\n", StyleConstants.ALIGN_LEFT);
290                     toggleInput(connected);
291                     socket.close();
292                     break;
293                 }
294             }
295         ...
296     })
297 }
```

Phương thức joinChatRoom():

- + Kết nối đến server với địa chỉ và cổng được cung cấp.
- + Đăng ký tên người dùng và xử lý phản hồi từ server.
- + Tạo một luồng để nhận và xử lý tin nhắn hoặc file từ server.

```

360 private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnSendActionPerformed
361     String inputMsg = msgArea.getText().trim();
362
363     try {
364         // Gửi tin nhắn
365         if (!inputMsg.isEmpty()) {
366             dataOut.writeUTF("MESSAGE"); // Gửi loại dữ liệu là MESSAGE
367             dataOut.writeUTF(inputMsg); // Gửi tin nhắn
368             appendToPane(chatDataPane, inputMsg + "\n", StyleConstants.ALIGN_RIGHT);
369         }
370
371         // Gửi file nếu có
372         if (selectedFile != null) {
373             // Gửi loại dữ liệu là FILE
374             dataOut.writeUTF("FILE");
375             // Gửi tên file
376             dataOut.writeUTF(selectedFile.getName());
377             // Gửi kích thước file
378             dataOut.writeLong(selectedFile.length());
379
380             // Mở FileInputStream để đọc dữ liệu từ file
381             try (FileInputStream fileInputStream = new FileInputStream(selectedFile)) {
382                 // Tạo một byte array để lưu trữ dữ liệu từ file
383                 byte[] buffer = new byte[8192];
384                 int bytesRead;
385                 while ((bytesRead = fileInputStream.read(buffer)) != -1) {
386                     // Gửi dữ liệu file
387                     dataOut.write(buffer, 0, bytesRead);
388                 }
389             }
390             appendToPane(chatDataPane, "attached file \'" + selectedFile.getName() + '\'' + " has been sent\n", StyleConstants.ALIGN_RIGHT);
391         }
392
393     } catch (IOException ex) {
394         // Xử lý ngoại lệ nếu có lỗi xảy ra
395         JOptionPane.showMessageDialog(null, "An error occurred while sending the message or file! Please try again.", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
396     }
397
398     // Xóa nội dung trong msgArea sau khi đã gửi xong
399     selectedFile = null;
400     btnCancelFile.setVisible(false);
401     lblFilename.setVisible(false);
402     msgArea.setText("");
403 }

```

Phương thức btnSendActionPerformed():

- + Gửi tin nhắn và file (nếu có) đến server.
- + Hiển thị tin nhắn đã gửi lên giao diện và xóa nội dung trong trường nhập liệu.

```

411 private void btnChooseFileMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_btnChooseFileMouseClicked
412     if (connected) {
413         JFileChooser fileChooser = new JFileChooser();
414         int returnValue = fileChooser.showOpenDialog(null);
415         if (returnValue == JFileChooser.APPROVE_OPTION) {
416             selectedFile = fileChooser.getSelectedFile();
417             String fileName = selectedFile.getName(); // Lấy tên của file
418             lblFilename.setText(fileName);
419             lblFilename.setVisible(true);
420             btnCancelFile.setVisible(true);
421         }
422     }
423 }

```

Phương thức btnChooseFileMouseClicked():

- + Mở hộp thoại chọn file, nếu có file được chọn, hiển thị tên file và nút hủy.

```

425 private void listFileReceivedValueChanged(javax.swing.event.ListSelectionEvent evt) { //GEN-FIRST:event_listFileReceivedValueChanged
426     if (!evt.getValueIsAdjusting()) {
427         int selectedIndex = listFileReceived.getSelectedIndex();
428         if (selectedIndex != -1) {
429             // Lấy dữ liệu file từ danh sách cục bộ tương ứng với vị trí được chọn
430             FileInfo fileInfo = fileList.get(selectedIndex);
431             // Hiển thị hộp thoại xác nhận
432             String msg = "Do you want to download the file: " + fileInfo.getFileName() + " from " + fileInfo.getFileOwner() + "?";
433             int option = JOptionPane.showConfirmDialog(null, msg, "Confirm Download", JOptionPane.YES_NO_OPTION);
434             if (option == JOptionPane.YES_OPTION) {
435                 // Nếu người dùng chọn Yes, thực hiện tải xuống
436                 downloadFile(fileInfo);
437             }
438         }
439         listFileReceived.clearSelection();
440     }
441 }

```

Phương thức listFileReceivedValueChanged():

- + Khi một file trong danh sách được chọn, hiển thị hộp thoại xác nhận tải xuống. Nếu người dùng chọn "Yes", sẽ tải file xuống.

```

443 private void btnCancelFileMouseClicked(java.awt.event.MouseEvent evt) { //GEN-FIRST:event_btnCancelFileMouseClicked
444     selectedFile = null;
445     btnCancelFile.setVisible(false);
446     lblFilename.setVisible(false);
447 }

```

Phương thức btnCancelFileMouseClicked(): Hủy file đã chọn, ẩn nhãn tên file và nút hủy.

```

449 private void downloadFile(FileInfo fileInfo) {
450     try {
451         // Tạo đường dẫn tới thư mục "Download"
452         Path downloadDir = Paths.get(System.getProperty("user.home"), "Downloads");
453
454         // Tạo đường dẫn tới file trong thư mục "Download"
455         Path filePath = downloadDir.resolve(fileInfo.getFileName());
456
457         // Ghi dữ liệu file vào file trong thư mục "Download"
458         Files.write(filePath, fileInfo.getFileData());
459
460         JOptionPane.showMessageDialog(null, "File saved to: " + filePath.toString(), "Successfully", JOptionPane.INFORMATION_MESSAGE);
461     } catch (IOException e) {
462         JOptionPane.showMessageDialog(null, "Fail to save file! Please try again.", "Fail", JOptionPane.ERROR_MESSAGE);
463     }
464 }

```

Phương thức downloadFile(FileInfo fileInfo):

- + Lưu dữ liệu file vào thư mục "Downloads" trên máy người dùng và hiển thị thông báo thành công hoặc thất bại.

- **FileInfo Class**

```
1 package main.TCP.FileTrans;
2
3 public class FileInfo {
4
5     private String fileOwner;
6     private String fileName;
7     private byte[] fileData;
8
9     public FileInfo(String fileOwner, String fileName, byte[] fileData) {
10         this.fileOwner = fileOwner;
11         this.fileName = fileName;
12         this.fileData = fileData;
13     }
14
15    public String getFileOwner() {
16        return fileOwner;
17    }
18
19    public String getFileName() {
20        return fileName;
21    }
22
23    public byte[] getFileData() {
24        return fileData;
25    }
26}
27
```

3.1.2.4 TCP – Chương trình File Encrypt AES (Mã hóa file)

- Server

```

1 package main.TCP.FileEncrypt;
2
3 import java.io.*;
4 import java.net.*;
5 import java.util.Base64;
6 import javax.crypto.*;
7 import javax.crypto.spec.SecretKeySpec;
8 import javax.swing.*;
9
10 public class TCPFileEncryptServer extends javax.swing.JFrame {
11
12     private ServerSocket serverSocket;
13     Socket clientSocket;
14     DataInputStream dataIn;
15     DataOutputStream dataOut;
16     private boolean connected = false;
17
18     public TCPFileEncryptServer() {
19         initComponents();
20     }

```

Khai báo biến:

- + serverSocket: Đối tượng ServerSocket để chấp nhận các kết nối từ các client.
- + clientSocket: Đối tượng Socket đại diện cho kết nối tới client.
- + dataIn: Đối tượng DataInputStream để đọc dữ liệu từ client.
- + dataOut: Đối tượng DataOutputStream để gửi dữ liệu đến client.
- + connected: Biến boolean để xác định trạng thái kết nối của server.

```

113  private void startServer() {
114      ...
115      connected = true;
116      Thread serverThread = new Thread(() -> {
117          while (!serverSocket.isClosed()) {
118              try {
119                  clientSocket = serverSocket.accept();
120
121                  Thread clientHandlerThread = new Thread(() -> {
122                      try {
123                          dataIn = new DataInputStream(clientSocket.getInputStream());
124                          dataOut = new DataOutputStream(clientSocket.getOutputStream());
125
126                          while (!clientSocket.isClosed()) {
127                              String msgIn = dataIn.readUTF();
128                              if ("ENCRYPT".equals(msgIn)) {
129                                  // Đọc secret key
130                                  String encodedKey = dataIn.readUTF();
131                                  byte[] decodedKey = Base64.getDecoder().decode(encodedKey);
132                                  SecretKeySpec secretKey = new SecretKeySpec(decodedKey, 0, decodedKey.length, "AES");
133
134                                  // Đọc tên file và kích thước file
135                                  String fileName = dataIn.readUTF();
136                                  long fileSize = dataIn.readLong();
137
138                                  // Đọc dữ liệu file
139                                  byte[] fileData = new byte[(int) fileSize];
140                                  dataIn.readFully(fileData);
141
142                                  File receivedFile = new File(fileName);
143                                  try (FileOutputStream fos = new FileOutputStream(receivedFile)) {
144                                      fos.write(fileData);
145                                      onlineUsersTextArea.append("File received: " + receivedFile.getName() + "\n");
146                                  }
147
148                                  // Thực hiện mã hóa
149                                  File encryptedFile = encryptFile(receivedFile, secretKey);
150                                  try (FileInputStream fis = new FileInputStream(encryptedFile)) {
151                                      // Gửi msgIn
152                                      dataOut.writeUTF("ENCRYPTED");
153                                      // Gửi tên file
154                                      dataOut.writeUTF(encryptedFile.getName());
155                                      // Gửi kích thước file
156                                      dataOut.writeLong(encryptedFile.length());
157
158                                      // Gửi data file mã hóa
159                                      byte[] buffer = new byte[8192];
160                                      int bytesRead;
161                                      while ((bytesRead = fis.read(buffer)) != -1) {
162                                          dataOut.write(buffer, 0, bytesRead);
163                                      }
164
165                                      onlineUsersTextArea.append("Encrypted file sent back to client.\n");
166                                  } else if ("LEAVE".equals(msgIn)) {
167                                      clientSocket.close();
168                                      onlineUsersTextArea.append("Client has disconnected to the encrypt room\n");
169                                  }
170                              }
171
172                      } catch (IOException e) {
173                          e.printStackTrace();
174                      }
175                  });
176
177                  onlineUsersTextArea.append("New client connected.\n");
178                  clientHandlerThread.start();
179
180              }
181          }
182      });
183
184      ...
185  }

```

Phương thức startServer():

- + Khởi tạo một server socket trên cổng được chỉ định.
- + Chấp nhận kết nối từ client và tạo một luồng xử lý riêng cho mỗi client.

- + Trong quá trình kết nối, đọc tin nhắn từ client. Nếu tin nhắn là "ENCRYPT", thực hiện việc nhận và mã hóa file, sau đó gửi file mã hóa lại cho client.
- + Nếu tin nhắn là "LEAVE", ngắt kết nối với client.

```

212 private static File encryptFile(File file, SecretKey secretKey) throws IOException {
213     File encryptedFile = new File("encrypted_" + file.getName());
214     try (FileInputStream fis = new FileInputStream(file); FileOutputStream fos = new FileOutputStream(encryptedFile)) {
215
216         Cipher cipher = Cipher.getInstance("AES");
217         cipher.init(Cipher.ENCRYPT_MODE, secretKey);
218
219         byte[] buffer = new byte[8192];
220         int bytesRead;
221         while ((bytesRead = fis.read(buffer)) != -1) {
222             byte[] output = cipher.update(buffer, 0, bytesRead);
223             if (output != null) {
224                 fos.write(output);
225             }
226         }
227         byte[] output = cipher.doFinal();
228         if (output != null) {
229             fos.write(output);
230         }
231     } catch (Exception e) {
232         e.printStackTrace();
233     }
234     return encryptedFile;
235 }
```

Phương thức encryptFile(File file, SecretKey secretKey):

- + Mã hóa một file bằng thuật toán AES sử dụng một secret key đã cho.
- + Đọc dữ liệu từ file, mã hóa và ghi dữ liệu đã mã hóa vào một file mới.

```

237 private void stopServer() {
238     try {
239         connected = false;
240
241         // Đóng ClientSocket
242         if (clientSocket != null && !clientSocket.isClosed()) {
243             // Gửi msgIn
244             dataOut.writeUTF("SERVER_SHUTDOWN");
245             dataIn.close();
246             clientSocket.close();
247             dataOut.close();
248         }
249
250         // Đóng ServerSocket
251         if (serverSocket != null && !serverSocket.isClosed()) {
252             serverSocket.close();
253         }
254
255         btnHandle.setText("Start");
256         textFieldPort.setEnabled(true);
257         onlineUsersTextArea.append("Server has been stopped.\n");
258
259     } catch (IOException e) {
260         JOptionPane.showMessageDialog(null, "Error while stopping the server: " + e.getMessage(), "Server Error", JOptionPane.ERROR_MESSAGE);
261     }
262 }
```

***Phương thức stopServer():* Dừng máy chủ và đóng tất cả các kết nối đang mở.**

- Client

```

1 package main.TCP.FileEncrypt;
2
3 import java.io.*;
4 import java.net.*;
5 import java.nio.file.Files;
6 import java.nio.file.Path;
7 import java.nio.file.Paths;
8 import java.security.NoSuchAlgorithmException;
9 import java.util.Base64;
10 import javax.crypto.*;
11 import javax.swing.*;
12 import javax.swing.event.*;
13 import main.TCP.FileTrans.FileInfo;
14
15 public class TCPFileEncryptClient extends javax.swing.JFrame {
16
17     private Socket socket;
18     private DataInputStream dataIn;
19     private DataOutputStream dataOut;
20     private File selectedFile;
21     private String secretKey;
22     private FileInfo receiveFile;
23     private boolean connected = false;
24
25     public TCPFileEncryptClient() {
26         initComponents();
27         DocumentListener validateInput = new DocumentListener() {
28
29             @Override
30             public void insertUpdate(DocumentEvent e) {
31                 validateTextField();
32             }
33
34             @Override
35             public void removeUpdate(DocumentEvent e) {
36                 validateTextField();
37             }
38
39             @Override
40             public void changedUpdate(DocumentEvent e) {
41                 validateTextField();
42             }
43
44             protected void validateTextField() {
45                 String inputServerName = tfdServerName.getText().trim();
46                 String inputPort = tfdPort.getText().trim();
47
48                 if (!inputServerName.equals("") && !inputPort.equals("")) {
49                     btnHandle.setEnabled(true);
50                 } else {
51                     btnHandle.setEnabled(false);
52                 }
53             }
54         };
55
56         tfdServerName.getDocument().addDocumentListener(validateInput);
57         tfdPort.getDocument().addDocumentListener(validateInput);
58         btnCancelFile.setVisible(false);
59     }

```

Khai báo biến:

- + socket: Đôi tượng Socket để kết nối tới máy chủ.
- + dataIn: Đôi tượng DataInputStream để nhận dữ liệu từ máy chủ.
- + dataOut: Đôi tượng DataOutputStream để gửi dữ liệu tới máy chủ.
- + selectedFile: Đôi tượng File đại diện cho tệp tin được chọn để gửi.
- + secretKey: Chuỗi lưu trữ khóa bí mật được sử dụng để mã hóa tệp tin.
- + receiveFile: Đôi tượng FileInfo đại diện cho tệp tin đã được mã hóa nhận được từ máy chủ.
- + connected: Biến boolean để xác định trạng thái kết nối với máy chủ.

Phương thức *TCPFileEncryptClient()*: Khởi tạo giao diện của client và thiết lập sự kiện cho các thành phần giao diện.

```

253  private void btnHandleActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnHandleActionPerformed
254      if (!connected) {
255          connectToServer();
256      } else {
257          leaveToServer();
258      }
259  }

```

Phương thức *btnHandleActionPerformed()*: Xử lý sự kiện khi người dùng nhấn nút "Connect/Leave".

```

261 private void connectToServer() {
262     String SERVER = tfdServerName.getText().trim();
263     String inputPORT = tfdPort.getText().trim();
264
265     try {
266         int PORT = Integer.parseInt(inputPORT);
267
268         if (PORT < 1 || PORT > 65535) {
269             throw new IllegalArgumentException("Port number must be between 1 and 65535");
270         }
271
272         socket = new Socket(SERVER, PORT);
273         dataIn = new DataInputStream(socket.getInputStream());
274         dataOut = new DataOutputStream(socket.getOutputStream());
275
276         // Tạo luồng để file mã hóa từ server
277         connected = true;
278         Thread clientThread = new Thread(() -> {
279             try {
280                 while (connected) {
281                     String msgIn = dataIn.readUTF();
282
283                     if ("ENCRYPTED".equals(msgIn)) {
284                         JOptionPane.showMessageDialog(null, "File sent to server for encryption..", "Successfully", JOptionPane.INFORMATION_MESSAGE);
285                         tbxResult.setText("File encryption is in progress ...");
286
287                     // Đọc tên file và kích thước file
288                     String fileName = dataIn.readUTF();
289                     long fileSize = dataIn.readLong();
290
291                     // Đọc dữ liệu file
292                     byte[] fileData = new byte[(int) fileSize];
293                     dataIn.readFully(fileData);
294
295                     // Giả lập độ trễ 4 giây
296                     try {
297                         Thread.sleep(4000);
298                     } catch (InterruptedException e) {
299                         Thread.currentThread().interrupt(); // Khôi phục trạng thái ngắt
300                     }
301
302                     receiveFile = new FileInfo(null, fileName, fileData);
303                     tbxResult.setText(fileName + " (click here to download)");
304                     tbxResult.setEnabled(true);
305
306                 } else if ("SERVER_SHUTDOWN".equals(msgIn)) {
307                     connected = false;
308                     JOptionPane.showMessageDialog(null, "Server has been shut down. You have been disconnected.\n", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
309                     toggleInput(connected);
310                     socket.close();
311                     break;
312                 }
313             }
314         } catch (IOException e) {
315             // Xử lý lỗi khi giao tiếp với server
316             if (connected) {
317                 JOptionPane.showMessageDialog(null, "An error occurred while receiving result! Please try connecting again..\n", "Server Shutdown", JOptionPane.ERROR_MESSAGE);
318                 leaveToServer();
319             }
320         });
321     });
322     clientThread.start();
323     toggleInput(connected);
324
325 } catch (NumberFormatException ex) {
326     JOptionPane.showMessageDialog(null, "Malformed port: " + inputPORT, "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
327 } catch (IllegalArgumentException ex) {
328     JOptionPane.showMessageDialog(null, ex.getMessage(), "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
329 } catch (UnknownHostException ex) {
330     JOptionPane.showMessageDialog(null, "Unknown server: " + SERVER, "Unknown Host", JOptionPane.ERROR_MESSAGE);
331 } catch (IOException ex) {
332     JOptionPane.showMessageDialog(null, "No corresponding server found", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
333 }
334 }
335 }

```

Phương thức connectToServer():

- + Kết nối đến server và khởi tạo luồng để nhận dữ liệu từ server.
- + Xử lý các ngoại lệ và hiển thị thông báo tương ứng.

```

382 private void btnEncryptActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnEncryptActionPerformed
383     // Gửi request yêu cầu mã hóa đến server
384     if (selectedFile != null && secretKey != null) {
385         try {
386             // Gửi msgIn
387             dataOut.writeUTF("ENCRYPT");
388             // Gửi secret key
389             dataOut.writeUTF(secretKey);
390             // Gửi tên file
391             dataOut.writeUTF(selectedFile.getName());
392             // Gửi kích thước file
393             dataOut.writeLong(selectedFile.length());
394
395             // Mở FileInputStream để đọc dữ liệu từ file
396             try (FileInputStream fileInputStream = new FileInputStream(selectedFile)) {
397                 // Tạo một byte array để lưu trữ dữ liệu từ file
398                 byte[] buffer = new byte[8192];
399                 int bytesRead;
400                 while ((bytesRead = fileInputStream.read(buffer)) != -1) {
401                     // Gửi dữ liệu file
402                     dataOut.write(buffer, 0, bytesRead);
403                 }
404             } catch (IOException ex) {
405                 JOptionPane.showMessageDialog(null, "An error occurred while sending file! Please try again.", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
406             }
407         } else {
408             JOptionPane.showMessageDialog(null, "No file selected or secret key generated!", "Error", JOptionPane.ERROR_MESSAGE);
409         }
410     }
411 }

```

Phương thức btnEncryptActionPerformed(ActionEvent evt):

- + Gửi yêu cầu mã hóa file đến server cùng với secret key đã tạo và dữ liệu của file.

```

413 private void btnSecretKeyActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnSecretKeyActionPerformed
414     try {
415         secretKey = generateSecretKey();
416         txbSecretKey.setText(secretKey);
417         if (selectedFile != null && secretKey != null) {
418             btnEncrypt.setEnabled(true);
419         }
420     } catch (NoSuchAlgorithmException ex) {
421         JOptionPane.showMessageDialog(null, "An error occurred while generating secret key! Please try again.", "Error", JOptionPane.ERROR_MESSAGE);
422     }
423 }

```

Phương thức btnSecretKeyActionPerformed(ActionEvent evt):

- + Tạo secret key mới sử dụng thuật toán AES và mã hóa thành chuỗi Base64 để gửi đến server.

```

425 private void tbxResultMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_tbxResultMouseClicked
426     if (receiveFile != null) {
427         downloadFile(receiveFile);
428         resetInput();
429     }
430 }//GEN-LAST:event_tbxResultMouseClicked
431
432 private void downloadFile(FileInfo fileInfo) {
433     try {
434         // Tạo đường dẫn tới thư mục "Download"
435         Path downloadDir = Paths.get(System.getProperty("user.home"), "Downloads");
436
437         // Tạo đường dẫn tới file trong thư mục "Download"
438         Path filePath = downloadDir.resolve(fileInfo.getFileName());
439
440         // Ghi dữ liệu file vào file trong thư mục "Download"
441         Files.write(filePath, fileInfo.getFileData());
442
443         JOptionPane.showMessageDialog(null, "File saved to: " + filePath.toString(), "Successfully", JOptionPane.INFORMATION_MESSAGE);
444     } catch (IOException e) {
445         JOptionPane.showMessageDialog(null, "Fail to save file! Please try again.", "Fail", JOptionPane.ERROR_MESSAGE);
446     }
447 }
448
449 private String generateSecretKey() throws NoSuchAlgorithmException {
450     KeyGenerator keyGen = KeyGenerator.getInstance("AES");
451     keyGen.init(256);
452     SecretKey secretKey = keyGen.generateKey();
453     String encodedKey = Base64.getEncoder().encodeToString(secretKey.getEncoded());
454     return encodedKey;
455 }

```

Phương thức *tbxResultMouseClicked(MouseEvent evt)*: Nếu có file đã mã hóa nhận được từ server, thực hiện việc tải file đó về máy và hiển thị thông báo kết quả.

Phương thức *downloadFile(FileInfo fileInfo)*: Tải file đã mã hóa về máy tính và lưu vào thư mục "Downloads".

Phương thức *generateSecretKey()*: Tạo một secret key mới sử dụng thuật toán AES và trả về chuỗi Base64 biểu diễn key đó.

3.1.2.5 TCP – Chương trình File Decrypt AES (Giải mã file)

- **Server**

Tương tự với chương trình File Encrypt AES Server

```
115 private void startServer() {
116     ...
117     Thread serverThread = new Thread(() -> {
118         while (!serverSocket.isClosed()) {
119             try {
120                 clientSocket = serverSocket.accept();
121
122                 Thread clientHandlerThread = new Thread(() -> {
123                     try {
124                         dataIn = new DataInputStream(clientSocket.getInputStream());
125                         dataOut = new DataOutputStream(clientSocket.getOutputStream());
126
127                         while (!clientSocket.isClosed()) {
128                             String msgIn = dataIn.readUTF();
129                             if ("DECRYPT".equals(msgIn)) {
130                                 // Đọc secret key
131                                 String encodedKey = dataIn.readUTF();
132
133                                 // Kiểm tra xem secret key có hợp lệ không
134                                 if (!isValidSecretKey(encodedKey)) {
135                                     // Nếu secret key không hợp lệ, gửi thông báo về client và không tiếp tục quá trình giải mã
136                                     dataOut.writeUTF("INVALID_KEY");
137                                     continue; // Chuyển sang vòng lặp tiếp theo
138                                 }
139
140                                 byte[] decodedKey = Base64.getDecoder().decode(encodedKey);
141                                 SecretKey secretKey = new SecretKeySpec(decodedKey, 0, decodedKey.length, "AES");
142
143                                 // Đọc tên file và kích thước file
144                                 String fileName = dataIn.readUTF();
145                                 long fileSize = dataIn.readLong();
146
147                                 // Đọc dữ liệu file
148                                 byte[] fileData = new byte[(int) fileSize];
149                                 dataIn.readFully(fileData);
150
151                                 // Giải mã dữ liệu
152                                 byte[] decryptedData = decryptData(fileData, secretKey);
153
154                                 // Tạo file mới để lưu dữ liệu giải mã
155                                 File decryptedFile = new File(fileName.replace("encrypted", "decrypted"));
156                                 try (FileOutputStream fos = new FileOutputStream(decryptedFile)) {
157                                     fos.write(decryptedData);
158                                 }
159
160                                 // Gửi tên file và kích thước file đã giải mã
161                                 dataOut.writeUTF("DECRYPTED");
162                                 dataOut.writeUTF(decryptedFile.getName());
163                                 dataOut.writeLong(decryptedFile.length());
164
165                                 // Gửi dữ liệu file đã giải mã
166                                 try (FileInputStream fis = new FileInputStream(decryptedFile)) {
167                                     byte[] buffer = new byte[8192];
168                                     int bytesRead;
169                                     while ((bytesRead = fis.read(buffer)) != -1) {
170                                         dataOut.write(buffer, 0, bytesRead);
171                                     }
172                                 }
173
174                                 onlineUsersTextArea.append("Decrypted file sent back to client.\n");
175                             } else if ("LEAVE".equals(msgIn)) {
176                                 clientSocket.close();
177                                 onlineUsersTextArea.append("Client has disconnected from the decrypt room\n");
178                             }
179                         }
180
181                     } catch (IOException e) {
182                         e.printStackTrace();
183                     }
184                 });
185
186                 onlineUsersTextArea.append("New client connected.\n");
187                 clientHandlerThread.start();
188             } catch (IOException e) {
189                 if (!serverSocket.isClosed()) {
190                     JOptionPane.showMessageDialog(null, "Server connection error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
191                 }
192             }
193         }
194     });
195
196 ...
197 }
```

Phương thức startServer():

- + Khởi tạo server socket và lắng nghe các kết nối từ client.

- + Xử lý yêu cầu giải mã file từ client bằng cách đọc secret key, tên file và dữ liệu file, sau đó thực hiện giải mã và gửi file đã giải mã về cho client.

```

222  private static byte[] decryptData(byte[] encryptedData, SecretKey secretKey) {
223      try {
224          Cipher cipher = Cipher.getInstance("AES");
225          cipher.init(Cipher.DECRYPT_MODE, secretKey);
226          return cipher.doFinal(encryptedData);
227      } catch (InvalidKeyException | NoSuchAlgorithmException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e) {
228          e.printStackTrace();
229          return null;
230      }
231  }

```

Phương thức *decryptData(byte[] encryptedData, SecretKey secretKey)*:

- + Sử dụng secret key để giải mã dữ liệu sử dụng thuật toán AES và trả về dữ liệu đã giải mã.

```

233  public static boolean isValidSecretKey(String encodedKey) {
234      // Kiểm tra độ dài của secret key (ở dạng base64)
235      if (encodedKey.length() != 24 && encodedKey.length() != 32 && encodedKey.length() != 44) {
236          return false;
237      }
238
239      // Kiểm tra xem secret key có đúng định dạng base64 không
240      try {
241          byte[] decodedKey = Base64.getDecoder().decode(encodedKey);
242          // Nếu không có ngoại lệ nào được ném, secret key là hợp lệ
243          return true;
244      } catch (IllegalArgumentException e) {
245          // Nếu có ngoại lệ IllegalArgumentException, secret key không hợp lệ
246          return false;
247      }
248  }

```

Phương thức *isValidSecretKey(String encodedKey)*:

- + Kiểm tra tính hợp lệ của secret key bằng cách kiểm tra độ dài của chuỗi encoded key (ở dạng Base64) và xem nó có thể giải mã được hay không.

• **Client**

Tương tự với chương trình File Encrypt AES Client

```

272 private void connectToServer() {
273     ...
274     Thread clientThread = new Thread(() -> {
275         try {
276             while (connected) {
277                 String msgIn = dataIn.readUTF();
278
279                 if ("DECRYPTED".equals(msgIn)) {
280                     JOptionPane.showMessageDialog(null, "File sent to server for decryption..", "Successfully", JOptionPane.INFORMATION_MESSAGE);
281                     tfdResult.setText("File decryption is in progress ...");
282
283                     // Đọc tên file và kích thước file
284                     String fileName = dataIn.readUTF();
285                     long fileSize = dataIn.readLong();
286
287                     // Đọc dữ liệu file
288                     byte[] fileData = new byte[(int) fileSize];
289                     dataIn.readFully(fileData);
290
291                     // Giả lập độ trễ 4 giây
292                     try {
293                         Thread.sleep(4000);
294                     } catch (InterruptedException e) {
295                         Thread.currentThread().interrupt(); // Khôi phục trạng thái ngắt
296                     }
297
298                     receiveFile = new FileInfo(null, fileName, fileData);
299                     tfdResult.setText(fileName + " (click here to download)");
300                     tfdResult.setEnabled(true);
301
302                 } else if ("INVALID_KEY".equals(msgIn)) {
303                     JOptionPane.showMessageDialog(null, "Invalid secret key! Please try again.", "Error", JOptionPane.ERROR_MESSAGE);
304
305                 } else if ("SERVER_SHUTDOWN".equals(msgIn)) {
306                     connected = false;
307                     JOptionPane.showMessageDialog(null, "Server has been shut down. You have been disconnected.\n", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
308                     toggleInput(connected);
309                     socket.close();
310                     break;
311                 }
312             }
313         } catch (IOException e) {
314             // Xử lý lỗi khi giao tiếp với server
315             if (connected) {
316                 JOptionPane.showMessageDialog(null, "An error occurred while receiving result! Please try connecting again..\n", "Server Shutdown", JOptionPane.ERROR_MESSAGE);
317                 leaveToServer();
318             }
319         }
320     });
321
322     ...
323 }

```

Phương thức connectToServer():

- + Thiết lập kết nối với server thông qua tên máy chủ và cổng.
- + Tạo luồng để nhận dữ liệu từ server, bao gồm cả thông báo về việc giải mã file và các thông báo khác từ server.

```

366 private void btnSelectFileActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnSelectFileActionPerformed
367     if (connected) {
368         JFileChooser fileChooser = new JFileChooser();
369         int option = fileChooser.showOpenDialog(null);
370         if (option == JFileChooser.APPROVE_OPTION) {
371             selectedFile = fileChooser.getSelectedFile();
372             String fileName = selectedFile.getName(); // Lấy tên của file
373             lblFilename.setText(fileName);
374             btnCancelFile.setVisible(true);
375             if (selectedFile != null && !tfdSecretKey.getText().trim().isEmpty()) {
376                 btnDecrypt.setEnabled(true);
377             }
378         }
379     }
380 }

```

Phương thức btnSelectFileActionPerformed(): Cho phép người dùng chọn một file để giải mã.

```

395 private void btnDecryptActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDecryptActionPerformed
396     // Gửi request yêu cầu giải mã đến server
397     String inputSecretKey = tfdSecretKey.getText().trim();
398     if (selectedfile != null && !inputSecretKey.isEmpty()) {
399
400         try {
401             // Gửi msgIn
402             dataOut.writeUTF("DECRYPT");
403             // Gửi secret key
404             dataOut.writeUTF(inputSecretKey);
405             // Gửi tên file
406             dataOut.writeUTF(selectedFile.getName());
407             // Gửi kích thước file
408             dataOut.writeLong(selectedFile.length());
409
410             // Mở FileInputStream để đọc dữ liệu từ file
411             try (FileInputStream fileInputStream = new FileInputStream(selectedFile)) {
412                 // Tạo một byte array để lưu trữ dữ liệu từ file
413                 byte[] buffer = new byte[8192];
414                 int bytesRead;
415                 while ((bytesRead = fileInputStream.read(buffer)) != -1) {
416                     // Gửi dữ liệu file
417                     dataOut.write(buffer, 0, bytesRead);
418                 }
419             }
420         } catch (IOException ex) {
421             JOptionPane.showMessageDialog(null, "An error occurred while sending file! Please try again.", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
422         }
423     } else {
424         JOptionPane.showMessageDialog(null, "No file selected or secret key generated!", "Error", JOptionPane.ERROR_MESSAGE);
425     }
426 }

```

Phương thức btnDecryptActionPerformed():

- + Gửi yêu cầu giải mã file đến server bao gồm tên file, kích thước và dữ liệu file cùng với secret key đã nhập.

```

388 private void btnCancelFileMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event.btnCancelFileMouseClicked
389     selectedFile = null;
390     btnCancelFile.setVisible(false);
391     lblFilename.setText("There is no file selected ...");
392     btnDecrypt.setEnabled(false);
393 }

```

```

428 private void tfdResultMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_tfdResultMouseClicked
429     if (receiveFile != null) {
430         downloadFile(receiveFile);
431         resetInput();
432     }
433 }//GEN-LAST:event_tfdResultMouseClicked
434
435 private void downloadFile( FileInfo fileInfo ) {
436     try {
437         // Tạo đường dẫn tới thư mục "Download"
438         Path downloadDir = Paths.get(System.getProperty("user.home"), "Downloads");
439
440         // Tạo đường dẫn tới file trong thư mục "Download"
441         Path filePath = downloadDir.resolve(fileInfo.getFileName());
442
443         // Ghi dữ liệu file vào file trong thư mục "Download"
444         Files.write(filePath, fileInfo.getFileData());
445
446         JOptionPane.showMessageDialog(null, "File saved to: " + filePath.toString(), "Successfully", JOptionPane.INFORMATION_MESSAGE);
447     } catch (IOException e) {
448         JOptionPane.showMessageDialog(null, "Fail to save file! Please try again.", "Fail", JOptionPane.ERROR_MESSAGE);
449     }
450 }

```

Phương thức btnCancelFileMouseClicked(): Hủy bỏ việc chọn file và cập nhật giao diện người dùng.

Phương thức *tfdResultMouseClicked()*: Khi người dùng nhấp chuột vào kết quả giải mã, file đã giải mã sẽ được tải xuống và lưu vào thư mục "Downloads".

Phương thức *downloadFile(FileInfo fileInfo)*: Lưu file đã giải mã vào thư mục "Downloads" trên máy tính của người dùng.

3.1.2.6 TCP – Chương trình Text Hashing (Mã hóa băm văn bản)

- **Server**

```

1 package main.TCP.TextHashing;
2
3 import java.io.*;
4 import java.net.*;
5 import java.security.MessageDigest;
6 import java.security.NoSuchAlgorithmException;
7 import java.util.ArrayList;
8 import javax.swing.*;
9
10 public class TCPTextHashingServer extends javax.swing.JFrame {
11
12     private ServerSocket serverSocket;
13     Socket clientSocket;
14     DataInputStream dataIn;
15     DataOutputStream dataOut;
16     private boolean connected = false;
17     private static final ArrayList<String> supportedAlgorithms = new ArrayList<>();
18
19     public TCPTextHashingServer() {
20         initComponents();
21         supportedAlgorithms.add("MD2");
22         supportedAlgorithms.add("MD5");
23         supportedAlgorithms.add("SHA-1");
24         supportedAlgorithms.add("SHA-224");
25         supportedAlgorithms.add("SHA-256");
26         supportedAlgorithms.add("SHA-384");
27         supportedAlgorithms.add("SHA-512");
28         supportedAlgorithms.add("SHA-512/224");
29         supportedAlgorithms.add("SHA-512/256");
30         supportedAlgorithms.add("SHA3-224");
31         supportedAlgorithms.add("SHA3-256");
32         supportedAlgorithms.add("SHA3-384");
33         supportedAlgorithms.add("SHA3-512");
34     }

```

Constructor *TCPTextHashingServer()*:

- + Trong constructor này, danh sách supportedAlgorithms được khởi tạo với các thuật toán băm được chọn sẵn như MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, và SHA3-512.

```

127  private void startServer() {
128      ...
129      Thread serverThread = new Thread(() -> {
130          while (!serverSocket.isClosed()) {
131              try {
132                  clientSocket = serverSocket.accept();
133
134                  Thread clientHandlerThread = new Thread(() -> {
135                      try {
136                          dataIn = new DataInputStream(clientSocket.getInputStream());
137                          dataOut = new DataOutputStream(clientSocket.getOutputStream());
138
139                          dataOut.writeUTF("TYPES");
140                          dataOut.writeInt(supportedAlgorithms.size());
141                          for (String algorithm : supportedAlgorithms) {
142                              dataOut.writeUTF(algorithm);
143                          }
144
145                          while (!clientSocket.isClosed()) {
146                              String msgIn = dataIn.readUTF();
147
148                              if ("HASH".equals(msgIn)) {
149                                  // Đọc loại mã hóa hash được chọn từ client
150                                  String hashAlgorithm = dataIn.readUTF();
151
152                                  // Đọc dữ liệu cần hash từ client
153                                  String text = dataIn.readUTF();
154
155                                  // Tính toán hash cho dữ liệu văn bản
156                                  String hashedText = hashText(text, hashAlgorithm);
157
158                                  // Gửi kết quả hash về client
159                                  dataOut.writeUTF("HASHERD");
160                                  dataOut.writeUTF(hashedText);
161
162                                  onlineUsersTextArea.append("Client request hash using " + hashAlgorithm + " algorithm.\n");
163                                  onlineUsersTextArea.append("Hashed text sent back to client.\n");
164
165                              } else if ("LEAVE".equals(msgIn)) {
166                                  clientSocket.close();
167                                  onlineUsersTextArea.append("Client has disconnected to the decrypt room\n");
168                              }
169                          }
170
171                      } catch (IOException e) {
172                          e.printStackTrace();
173                      }
174                  });
175
176                  onlineUsersTextArea.append("New client connected.\n");
177                  clientHandlerThread.start();
178              } catch (IOException e) {
179                  if (!serverSocket.isClosed()) {
180                      JOptionPane.showMessageDialog(null, "Server connection error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
181                  }
182              }
183          }
184      });
185
186      ...
187  }

```

Phương thức startServer():

- + Bắt đầu server trên cổng được chỉ định.
- + Gửi danh sách các thuật toán băm được hỗ trợ đến client khi client kết nối.

```

209  public static String hashText(String text, String algorithm) {
210      try {
211          // Tạo đối tượng MessageDigest với thuật toán được chỉ định
212          MessageDigest digest = MessageDigest.getInstance(algorithm);
213
214          // Cập nhật đối tượng MessageDigest với dữ liệu văn bản đầu vào
215          digest.update(text.getBytes());
216
217          // Lấy giá trị hash dưới dạng mảng byte
218          byte[] hashedBytes = digest.digest();
219
220          // Chuyển đổi mảng byte thành chuỗi hex
221          StringBuilder hexString = new StringBuilder();
222          for (byte hashedByte : hashedBytes) {
223              String hex = Integer.toHexString(0xff & hashedByte);
224              if (hex.length() == 1) {
225                  hexString.append('0');
226              }
227              hexString.append(hex);
228          }
229          return hexString.toString();
230      } catch (NoSuchAlgorithmException e) {
231          // Xử lý ngoại lệ nếu thuật toán không tồn tại
232          e.printStackTrace();
233          return null;
234      }
235  }

```

Phương thức hashText(String text, String algorithm):

- + Tính toán giá trị băm cho dữ liệu văn bản sử dụng thuật toán băm được chỉ định.
- + Phương thức này sử dụng MessageDigest để thực hiện băm và trả về giá trị băm dưới dạng chuỗi hex.

- **Client**

```

220  private void connectToServer() {
221      ...
222      // Tạo luồng để file mã hóa từ server
223      connected = true;
224      Thread clientThread = new Thread(() -> {
225          try {
226              String msgIn = dataIn.readUTF();
227
228              if ("TYPE".equals(msgIn)) {
229                  // Đọc số lượng thuật toán
230                  int numOfAlgorithms = dataIn.readInt();
231
232                  // Đọc và hiển thị danh sách các thuật toán
233                  ArrayList<String> supportedAlgorithms = new ArrayList<>();
234                  for (int i = 0; i < numOfAlgorithms; i++) {
235                      String algorithm = dataIn.readUTF();
236                      supportedAlgorithms.add(algorithm);
237                  }
238
239                  // Hiển thị danh sách thuật toán cho người dùng (cập nhật giao diện người dùng)
240                  DefaultComboBoxModel<String> model = new DefaultComboBoxModel<>(supportedAlgorithms.toArray(new String[0]));
241                  cbxAlgorithm.setModel(model);
242                  cbxAlgorithm.setEnabled(true);
243                  btnHash.setEnabled(true);
244              }
245
246              while (connected) {
247                  msgIn = dataIn.readUTF();
248
249                  if ("HASHED".equals(msgIn)) {
250                      // Đọc kết quả hash
251                      String hashedText = dataIn.readUTF();
252                      taaResult.setText("Text hashing is in progress ...");
253
254                      // Giả lập độ trễ 4 giây
255                      try {
256                          Thread.sleep(4000);
257                      } catch (InterruptedException e) {
258                          Thread.currentThread().interrupt(); // Khôi phục trạng thái ngắt
259                      }
260
261                      taaResult.setText(hashedText);
262
263                  } else if ("SERVER_SHUTDOWN".equals(msgIn)) {
264                      connected = false;
265                      JOptionPane.showMessageDialog(null, "Server has been shut down. You have been disconnected.\n", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
266                      toggleInput(connected);
267                      resetInput();
268                      socket.close();
269                      break;
270                  }
271
272              } catch (IOException e) {
273                  // Xử lý lỗi khi giao tiếp với server
274                  if (connected) {
275                      JOptionPane.showMessageDialog(null, "An error occurred while receiving result! Please try connecting again..\n", "Server Shutdown", JOptionPane.ERROR_MESSAGE);
276                      leaveToServer();
277                  }
278              });
279      });
280  ...
281 }
282 }
```

Phương thức connectToServer():

- + Kết nối đến server trên cổng được chỉ định.
- + Gửi yêu cầu lấy danh sách các thuật toán băm được hỗ trợ đến server khi kết nối thành công.
- + Nhận danh sách các thuật toán từ server và cập nhật giao diện người dùng.

```

329  private void btnHashActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnHashActionPerformed
330      // Lấy nội dung văn bản cần hash từ TextArea
331      String content = taaInput.getText().trim();
332      // Lấy thuật toán hash được chọn từ JComboBox
333      String selectedAlgorithm = (String) cbxAlgorithm.getSelectedItem();
334
335      if (!content.isEmpty() && selectedAlgorithm != null) {
336          try {
337              // Gửi lệnh HASH đến server
338              dataOut.writeUTF("HASH");
339              // Gửi thuật toán hash được chọn đến server
340              dataOut.writeUTF(selectedAlgorithm);
341              // Gửi nội dung cần hash đến server
342              dataOut.writeUTF(content);
343          } catch (IOException ex) {
344              JOptionPane.showMessageDialog(null, "An error occurred while sending the hash request! Please try again.", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
345          }
346      } else {
347          JOptionPane.showMessageDialog(null, "No text to hash or no algorithm selected!", "Error", JOptionPane.ERROR_MESSAGE);
348      }
349  }
```

Phương thức *btnHashActionPerformed()*:

- + Lấy nội dung văn bản từ JTextArea và thuật toán băm từ JComboBox.
- + Gửi yêu cầu băm văn bản đến server với nội dung và thuật toán đã chọn.

3.1.2.7 UDP – Chương trình Multi Chat Room (Chat nhiều người)

- **Server**

```

1 package main.UDP.ChatRoom;
2
3 import java.io.*;
4 import java.net.*;
5 import java.util.*;
6 import javax.swing.*;
7
8 public class UDPChatServer extends javax.swing.JFrame {
9
10    private DatagramSocket serverSocket;
11    private boolean connected = false;
12    private final Set<UDPClientInfo> connectedClients = new HashSet<>();
13
14    public UDPChatServer() {
15        initComponents();
16    }

```

Khai báo biến:

- + serverSocket: Một DatagramSocket để nhận và gửi dữ liệu qua UDP.
- + connected: Biến đánh dấu trạng thái kết nối của máy chủ.
- + connectedClients: Một tập hợp các UDPClientInfo đại diện cho các client đã kết nối.

```

101   private void btnHandleActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnHandleActionPerformed
102       if (!connected) {
103           startServer();
104       } else {
105           stopServer();
106       }
107   }

```

Phương thức *btnHandleActionPerformed()*: Bắt đầu hoặc dừng máy chủ dựa trên trạng thái kết nối hiện tại.


```

109 private void startServer() {
110     String port = textFieldPort.getText().trim();
111
112     try {
113         int Iport = Integer.parseInt(port);
114
115         if (Iport < 1 || Iport > 65535) {
116             throw new IllegalArgumentException("Port number must be between 1 and 65535");
117         }
118
119         serverSocket = new DatagramSocket(Iport);
120         byte[] buf = new byte[256];
121         DatagramPacket packetIn = new DatagramPacket(buf, buf.length);
122
123         connected = true;
124         Thread serverThread = new Thread(() -> {
125             try {
126                 while (connected) {
127                     serverSocket.receive(packetIn);
128                     String msgIn = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
129                     InetAddress clientAddress = packetIn.getAddress();
130                     int clientPort = packetIn.getPort();
131                     String clientUsername = findUsername(clientAddress, clientPort);
132
133                     if (msgIn.startsWith("[JOIN]")) {
134                         String username = msgIn.substring(6);
135
136                         boolean usernameExists = connectedClients.stream()
137                             .anyMatch(client -> client.getUsername().equals(username));
138
139                         if (usernameExists) {
140                             sendMessage("USERNAME_EXISTS", clientAddress, clientPort);
141                         } else {
142                             sendMessage("USERNAME_ACCEPTED", clientAddress, clientPort);
143
144                             // Gửi danh sách các tên người dùng hiện tại cho client mới
145                             StringBuilder usersList = new StringBuilder();
146                             if (!connectedClients.isEmpty()) {
147                                 synchronized (connectedClients) {
148                                     for (UDPClientInfo client : connectedClients) {
149                                         usersList.append(client.getUsername()).append(", ");
150                                     }
151                                 }
152                                 usersList.setLength(usersList.length() - 2);
153                                 usersList.append("& You are already in the chat room");
154                             } else {
155                                 usersList.append("You are the only one in the chat room");
156                             }
157
158                             sendMessage(usersList.toString(), clientAddress, clientPort);
159
160                             // Tạo một ClientInfo mới và thêm vào danh sách connectedClients
161                             UDPClientInfo client = new UDPClientInfo(username, clientAddress, clientPort);
162                             connectedClients.add(client);
163
164                             // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
165                             onlineUsersTextArea.append(username + ":" + clientAddress + " at port " + clientPort + " has connected to the chat room\n");
166
167                             // Thông báo client mới đã tham gia cho các client trước đó
168                             broadcastMessage(username, "has joined the chat room");
169
170                         }
171                     } else if (msgIn.startsWith("[CHAT]")) {
172                         String content = msgIn.substring(6);
173
174                         if ("LEAVE_CHAT_ROOM".equals(content)) {
175                             // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
176                             onlineUsersTextArea.append(clientUsername + ":" + clientAddress + " at port " + clientPort + " has disconnected from the chat room\n");
177
178                             // Thông báo client đã rời khỏi phòng cho các client khác
179                             broadcastMessage(clientUsername, "has left the chat room");
180
181                             // Ngắt kết nối khi client thoát
182                             synchronized (connectedClients) {
183                                 connectedClients.removeIf(client -> client.getAddress().equals(clientAddress) && client.getPort() == clientPort);
184                             }
185
186                             continue;
187                         }
188
189                         broadcastMessage(clientUsername, content);
190                     }
191                 }
192             } catch (IOException ex) {
193                 if (connected) {
194                     JOptionPane.showMessageDialog(null, ex.getMessage(), "IOException", JOptionPane.ERROR_MESSAGE);
195                 }
196             }
197         });
198         serverThread.start();
199
200         btnHandle.setText("Stop");
201         textFieldPort.setEnabled(false);
202         onlineUsersTextArea.setText("Server was started ...\\n\\n");
203
204     } catch (NumberFormatException ex) {
205         JOptionPane.showMessageDialog(null, "Malformed port: " + port, "Invalid Port Number", JOptionPane.WARNING_MESSAGE);
206     } catch (IllegalArgumentException ex) {
207         JOptionPane.showMessageDialog(null, ex.getMessage(), "Invalid Port Number", JOptionPane.WARNING_MESSAGE);
208     } catch (IOException ex) {
209         JOptionPane.showMessageDialog(null, ex.getMessage(), "Server Connection Error", JOptionPane.ERROR_MESSAGE);
210     }
211 }

```

Phương thức startServer():

- + Khởi tạo socket của máy chủ và bắt đầu lắng nghe các gói tin đến trên cổng đã chỉ định.
- + Xử lý các tin nhắn đến từ máy khách, phân biệt yêu cầu tham gia và tin nhắn trò chuyện.
- + Quản lý kết nối của máy khách, phát sóng tin nhắn cho tất cả các máy khách hoặc xử lý ngắt kết nối của máy khách.

```

214  private void stopServer() {
215      try {
216          connected = false;
217
218          // Gửi yêu cầu ngắt kết nối đến tất cả các client
219          synchronized (connectedClients) {
220              for (UDPClientInfo client : connectedClients) {
221                  sendMessage("SERVER_SHUTDOWN", client.getAddress(), client.getPort());
222              }
223              connectedClients.clear();
224          }
225
226          // Đóng ServerSocket
227          if (serverSocket != null && !serverSocket.isClosed()) {
228              serverSocket.close();
229          }
230
231          btnHandle.setText("Start");
232          textFieldPort.setEnabled(true);
233          onlineUsersTextArea.append("Server has been stopped.\n");
234
235      } catch (IOException e) {
236          JOptionPane.showMessageDialog(null, "Error while stopping the server: " + e.getMessage(), "Server Error", JOptionPane.ERROR_MESSAGE);
237      }
238  }

```

Phương thức stopServer():

- + Đóng socket của máy chủ và ngắt kết nối tất cả các máy khách đã kết nối.
- Cập nhật giao diện người dùng để phản ánh trạng thái dừng của máy chủ.

```

240  private void sendMessage(String message, InetAddress address, int port) throws IOException {
241      byte[] buf = message.getBytes("UTF-8");
242      DatagramPacket packetOut = new DatagramPacket(buf, buf.length, address, port);
243      serverSocket.send(packetOut);
244  }

```

Phương thức sendMessage(String message, InetAddress address, int port):

- + Gửi một tin nhắn đến một máy khách cụ thể được xác định bằng địa chỉ và cổng của nó.

```

246  private String findUsername(InetAddress address, int port) {
247      synchronized (connectedClients) {
248          for (UDPClientInfo client : connectedClients) {
249              if (client.getAddress().equals(address) && client.getPort() == port) {
250                  return client.getUsername();
251              }
252          }
253          return null;
254      }
255  }
256

```

Phương thức *findUsername(InetAddress address, int port)*: Tìm tên người dùng được liên kết với địa chỉ và cổng của một máy khách.

```

257  private void broadcastMessage(String username, String message) {
258      synchronized (connectedClients) {
259          for (UDPClientInfo client : connectedClients) {
260              if (!client.getUsername().equals(username)) {
261                  try {
262                      sendMessage("\n" + username + ":" + message, client.getAddress(), client.getPort());
263                  } catch (IOException e) {
264                      JOptionPane.showMessageDialog(null, "Error broadcasting message: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
265                  }
266              }
267          }
268      }
269  }

```

Phương thức *broadcastMessage(String username, String message)*:

- + Phát sóng một tin nhắn đến tất cả các máy khách đã kết nối ngoại trừ máy khách có tên người dùng cụ thể.

```

271  private void formWindowClosing(java.awt.event.WindowEvent evt) { //GEN-FIRST:event_formWindowClosing
272      if (serverSocket != null && !serverSocket.isClosed()) {
273          stopServer();
274      }
275  }

```

Phương thức *formWindowClosing()*: Xử lý sự kiện đóng cửa sổ, đảm bảo máy chủ được dừng khi cửa sổ GUI được đóng lại.

- **Client**

```

1 package main.UDP.ChatRoom;
2
3 import java.io.*;
4 import java.net.*;
5 import javax.swing.*;
6 import javax.swing.event.*;
7 import javax.swing.text.*;
8
9 public class UDPChatClient extends javax.swing.JFrame {
10
11     private DatagramSocket socket;
12     private InetAddress serverAddress;
13     private int serverPort;
14     private boolean connected = false;
15
16     public UDPChatClient() {
17         initComponents();
18         DocumentListener validateInput = new DocumentListener() {
19
20             @Override
21             public void insertUpdate(DocumentEvent e) {
22                 validateTextField();
23             }
24
25             @Override
26             public void removeUpdate(DocumentEvent e) {
27                 validateTextField();
28             }
29
30             @Override
31             public void changedUpdate(DocumentEvent e) {
32                 validateTextField();
33             }
34
35             protected void validateTextField() {
36                 String inputServerName = tfdServerName.getText().trim();
37                 String inputPort = tfdPort.getText().trim();
38                 String inputName = tfdName.getText().trim();
39
40                 if (!inputServerName.equals("") && !inputPort.equals("") && !inputName.equals("")) {
41                     btnHandle.setEnabled(true);
42                 } else {
43                     btnHandle.setEnabled(false);
44                 }
45             }
46         };
47
48         DocumentListener validateMsg = new DocumentListener() {
49
50             @Override
51             public void insertUpdate(DocumentEvent e) {
52                 validateTextField();
53             }
54
55             @Override
56             public void removeUpdate(DocumentEvent e) {
57                 validateTextField();
58             }
59
60             @Override
61             public void changedUpdate(DocumentEvent e) {
62                 validateTextField();
63             }
64
65             protected void validateTextField() {
66                 String inputMsg = msgArea.getText().trim();
67                 if (!inputMsg.equals("")) {
68                     btnSend.setEnabled(true);
69                 } else {
70                     btnSend.setEnabled(false);
71                 }
72             }
73         };
74
75         tfdServerName.getDocument().addDocumentListener(validateInput);
76         tfdPort.getDocument().addDocumentListener(validateInput);
77         tfdName.getDocument().addDocumentListener(validateInput);
78         msgArea.getDocument().addDocumentListener(validateMsg);
79     }

```

Khai báo biến:

- + socket: Một DatagramSocket để nhận và gửi dữ liệu qua UDP.
- + serverAddress và serverPort: Địa chỉ và cổng của máy chủ mà client sẽ kết nối đến.
- +connected: Biến đánh dấu trạng thái kết nối của client.

```

226 private void joinChatRoom() {
227     String SERVER = tfdServerName.getText().trim();
228     String inputPORT = tfdPort.getText().trim();
229     String USERNAME = tfdName.getText().trim();
230
231     try {
232         int PORT = Integer.parseInt(inputPORT);
233         if (PORT < 1 || PORT > 65535) {
234             throw new IllegalArgumentException("Port number must be between 1 and 65535");
235         }
236
237         serverAddress = InetAddress.getByName(SERVER);
238         serverPort = PORT;
239         socket = new DatagramSocket();
240
241         // Send username to server
242         sendMessage("[JOIN]" + USERNAME);
243         byte[] buf = new byte[256];
244         DatagramPacket packet = new DatagramPacket(buf, buf.length);
245
246         socket.setSoTimeout(5000);
247         try {
248             socket.receive(packet);
249             String response = new String(packet.getData(), 0, packet.getLength(), "UTF-8");
250
251             if ("USERNAME_EXISTS".equals(response)) {
252                 JOptionPane.showMessageDialog(null, "Username already taken. Please choose another name.", "Username Error", JOptionPane.ERROR_MESSAGE);
253                 socket.close();
254                 return;
255             } else if ("USERNAME_ACCEPTED".equals(response)) {
256                 appendToPane(chatDataPane, "Connected to the server as " + USERNAME + "\n", StyleConstants.ALIGN_LEFT);
257             }
258
259             // Đặt lại thời gian chờ của socket sau khi kết nối thành công
260             socket.setSoTimeout(0);
261         } catch (SocketTimeoutException e) {
262             JOptionPane.showMessageDialog(null, "Server not responding. Please try again later.", "Connection Timeout", JOptionPane.ERROR_MESSAGE);
263             socket.close();
264             return;
265         }
266     }
267
268     connected = true;
269     Thread clientThread = new Thread(() -> {
270         try {
271             byte[] bufIn = new byte[256];
272             DatagramPacket packetIn = new DatagramPacket(bufIn, bufIn.length);
273             while (connected) {
274                 socket.receive(packetIn);
275                 String msgIn = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
276
277                 if ("SERVER_SHUTDOWN".equals(msgIn)) {
278                     connected = false;
279                     appendToPane(chatDataPane, "\nServer has been shut down. You have been disconnected.\n", StyleConstants.ALIGN_LEFT);
280                     toggleInput(connected);
281                     socket.close();
282                     break;
283                 }
284
285                 appendToPane(chatDataPane, msgIn + "\n", StyleConstants.ALIGN_LEFT);
286             }
287         } catch (IOException e) {
288             if (connected) {
289                 appendToPane(chatDataPane, "\nAn error occurred while receiving the message! Please try connecting again.\n", StyleConstants.ALIGN_LEFT);
290                 leaveChatRoom();
291             }
292         }
293     });
294     clientThread.start();
295     toggleInput(connected);
296     this.setTitle(USERNAME);
297
298     } catch (NumberFormatException ex) {
299         JOptionPane.showMessageDialog(null, "Malformed port: " + inputPORT, "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
300     } catch (IllegalArgumentException ex) {
301         JOptionPane.showMessageDialog(null, ex.getMessage(), "Invalid Port Number", JOptionPane.ERROR_MESSAGE);
302     } catch (UnknownHostException ex) {
303         JOptionPane.showMessageDialog(null, "Unknown server: " + SERVER, "Unknown Host", JOptionPane.ERROR_MESSAGE);
304     } catch (SocketTimeoutException e) {
305         JOptionPane.showMessageDialog(null, "Server not responding. Please try again later.", "Connection Timeout", JOptionPane.ERROR_MESSAGE);
306     } catch (IOException ex) {
307         JOptionPane.showMessageDialog(null, "No corresponding server found", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
308     }
309 }
310 }

```

Phương thức joinChatRoom():

- + Thực hiện quá trình tham gia vào phòng chat bằng cách gửi tên người dùng và nhận xác nhận từ máy chủ.

- + Bắt đầu một luồng để lắng nghe và xử lý các tin nhắn đến từ máy chủ.

```

312  private void leaveChatRoom() {
313      try {
314          sendMessage("[CHAT]LEAVE_CHAT_ROOM");
315          connected = false; // Dừng luồng nhận tin nhắn
316          chatDataPane.setText("");
317          JOptionPane.showMessageDialog(null, "You have successfully left the chat room.", "Successfully", JOptionPane.INFORMATION_MESSAGE);
318          socket.close(); // Đóng socket đến server
319      } catch (IOException e) {
320          e.printStackTrace();
321      }
322      toggleInput(connected);
323  }

```

Phương thức leaveChatRoom():

- + Rời khỏi phòng chat bằng cách gửi thông điệp đến máy chủ và đóng socket.
- + Vô hiệu hóa các phần nhập liệu trừ khi kết nối lại.

```

326  private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnSendActionPerformed
327      String inputMsg = msgArea.getText().trim();
328
329      try {
330          sendMessage("[CHAT]" + inputMsg);
331          appendToPane(chatDataPane, inputMsg + "\n", StyleConstants.ALIGN_RIGHT);
332      } catch (IOException ex) {
333          JOptionPane.showMessageDialog(null, "An error occurred while sending the message! Please try connecting again.", "Server Connection Error", JOptionPane.ERROR_MESSAGE);
334      }
335
336      msgArea.setText("");
337  }

```

Phương thức btnSendActionPerformed(): Gửi một tin nhắn đến máy chủ trong phòng chat và hiển thị nó trên khung chat của người dùng.

```

339  private void sendMessage(String message) throws IOException {
340      byte[] buf = message.getBytes("UTF-8");
341      DatagramPacket packet = new DatagramPacket(buf, buf.length, serverAddress, serverPort);
342      socket.send(packet);
343  }

```

Phương thức sendMessage(String message): Gửi một tin nhắn đến máy chủ qua giao thức UDP.

```

345  private void formWindowClosing(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_formWindowClosing
346      if (socket != null && !socket.isClosed()) {
347          leaveChatRoom();
348      }
349  }

```

Phương thức *formWindowClosing()*: Xử lý sự kiện đóng cửa sổ, đảm bảo rằng người dùng sẽ rời khỏi phòng chat khi cửa sổ đóng lại.

```

351  private void toggleInput(boolean connected) {
352      btnHandle.setText(connected ? "Leave" : "Join");
353      tfdServerName.setEnabled(!connected);
354      tfdPort.setEnabled(!connected);
355      tfdName.setEnabled(!connected);
356      msgArea.setText("");
357      msgArea.setEnabled(connected);
358  }

```

Phương thức *toggleInput(boolean connected)*: Bật hoặc tắt các phần nhập liệu dựa trên trạng thái kết nối.

```

360  private void appendToPane(JTextPane tp, String msg, int alignment) {
361      StyledDocument doc = tp.getStyledDocument();
362      SimpleAttributeSet attr = new SimpleAttributeSet();
363      StyleConstants.setAlignment(attr, alignment);
364
365      try {
366          doc.insertString(doc.getLength(), msg, attr);
367          doc.setParagraphAttributes(doc.getLength() - msg.length(), msg.length(), attr, false);
368      } catch (BadLocationException e) {
369          e.printStackTrace();
370      }
371  }

```

Phương thức *appendToPane(JTextPane tp, String msg, int alignment)*:

- + Thêm một chuỗi vào JTextPane và thiết lập căn chỉnh cho nó.

- **UDPClientInfo**

```
1 package main.UDP.ChatRoom;
2
3 import java.net.InetAddress;
4
5 public class UDPClientInfo {
6
7     private final String username;
8     private final InetAddress address;
9     private final int port;
10
11    public UDPClientInfo(String username, InetAddress address, int port) {
12        this.username = username;
13        this.address = address;
14        this.port = port;
15    }
16
17    public String getUsername() {
18        return username;
19    }
20
21    public InetAddress getAddress() {
22        return address;
23    }
24
25    public int getPort() {
26        return port;
27    }
28}
29
```

3.1.2.8 UDP – Chương trình TicTacToe Game (Trò chơi XO)

- **Server**

```

1 package main.UDP.TicTacToe;
2
3 import main.UDP.ChatRoom.*;
4 import java.io.*;
5 import java.net.*;
6 import java.util.*;
7 import javax.swing.*;
8
9 public class UDPTicTacToeServer extends javax.swing.JFrame {
10
11     private DatagramSocket serverSocket;
12     private boolean connected = false;
13     private char[][] board = new char[3][3];
14     private int connectedCount = 0; // Số lượng client đã kết nối
15
16     private final Set<UDPClientInfo> connectedClients = new HashSet<>();
17
18     public UDPTicTacToeServer() {
19         initComponents();
20     }

```

Khai báo biến:

- + serverSocket: Một DatagramSocket để nhận và gửi dữ liệu qua UDP.
- + connected: Biến đánh dấu trạng thái kết nối của server.
- + board: Một mảng 2 chiều dùng để lưu trạng thái của bảng Tic Tac Toe.
- + connectedCount: Số lượng client đã kết nối.
- + connectedClients: Một tập hợp các đối tượng UDPClientInfo đại diện cho các client đã kết nối.

```

113 private void startServer() {
114     ...
115     Thread serverThread = new Thread(() -> {
116         try {
117             while (connected) {
118                 serverSocket.receive(packetIn);
119                 String msgIn = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
120                 InetAddress clientAddress = packetIn.getAddress();
121                 int clientPort = packetIn.getPort();
122                 String clientUsername = findUsername(clientAddress, clientPort);
123
124                 // Chỉ chấp nhận kết nối từ hai client cùng lúc
125                 if (connectedCount == 2 && clientUsername == null) {
126                     // Gửi thông báo từ chủ kết nối đến client mới
127                     sendMessage("CONNECTION_REFUSED", clientAddress, clientPort);
128                     continue;
129                 }
130
131                 if ("[JOIN]".equals(msgIn)) {
132                     String username;
133                     // Gửi danh sách các tên người dùng hiện tại cho client mới
134                     if (connectedCount == 0) {
135                         username = "X";
136                         sendMessage("[WAIT]", clientAddress, clientPort);
137                         // Gửi thông tin lượt chơi cho client 1 là 'X'
138                         sendMessage(username, clientAddress, clientPort);
139                     } else {
140                         username = "O";
141                         sendMessage("[START]", clientAddress, clientPort);
142                         // Gửi thông tin lượt chơi cho client 2 là 'O'
143                         sendMessage(username, clientAddress, clientPort);
144                     }
145
146                     // Tạo một ClientInfo mới và thêm vào danh sách connectedClients
147                     UDPClientInfo client = new UDPClientInfo(username, clientAddress, clientPort);
148                     connectedClients.add(client);
149                     connectedCount++;
150
151                     if (connectedCount == 2) {
152                         // Gửi tín hiệu bắt đầu trò chơi đến cả 2 client
153                         broadcastMessage(null, "[GO]X");
154                     }
155
156                     // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
157                     onlineUsersTextArea.append(username + ":" + clientAddress + " at port " + clientPort + " has connected to the game room\n");
158
159                 } else if (msgIn.startsWith("[PLAY]")) {
160                     String[] positionStr = msgIn.split(" ");
161                     int position1 = Integer.parseInt(positionStr[1]);
162                     int position2 = Integer.parseInt(positionStr[2]);
163
164                     // Gửi thông điệp vị trí đánh cho client còn lại
165                     broadcastMessage(clientUsername, "[FILL] " + position1 + " " + position2);
166
167                     // Thực hiện nút di và kiểm tra kết quả
168                     boolean gameEnded = makeMoveAndCheckResult(clientUsername, position1, position2);
169                     if (gameEnded) {
170                         // Gửi thông điệp kết thúc trò chơi cho cả hai người chơi
171                         broadcastMessage(null, "[END] " + getWinner());
172                     } else {
173                         // Nếu trò chơi chưa kết thúc, gửi thông điệp lượt đánh cho cả hai người chơi
174                         String nextTurn = clientUsername.equals("X") ? "O" : "X";
175                         broadcastMessage(null, "[GO] " + nextTurn);
176                     }
177
178                     // Cập nhật giao diện để hiển thị lượt đánh
179                     onlineUsersTextArea.append(clientUsername + ":" + clientAddress + " at port " + clientPort
180                         + " has filled to " + position1 + " " + position2 + "\n");
181
182                 } else if ("LEAVE_CHAT_ROOM".equals(msgIn)) {
183                     // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
184                     onlineUsersTextArea.append(clientUsername + ":" + clientAddress + " at port " + clientPort + " has disconnected to the game room\n");
185
186                     // Thông báo client đã rời khỏi phòng cho client khác
187                     String absoluteWinner = clientUsername.equals("X") ? "O" : "X";
188                     broadcastMessage(null, "[END] " + absoluteWinner);
189
190                     // Ngắt kết nối khi client thoát
191                     synchronized (connectedClients) {
192                         connectedClients.removeIf(client -> client.getAddress().equals(clientAddress) && client.getPort() == clientPort);
193                         connectedCount--;
194                     }
195                 }
196             }
197         } catch (IOException ex) {
198             if (connected) {
199                 JOptionPane.showMessageDialog(null, ex.getMessage(), "IOException", JOptionPane.ERROR_MESSAGE);
200             }
201         }
202     });
203
204     ...
205 }

```

Phương thức startServer():

+ Bắt đầu máy chủ trò chơi bằng cách mở một socket trên cổng được chỉ định.

- + Chấp nhận kết nối từ hai người chơi.
- + Thực hiện quản lý vị trí của các nước đi, gửi thông điệp về kết quả và kiểm tra trò chơi.

```

290 // Lưu vị trí nước đi vào mảng 2 chiều và kiểm tra kết quả sau mỗi nước đi
291 private boolean makeMoveAndCheckResult(String clientUsername, int position1, int position2) {
292     // Kiểm tra xem nước đi có hợp lệ không (nằm trong phạm vi mảng và ô đó chưa được đánh)
293     if (position1 < 0 || position1 > 2 || position2 < 0 || position2 > 2 || board[position1][position2] != '\u0000') {
294         // Nước đi không hợp lệ, trả về false
295         return false;
296     }
297
298     // Lưu nước đi vào bảng
299     board[position1][position2] = (clientUsername.equals("X")) ? 'X' : 'O';
300
301     // Kiểm tra xem có ai chiến thắng không sau mỗi nước đi
302     char winner = getWinner();
303
304     // Nước đi hợp lệ và không có ai chiến thắng, trả về false
305     return winner != '-';
306 }
```

Phương thức makeMoveAndCheckResult(String clientUsername, int position1, int position2): Lưu vị trí nước đi vào bảng và kiểm tra kết quả sau mỗi nước đi.

```

308 // Hàm kiểm tra kết quả của trò chơi sau mỗi nước đi
309 private char getWinner() {
310     // Kiểm tra theo các hàng, cột và đường chéo
311     for (int i = 0; i < 3; i++) {
312         if (board[i][0] != '\u0000' && board[i][0] == board[i][1] && board[i][0] == board[i][2]) {
313             return board[i][0]; // Người chiến thắng
314         }
315         if (board[0][i] != '\u0000' && board[0][i] == board[1][i] && board[0][i] == board[2][i]) {
316             return board[0][i]; // Người chiến thắng
317         }
318     }
319     // Kiểm tra đường chéo chính
320     if (board[0][0] != '\u0000' && board[0][0] == board[1][1] && board[0][0] == board[2][2]) {
321         return board[0][0]; // Người chiến thắng
322     }
323     // Kiểm tra đường chéo phụ
324     if (board[0][2] != '\u0000' && board[0][2] == board[1][1] && board[0][2] == board[2][1]) {
325         return board[0][2]; // Người chiến thắng
326     }
327
328     // Nếu không có ai chiến thắng
329     return '-';
330 }
```

Phương thức getWinner(): Kiểm tra kết quả của trò chơi sau mỗi nước đi để xác định người chiến thắng hoặc trạng thái tiếp theo của trò chơi.

- Client

```

1 package main.UDP.TicTacToe;
2
3 import java.io.*;
4 import java.net.*;
5 import javax.swing.*;
6
7 public class UDPTicTacToeClient extends javax.swing.JFrame {
8
9     private DatagramSocket socket;
10    private InetAddress serverAddress;
11    private int serverPort;
12    private boolean connected = false;
13    private final String SERVER = "localhost";
14    private final JButton[][] board = new JButton[3][3];
15    private String turn;
16
17    public UDPTicTacToeClient() {
18        initComponents();
19        board[0][0] = btn0;
20        board[0][1] = btn1;
21        board[0][2] = btn2;
22        board[1][0] = btn3;
23        board[1][1] = btn4;
24        board[1][2] = btn5;
25        board[2][0] = btn6;
26        board[2][1] = btn7;
27        board[2][2] = btn8;
28        toggleBoard(false);
29    }

```

Khai báo biến:

- + Biến board là một mảng 2 chiều chứa các nút JButton đại diện cho bảng Tic-Tac-Toe. Mỗi nút này sẽ được gán giá trị từ các thành phần của giao diện người dùng.
- + Biến connected được sử dụng để theo dõi trạng thái kết nối với máy chủ.

- + Biến turn sẽ chứa thông tin về lượt đánh của người chơi hiện tại.
- + Các nút trên bảng Tic-Tac-Toe được gán sự kiện ActionPerformed để xử lý khi người chơi nhấn vào chúng và gửi vị trí của nước đi đến máy chủ.

```

229 private void joinGameRoom() {
230 ...
231     serverAddress = InetAddress.getByName(SERVER);
232     serverPort = PORT;
233     socket = new DatagramSocket();
234
235     // Gửi yêu cầu tham gia trò chơi đến máy chủ
236     sendMessage("[JOIN]");
237
238     // Nhận phản hồi từ máy chủ
239     byte[] buf = new byte[256];
240     DatagramPacket packet = new DatagramPacket(buf, buf.length);
241     socket.setSoTimeout(5000);
242
243     try {
244         socket.receive(packet);
245         String response = new String(packet.getData(), 0, packet.getLength(), "UTF-8");
246
247         switch (response) {
248             case "CONNECTION_REFUSED" -> {
249                 lblMessage.setText("Sorry, there are currently enough players!");
250                 return;
251             }
252             case "[WAIT]" -> {
253                 socket.receive(packet);
254                 turn = new String(packet.getData(), 0, packet.getLength(), "UTF-8");
255                 lblMessage.setText("Your turn is \"\\" + turn + "\' ~ Waiting for your opponent...");
256             }
257             case "[START]" -> {
258                 socket.receive(packet);
259                 turn = new String(packet.getData(), 0, packet.getLength(), "UTF-8");
260             }
261             default -> {
262             }
263         }
264     }
265
266     // Đặt lại thời gian chờ của socket sau khi kết nối thành công
267     socket.setSoTimeout(0);
268
269 } catch (SocketTimeoutException e) {
270     JOptionPane.showMessageDialog(null, "Server not responding. Please try again later.", "Connection Timeout", JOptionPane.ERROR_MESSAGE);
271     socket.close();
272     return;
273 }
274
275 connected = true;
276 Thread clientThread = new Thread(() -> {
277     try {
278         byte[] bufIn = new byte[256];
279         DatagramPacket packetIn = new DatagramPacket(bufIn, bufIn.length);
280         while (connected) {
281             socket.receive(packetIn);
282             String msgIn = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
283
284             if (msgIn.startsWith("[GO]")) {
285                 String currentTurn = msgIn.substring(msgIn.length() - 1);
286
287                 if (turn.equals(currentTurn)) {
288                     lblMessage.setText("Now is your turn.");
289                     toggleBoard(true);
290                 } else {
291                     lblMessage.setText("Wait for \"\\" + currentTurn + "\' turn.");
292                     toggleBoard(false);
293                 }
294             }
295             if (msgIn.startsWith("[FILL]")) {
296                 String[] positionStr = msgIn.split(" ");
297                 int position1 = Integer.parseInt(positionStr[1]);
298                 int position2 = Integer.parseInt(positionStr[2]);
299                 String opponentTurn = turn.equals("X") ? "O" : "X";
300                 board[position1][position2].setText(opponentTurn);
301             }
302             if (msgIn.startsWith("[END]")) {
303                 char winner = msgIn.charAt(msgIn.length() - 1);
304                 String message;
305                 if (winner == '-') {
306                     message = "The game ended in a draw.";
307                 } else {
308                     message = "Player " + winner + " wins!";
309                 }
310                 lblMessage.setText(message);
311                 toggleBoard(false);
312             }
313             if ("SERVER_SHUTDOWN".equals(msgIn)) {
314                 connected = false;
315                 JOptionPane.showMessageDialog(null, "\nServer has been shut down. You have been disconnected.\n", "Server Shutdown", JOptionPane.ERROR_MESSAGE);
316                 toggleInput(connected);
317                 socket.close();
318                 break;
319             }
320         }
321     }
322     } catch (IOException e) {
323         if (connected) {
324             leaveGameRoom();
325         }
326     }
327 });
328 ...
329 }
330 }

```

Phương thức joinGameRoom():

- + Được sử dụng để tham gia phòng chơi trò chơi Tic-Tac-Toe thông qua máy chủ.
- + Thiết lập kết nối và xử lý các thông điệp từ máy chủ như thông báo, lượt chơi, và kết quả của trò chơi.

```

379  private void clearBoard() {
380      for (JButton[] subList : board) {
381          for (JButton board : subList) {
382              board.setText("");
383          }
384      }
385  }
386
387  private void toggleBoard(boolean status) {
388      for (JButton[] subList : board) {
389          for (JButton board : subList) {
390              board.setEnabled(status);
391          }
392      }
393  }

437  private void sendPosition(int position1, int position2) {
438      // Kiểm tra vị trí đã đánh dấu chưa
439      if (!board[position1][position2].getText().isEmpty()) {
440          return;
441      }
442      // Gửi vị trí đánh cho server
443      try {
444          // Tạo thông điệp chứa vị trí đánh và Gửi thông điệp đến server
445          sendMessage("[PLAY] " + position1 + " " + position2);
446          board[position1][position2].setText(turn);
447      } catch (IOException e) {
448          e.printStackTrace();
449      }
450  }
451

```

Các phương thức phụ trợ khác:

- + ***clearBoard()***: Xóa toàn bộ bảng Tic-Tac-Toe bằng cách đặt trạng thái của các nút về trống.
- + ***toggleBoard(boolean status)***: Cho phép hoặc vô hiệu hóa các nút trên bảng dựa trên trạng thái được chỉ định.
- + ***sendPosition(int position1, int position2)***: Gửi thông điệp đến máy chủ để thông báo về vị trí đánh của người chơi.

3.1.2.9 UDP – Chương trình Length Converter (Chuyển đổi độ dài)

- **Server**

```

1  package main.UDP.LengthConverter;
2
3  import java.io.*;
4  import java.net.*;
5  import java.util.*;
6  import javax.swing.*;
7
8  public class UDPLengthConverterServer extends javax.swing.JFrame {
9
10     private DatagramSocket serverSocket;
11     private final Map<String, Double> lengthUnits = new HashMap<>();
12     private boolean connected = false;
13
14     public UDPLengthConverterServer() {
15         initComponents();
16
17         lengthUnits.put("meter", 1.0);
18         lengthUnits.put("kilometer", 1000.0);
19         lengthUnits.put("centimeter", 0.01);
20         lengthUnits.put("millimeter", 0.001);
21         lengthUnits.put("mile", 1609.34);
22         lengthUnits.put("yard", 0.9144);
23         lengthUnits.put("foot", 0.3048);
24         lengthUnits.put("inch", 0.0254);
25     }

```

Khai báo biến:

- + Biến lengthUnits lưu trữ các hệ số chuyển đổi cho các đơn vị độ dài khác nhau.
- + Những hệ số này được sử dụng để chuyển đổi độ dài từ một đơn vị sang đơn vị khác.
- + Trong hàm tạo, các thành phần Swing được khởi tạo, và biến lengthUnits được điền vào với các hệ số chuyển đổi cho các đơn vị độ dài khác nhau

```

118  private void startServer() {
119      ...
120      connected = true;
121      Thread serverThread = new Thread(() -> {
122          try {
123              while (connected) {
124                  serverSocket.receive(packetIn);
125                  String msgIn = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
126                  InetAddress clientAddress = packetIn.getAddress();
127                  int clientPort = packetIn.getPort();
128
129                  if ("[JOIN]".equals(msgIn)) {
130                      // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
131                      onlineUsersTextArea.append(clientAddress + " at port " + clientPort + " has connected to the Length Converter room\n");
132
133                      // Gửi danh sách các đơn vị độ dài mà server hỗ trợ cho client
134                      StringBuilder units = new StringBuilder();
135                      for (String unit : lengthUnits.keySet()) {
136                          units.append(unit).append(",");
137                      }
138                      String unitsStr = units.toString();
139                      sendMessage(unitsStr, clientAddress, clientPort);
140
141                  } else if (msgIn.startsWith("[CONVERT]")) {
142                      // Đọc thông tin chuyển đổi từ client
143                      String[] parts = msgIn.split(",");
144                      if (parts.length == 4) {
145                          String fromUnit = parts[1];
146                          String toUnit = parts[2];
147                          String valueStr = parts[3];
148
149                          try {
150                              double value = Double.parseDouble(valueStr);
151                              // Chuyển đổi độ dài
152                              double result = convertLength(fromUnit, toUnit, value);
153                              String resultStr = "" + result;
154                              sendMessage("[RESULT]", clientAddress, clientPort);
155                              sendMessage(resultStr, clientAddress, clientPort);
156
157                              String reqMsg = "Convert " + valueStr + " " + fromUnit + " to " + toUnit;
158                              onlineUsersTextArea.append(clientAddress + " at port " + clientPort + " has requested " + reqMsg + "\n");
159
160                          } catch (NumberFormatException e) {
161                              // Gửi thông báo lỗi về cho client nếu dữ liệu không phải là số
162                              String errorMsg = "Invalid number format!";
163                              sendMessage("[ERROR]", clientAddress, clientPort);
164                              sendMessage(errorMsg, clientAddress, clientPort);
165                          }
166
167                  } else if ("[LEAVE]".equals(msgIn)) {
168                      // Cập nhật giao diện để hiển thị danh sách người dùng đang kết nối trên server
169                      onlineUsersTextArea.append(clientAddress + " at port " + clientPort + " has disconnected from the Length Converter room\n");
170                  }
171
172          } catch (IOException ex) {
173              if (connected) {
174                  JOptionPane.showMessageDialog(null, ex.getMessage(), "IOException", JOptionPane.ERROR_MESSAGE);
175              }
176          }
177      });
178
179      ...
180  }

```

Phương thức startServer():

- + Phương thức này khởi tạo và bắt đầu máy chủ UDP.

- + Nó lắng nghe các gói tin UDP đến, xử lý chúng và phản hồi tương ứng.
- + Khi nhận được thông điệp [JOIN] từ một client, nó gửi lại một danh sách các đơn vị độ dài được hỗ trợ.
- + Khi nhận được một thông điệp [CONVERT] từ một client, nó thực hiện việc chuyển đổi yêu cầu và gửi lại kết quả.
- + Nếu một client gửi một thông điệp LEAVE, nó cập nhật giao diện máy chủ tương ứng.

```

223  private double convertLength(String fromUnit, String toUnit, double value) {
224      if (!lengthUnits.containsKey(fromUnit) || !lengthUnits.containsKey(toUnit)) {
225          throw new IllegalArgumentException("Invalid length unit");
226      }
227      double fromFactor = lengthUnits.get(fromUnit);
228      double toFactor = lengthUnits.get(toUnit);
229      double result = value * (fromFactor / toFactor);
230      return Math.round(result * 100.0) / 100.0;
231  }

```

Phương thức *convertLength()*: Phương thức này tính toán độ dài chuyển đổi dựa trên các hệ số chuyển đổi được cung cấp và giá trị đầu vào.

- **Client**

```

174 private void connectToServer() {
175 ...
176     serverAddress = InetAddress.getByName(SERVER);
177     serverPort = PORT;
178     socket = new DatagramSocket();
179
180     // Gửi yêu cầu tham gia trò chơi đến máy chủ
181     sendMessage("[JOIN]");
182
183     // Nhận phản hồi từ máy chủ
184     byte[] buf = new byte[256];
185     DatagramPacket packetIn = new DatagramPacket(buf, buf.length);
186     socket.setSoTimeout(5000);
187
188     try {
189         socket.receive(packetIn);
190         String response = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
191
192         if (!response.isEmpty()) {
193             ArrayList<String> unitList = parseUnits(response);
194
195             // Diền các đơn vị độ dài vào hai JComboBox
196             fillComboBoxes(unitList);
197
198             // Kích hoạt nút Convert
199             btnConvert.setEnabled(true);
200         }
201
202         // Đặt lại thời gian chờ của socket sau khi kết nối thành công
203         socket.setSoTimeout(0);
204
205     } catch (SocketTimeoutException e) {
206         JOptionPane.showMessageDialog(null, "Server not responding. Please try again later.", "Connection Timeout", JOptionPane.ERROR_MESSAGE);
207         socket.close();
208         return;
209     }
210
211     connected = true;
212     Thread clientThread = new Thread(() -> {
213         try {
214             while (connected) {
215                 socket.receive(packetIn);
216                 String msgIn = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
217
218                 if ("[RESULT]".equals(msgIn)) {
219                     // Nhận kết quả từ máy chủ và hiển thị lên trường kết quả
220                     socket.receive(packetIn);
221                     String result = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
222                     tfdResult.setText(result);
223
224                 } else if ("[ERROR]".equals(msgIn)) {
225                     socket.receive(packetIn);
226                     String errorMsg = new String(packetIn.getData(), 0, packetIn.getLength(), "UTF-8");
227                     JOptionPane.showMessageDialog(null, "Error: " + errorMsg, "Error", JOptionPane.ERROR_MESSAGE);
228
229                 } else if ("[SERVER_SHUTDOWN]".equals(msgIn)) {
230                     connected = false;
231                     JOptionPane.showMessageDialog(null, "\nServer has been shut down. You have been disconnected.\n", "Server Shutdown", JOptionPane.ERROR_MESSAGE);
232                     toggleInput(connected);
233                     socket.close();
234                     break;
235                 }
236             }
237         }
238     });
239     ...
240 }

```

Phương thức connectToServer():

- + Phương thức này thực hiện kết nối tới máy chủ UDP.
- + Khi kết nối thành công, nó gửi một yêu cầu [JOIN] tới máy chủ để tham gia trò chơi và nhận phản hồi từ máy chủ.
- + Nếu nhận được phản hồi từ máy chủ, nó xử lý dữ liệu đơn vị độ dài được gửi từ máy chủ và điền vào các JComboBox tương ứng.
- + Sau đó, nó khởi tạo một luồng để nhận dữ liệu từ máy chủ, xử lý và hiển thị kết quả hoặc thông báo lỗi tương ứng.

```

275  private void btnConvertActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_btnConvertActionPerformed
276      String value = tfdInput.getText().trim();
277      if (!value.isEmpty()) {
278          // Lấy đơn vị từ JComboBox thứ nhất
279          String fromUnit = (String) cbxUnit1.getSelectedItem();
280
281          // Lấy đơn vị từ JComboBox thứ hai
282          String toUnit = (String) cbxUnit2.getSelectedItem();
283
284          String message = "[CONVERT]," + fromUnit + "," + toUnit + "," + value;
285          try {
286              sendMessage(message);
287          } catch (IOException e) {
288              e.printStackTrace();
289          }
290      } else {
291          JOptionPane.showMessageDialog(null, "Input value cannot be empty!", "Error", JOptionPane.ERROR_MESSAGE);
292      }
293  }
294
295 }

```

Phương thức btnConvertActionPerformed():

- + Phương thức này được gọi khi người dùng nhấn nút "Convert".
- + Nó kiểm tra xem giá trị đầu vào có trống không.
- + Nếu giá trị không trống, nó lấy các đơn vị độ dài từ các JComboBox và gửi yêu cầu chuyển đổi tới máy chủ với giá trị và đơn vị tương ứng.

```

297  private ArrayList<String> parseUnits(String units) {
298      ArrayList<String> unitList = new ArrayList<>();
299      String[] unitArray = units.split(",");
300      for (String unit : unitArray) {
301          unitList.add(unit.trim());
302      }
303      return unitList;
304  }

```

***Phương thức parseUnits():* Chuyển đổi chuỗi các đơn vị độ dài từ máy chủ thành một danh sách các đơn vị.**

```
306  private void fillComboBoxes(ArrayList<String> units) {  
307      DefaultComboBoxModel<String> modelFrom = new DefaultComboBoxModel<>();  
308      DefaultComboBoxModel<String> modelTo = new DefaultComboBoxModel<>();  
309      for (String unit : units) {  
310          modelFrom.addElement(unit);  
311          modelTo.addElement(unit);  
312      }  
313      cbxUnit1.setModel(modelFrom);  
314      cbxUnit2.setModel(modelTo);  
315  }
```

Phương thức fillComboBoxes(): Đưa danh sách các đơn vị độ dài vào các JComboBox tương ứng.

3.1.2.10 UDP – Chương trình Weight Converter (Chuyển đổi cân nặng)

- **Server**

```

1 package main.UDP.WeightConverter;
2
3 import java.io.*;
4 import java.net.*;
5 import java.util.*;
6 import javax.swing.*;
7
8 public class UDPWeightConverterServer extends javax.swing.JFrame {
9
10    private DatagramSocket serverSocket;
11    private final Map<String, Double> weightUnits = new HashMap<>();
12    private boolean connected = false;
13
14    public UDPWeightConverterServer() {
15        initComponents();
16
17        weightUnits.put("kilogram", 1.0);
18        weightUnits.put("gram", 0.001);
19        weightUnits.put("milligram", 0.000001);
20        weightUnits.put("microgram", 0.00000001);
21        weightUnits.put("ton", 1000.0);
22        weightUnits.put("pound", 0.453592);
23        weightUnits.put("ounce", 0.0283495);
24        weightUnits.put("stone", 6.35029);
25    }

```

Khai báo biến:

- + Biến weightUnits lưu trữ các hệ số chuyển đổi cho các đơn vị cân nặng khác nhau.
- + Những hệ số này được sử dụng để chuyển đổi cân nặng từ một đơn vị sang đơn vị khác.
- + Trong hàm tạo, các thành phần Swing được khởi tạo, và biến weightUnits được điền vào với các hệ số chuyển đổi cho các đơn vị cân nặng khác nhau.

```

226  private double convertWeight(String fromUnit, String toUnit, double value) {
227      if (!weightUnits.containsKey(fromUnit) || !weightUnits.containsKey(toUnit)) {
228          throw new IllegalArgumentException("Invalid weight unit");
229      }
230      double fromFactor = weightUnits.get(fromUnit);
231      double toFactor = weightUnits.get(toUnit);
232      double result = value * (fromFactor / toFactor);
233      return Math.round(result * 100.0) / 100.0;
234  }

```

Phương thức convertWeight():

- + Phương thức này nhận các thông số là đơn vị từ, đơn vị đến và giá trị cần chuyển đổi.
- + Nó kiểm tra tính hợp lệ của đơn vị trọng lượng từ và đến.
- + Sau đó, nó thực hiện chuyển đổi giá trị dựa trên tỷ lệ giữa các đơn vị và trả về kết quả đã làm tròn đến hai chữ số sau dấu thập phân.

- **Client**

Tương tự như code của Client trong các ứng dụng chuyển đổi khác

3.1.2.11 UDP – Chương trình Currency Converter (Chuyển đổi tiền tệ)

- **Server**

```
1 package main.UDP.CurrencyConverter;
2
3 import java.io.*;
4 import java.net.*;
5 import java.util.*;
6 import javax.swing.*;
7
8 public class UDPCurrencyConverterServer extends javax.swing.JFrame {
9
10    private DatagramSocket serverSocket;
11    private final Map<String, Double> currencyRates = new HashMap<>();
12    private boolean connected = false;
13
14    public UDPCurrencyConverterServer() {
15        initComponents();
16
17        currencyRates.put("USD", 1.0);
18        currencyRates.put("EUR", 0.85);
19        currencyRates.put("JPY", 110.0);
20        currencyRates.put("GBP", 0.75);
21        currencyRates.put("AUD", 1.4);
22        currencyRates.put("CAD", 1.25);
23        currencyRates.put("CHF", 0.92);
24        currencyRates.put("CNY", 6.45);
25        currencyRates.put("SEK", 8.6);
26        currencyRates.put("NZD", 1.5);
27        currencyRates.put("INR", 73.0);
28        currencyRates.put("BRL", 5.3);
29        currencyRates.put("RUB", 74.0);
30        currencyRates.put("MXN", 20.0);
31        currencyRates.put("ZAR", 14.5);
32        currencyRates.put("TRY", 8.3);
33        currencyRates.put("SAR", 3.75);
34        currencyRates.put("SGD", 1.35);
35        currencyRates.put("HKD", 7.75);
36        currencyRates.put("NOK", 8.5);
37        currencyRates.put("KRW", 1175.0);
38        currencyRates.put("THB", 33.0);
39        currencyRates.put("MYR", 4.1);
40        currencyRates.put("IDR", 14200.0);
41        currencyRates.put("DKK", 6.3);
42        currencyRates.put("PLN", 3.8);
43        currencyRates.put("HUF", 300.0);
44        currencyRates.put("CZK", 21.5);
45        currencyRates.put("ILS", 3.25);
46        currencyRates.put("CLP", 780.0);
47        currencyRates.put("PHP", 50.0);
48        currencyRates.put("AED", 3.67);
49        currencyRates.put("COP", 3800.0);
50        currencyRates.put("EGP", 15.7);
51        currencyRates.put("KZT", 425.0);
52        currencyRates.put("PKR", 160.0);
53        currencyRates.put("BDT", 85.0);
54        currencyRates.put("NGN", 410.0);
55        currencyRates.put("VND", 23000.0);
56    }
```

Khai bảo biến:

- + Biến currencyUnits lưu trữ các hệ số chuyển đổi cho các đơn vị tiền tệ khác nhau.
- + Những hệ số này được sử dụng để chuyển đổi tiền tệ từ một đơn vị sang đơn vị khác.
- + Trong hàm tạo, các thành phần Swing được khởi tạo, và biến currencyUnits được điền vào với các hệ số chuyển đổi cho các đơn vị tiền tệ khác nhau.

```

257  private double convertCurrency(String fromUnit, String toUnit, double value) {
258      if (!currencyRates.containsKey(fromUnit) || !currencyRates.containsKey(toUnit)) {
259          throw new IllegalArgumentException("Invalid currency unit");
260      }
261      double fromFactor = currencyRates.get(fromUnit);
262      double toFactor = currencyRates.get(toUnit);
263      double result = value * (toFactor / fromFactor);
264      return Math.round(result * 100.0) / 100.0;
265  }

```

Phương thức convertCurrency():

- + Phương thức này nhận các thông số là tiền tệ từ, tiền tệ đến và giá trị cần chuyển đổi.
- + Nó kiểm tra tính hợp lệ của đơn vị tiền tệ từ và đến.
- + Sau đó, nó thực hiện chuyển đổi giá trị dựa trên tỷ lệ giữa các đơn vị và trả về kết quả đã làm tròn đến hai chữ số sau dấu thập phân.

- **Client**

Tương tự như code của Client trong các ứng dụng chuyển đổi khác

3.1.2.12 UDP – Chương trình Temperature Converter (Chuyển đổi nhiệt độ)

- **Server**

Tương tự với code của Server trong các ứng dụng chuyển đổi khác

```

211  private double convertTemperature(String fromUnit, String toUnit, double value) {
212      double result = 0;
213      if (fromUnit.equals(toUnit)) {
214          // Nếu đơn vị đầu vào và đầu ra giống nhau, kết quả là giá trị ban đầu
215          result = value;
216      } else if (fromUnit.equals("C") && toUnit.equals("F")) {
217          result = (value * 9 / 5) + 32;
218      } else if (fromUnit.equals("F") && toUnit.equals("C")) {
219          result = (value - 32) * 5 / 9;
220      } else if (fromUnit.equals("C") && toUnit.equals("K")) {
221          result = value + 273.15;
222      } else if (fromUnit.equals("K") && toUnit.equals("C")) {
223          result = value - 273.15;
224      } else if (fromUnit.equals("F") && toUnit.equals("K")) {
225          result = (value - 32) * 5 / 9 + 273.15;
226      } else if (fromUnit.equals("K") && toUnit.equals("F")) {
227          result = (value - 273.15) * 9 / 5 + 32;
228      } else {
229          throw new IllegalArgumentException("Invalid temperature unit");
230      }
231      return Math.round(result * 100.0) / 100.0; // Làm tròn đến 2 chữ số thập phân
232  }

```

Phương thức convertTemperature ():

- + Thực hiện chuyển đổi nhiệt độ từ một đơn vị sang một đơn vị khác. Nó xử lý các trường hợp chuyển đổi giữa độ Celsius (C), Fahrenheit (F) và Kelvin (K).
- + Nếu đơn vị đầu vào và đầu ra giống nhau, phương thức trả về giá trị ban đầu. Nếu không, nó sử dụng các công thức chuyển đổi phù hợp và trả về kết quả, được làm tròn đến hai chữ số sau dấu thập phân.
- + Nếu đơn vị nhiệt độ không hợp lệ, phương thức ném ra một ngoại lệ với thông báo lỗi "Invalid temperature unit".

- **Client**

Tương tự như code của Client trong các ứng dụng chuyển đổi khác

3.1.3 Test Demo

3.1.3.1 Hướng dẫn chạy chương trình

Bước 0: Cần cài đặt JDK21 để có thể tương thích tốt nhất với chương trình.

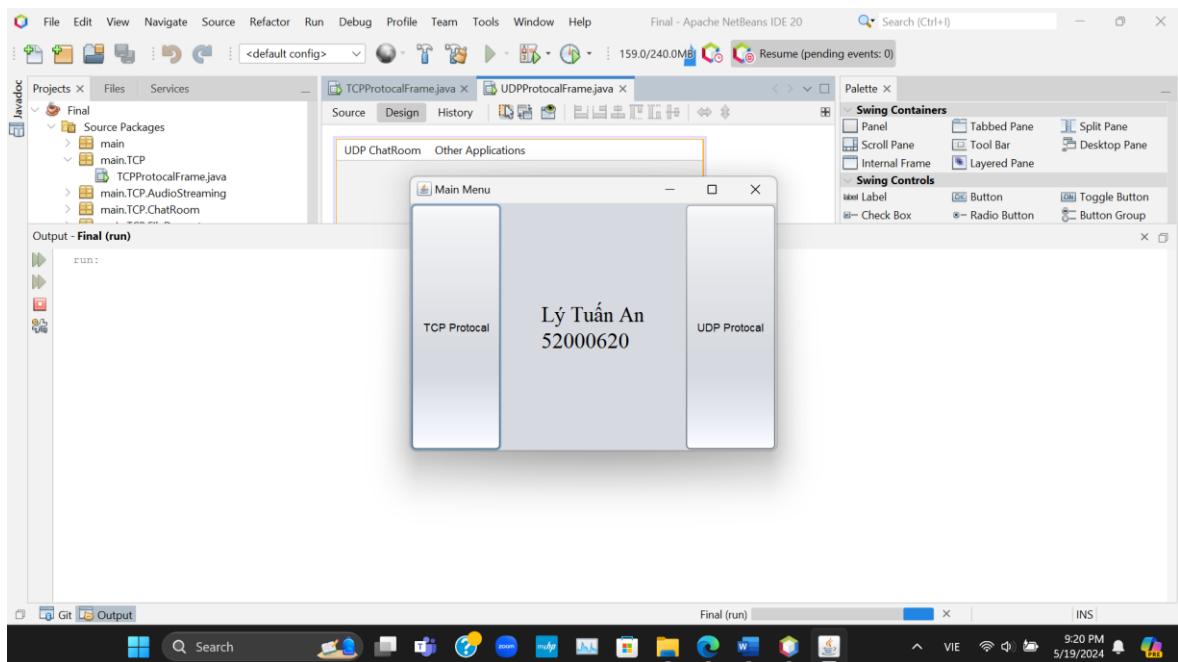
Bước 1:

Cách 1: Thực hiện mở file mainmenu.jar hoặc nhập lệnh java -jar mainmenu.jar để có thể chạy chương trình ở môi trường production.

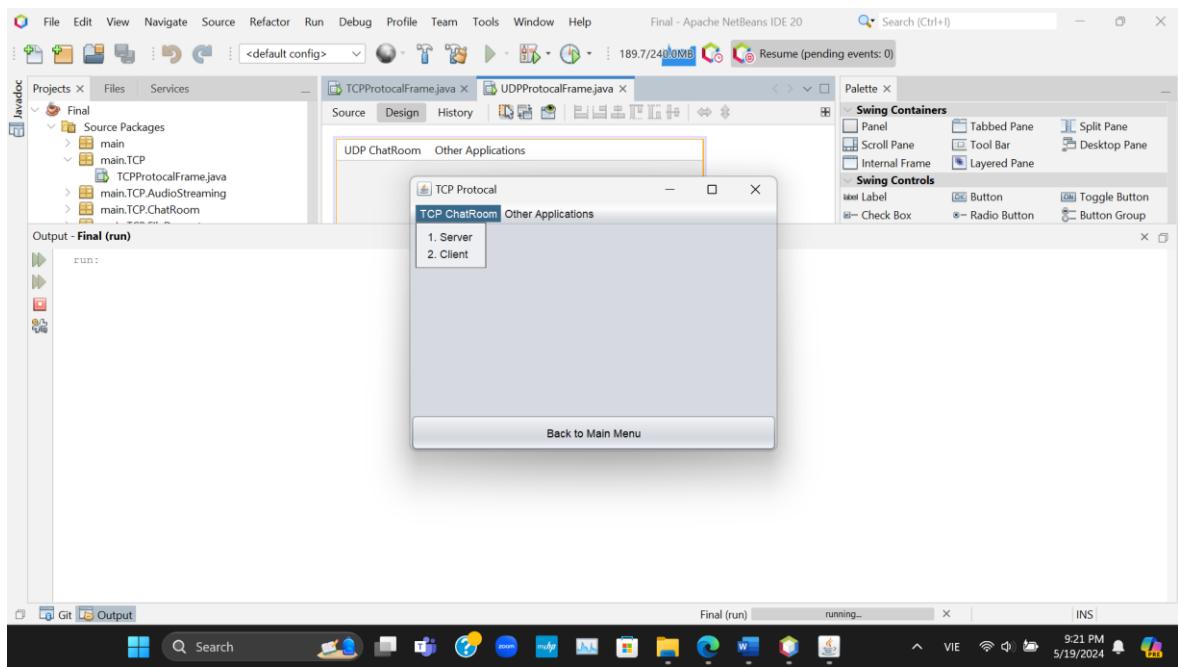
Cách 2: Nhập project Final vào IDE Netbeans và chạy chương trình ở môi trường develop.

Bước 2: Thực hiện các chức năng của chương trình.

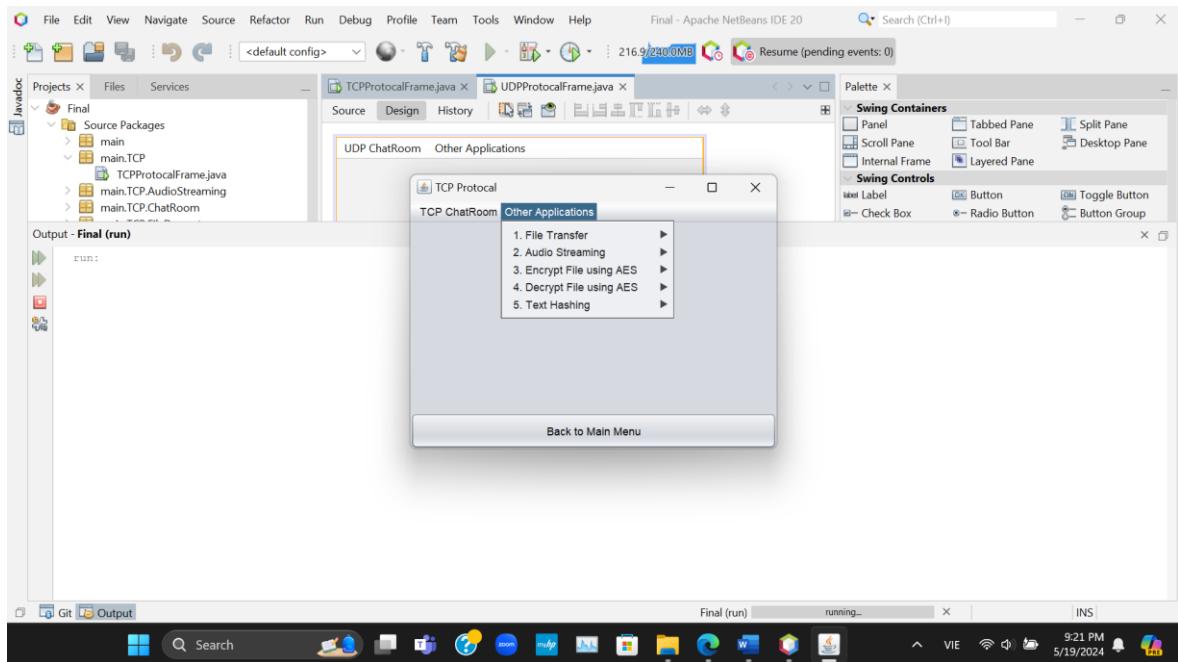
3.1.3.2 Kết quả test chương trình



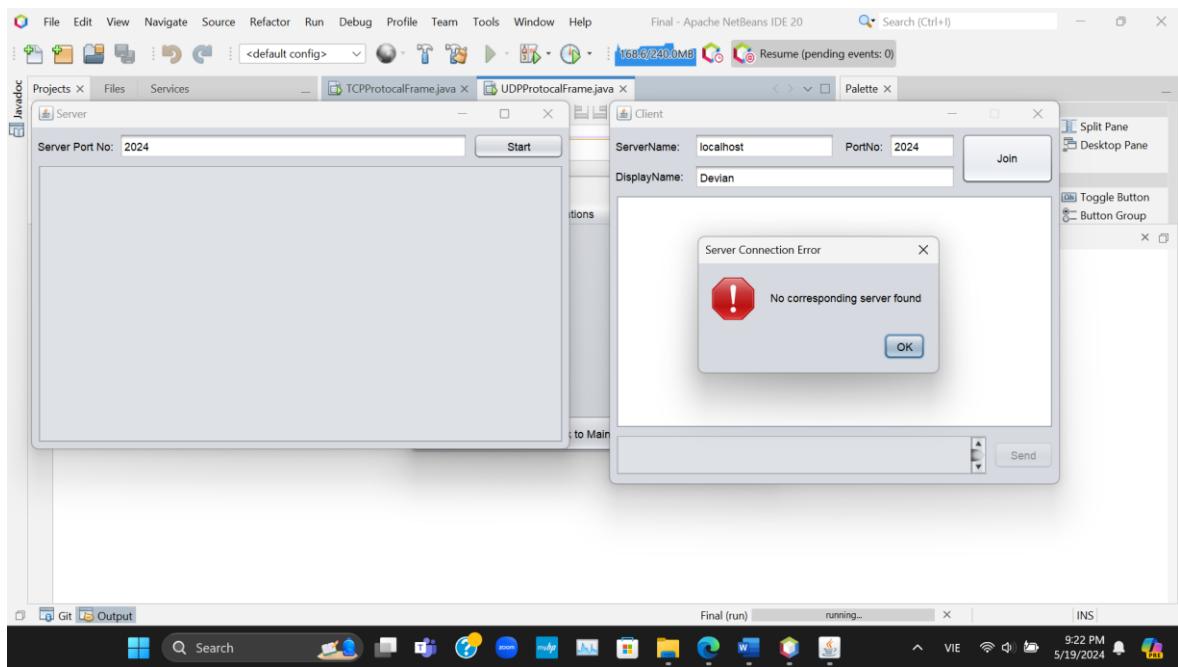
Giao diện Main Menu



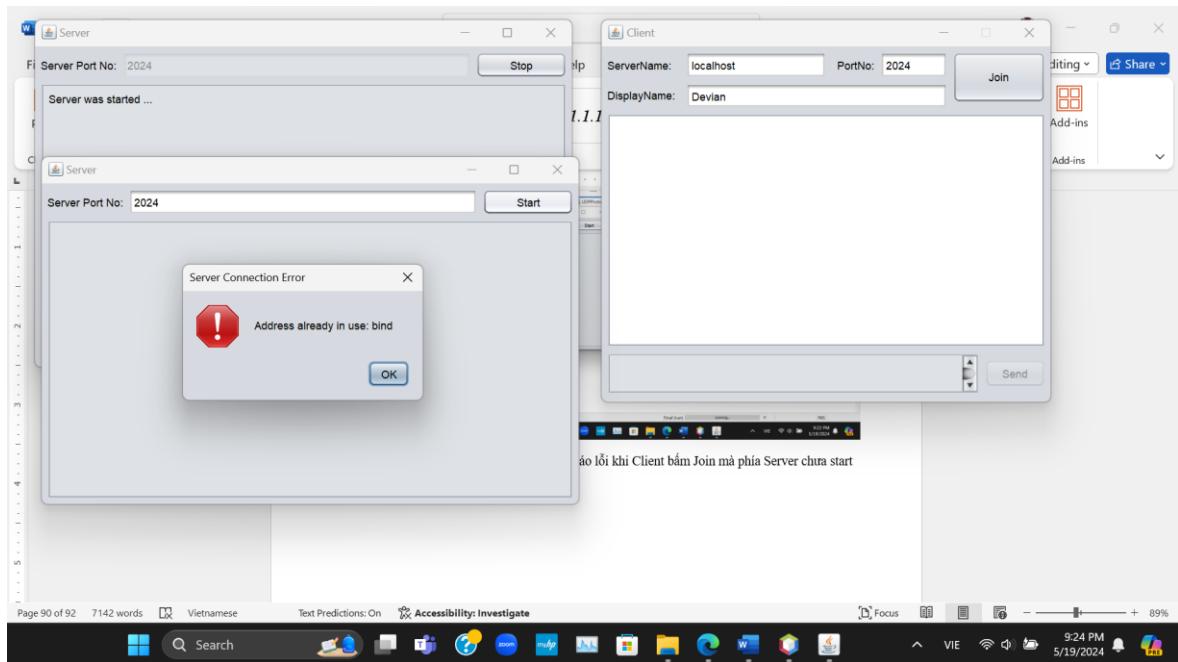
TCP Protocol Menu Multi Chat Room



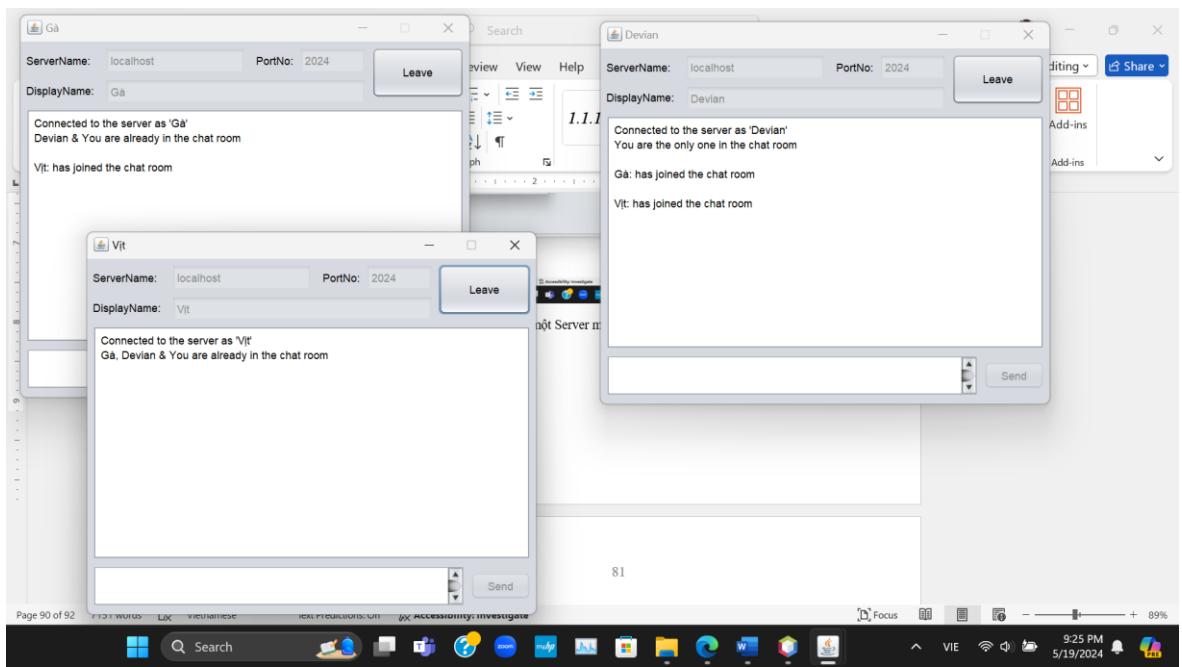
TCP Protocol Menu Other Applications



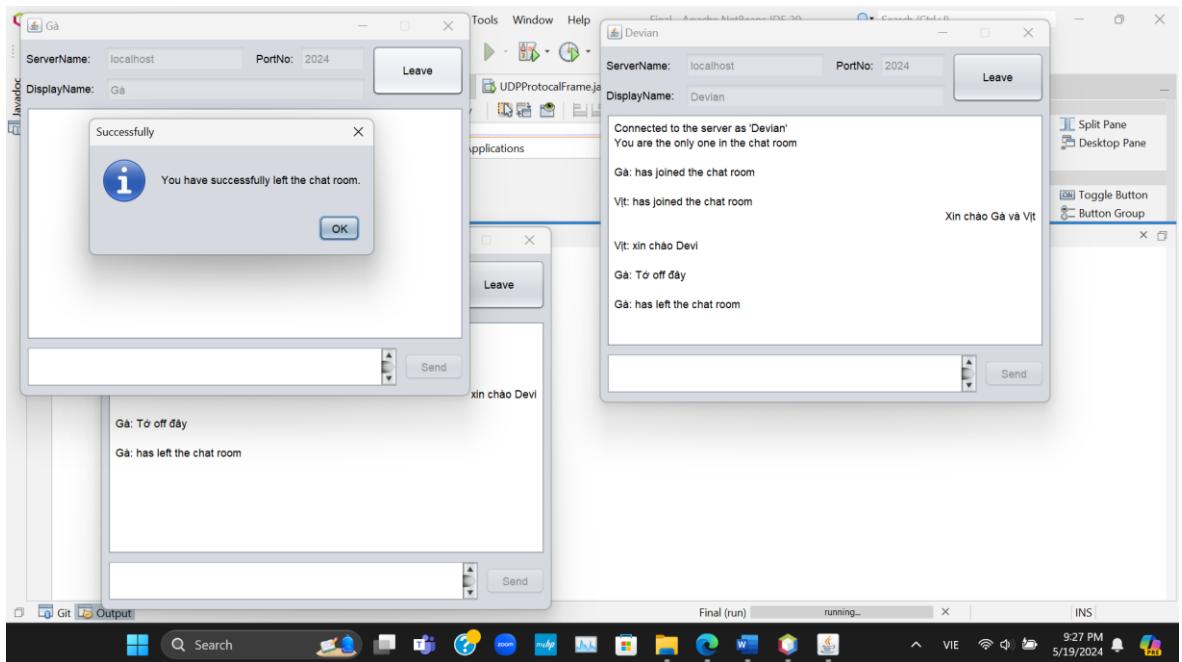
Ứng dụng TCP Multi Chat Room; Báo lỗi khi Client bấm Join mà phía Server chưa start



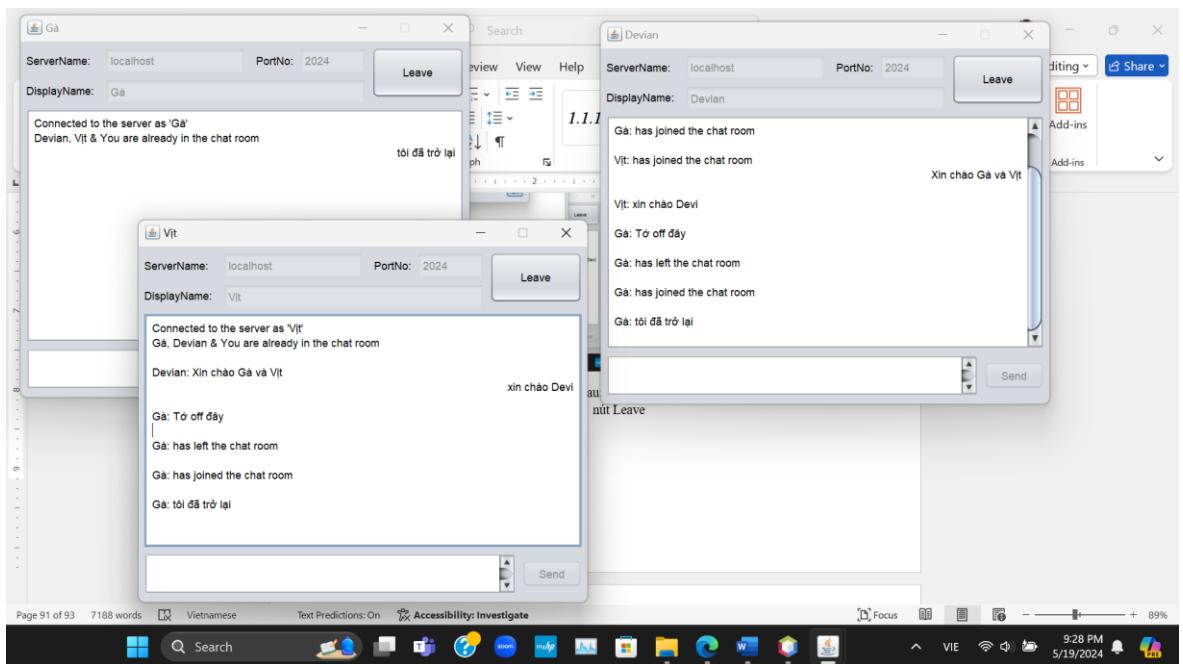
Start một Server mới nhưng port đã sử dụng



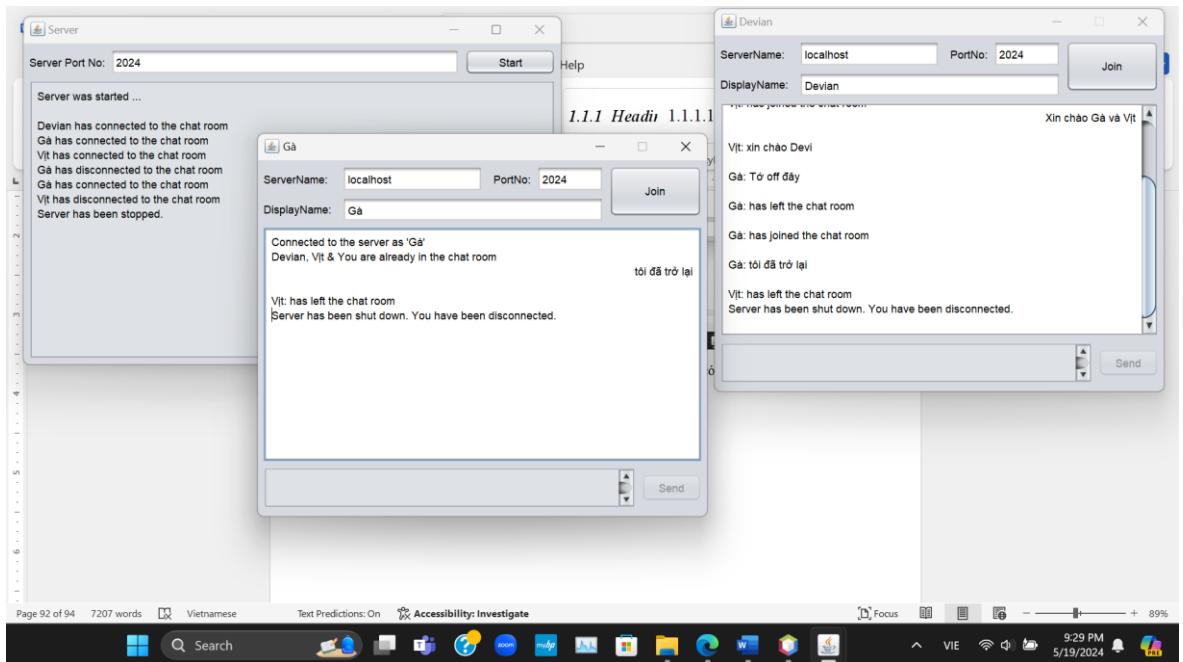
3 Client kết nối đến Server port 2024; Hiển thị các thông tin cần thiết cho Client



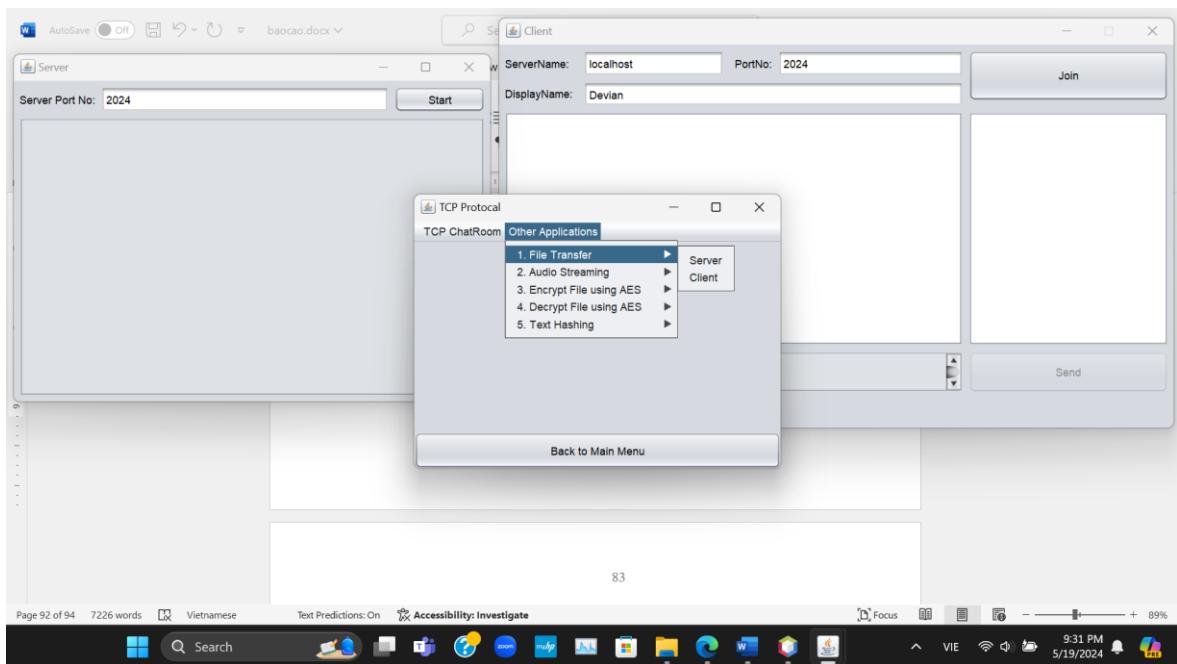
Hiển thị tin nhắn giữa các client với nhau; Thông báo Rời khỏi phòng khi một Client nhấp nút Leave



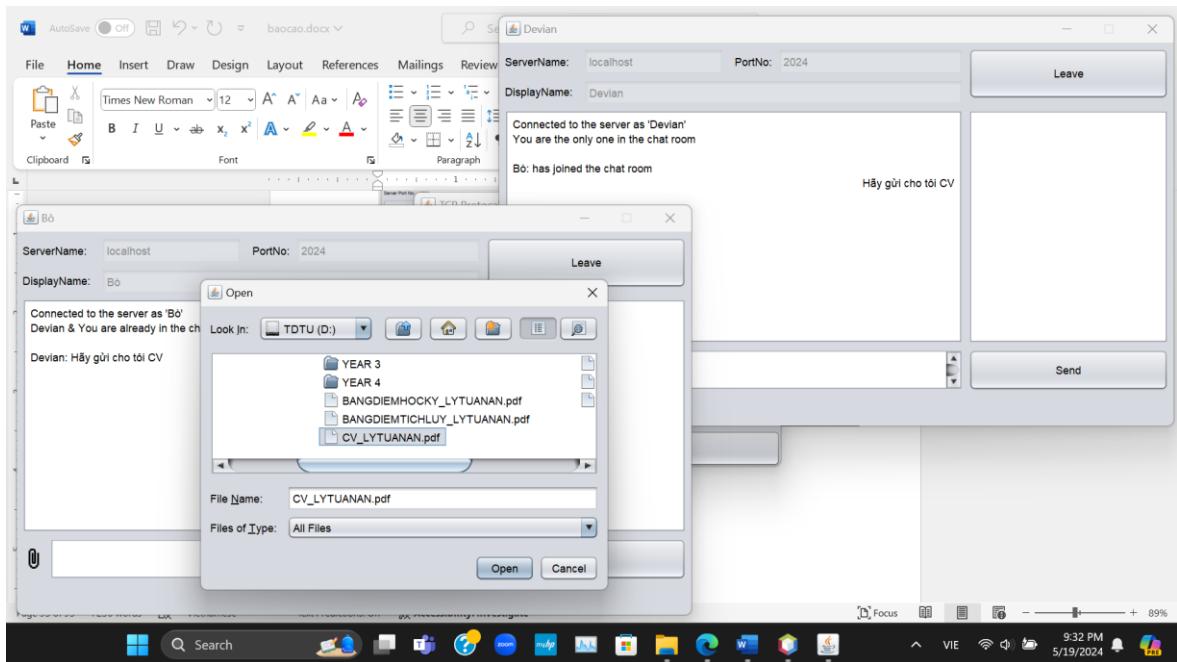
Khi join trở lại cũng sẽ thông báo cho các client khác và có thể tiếp tục trò chuyện



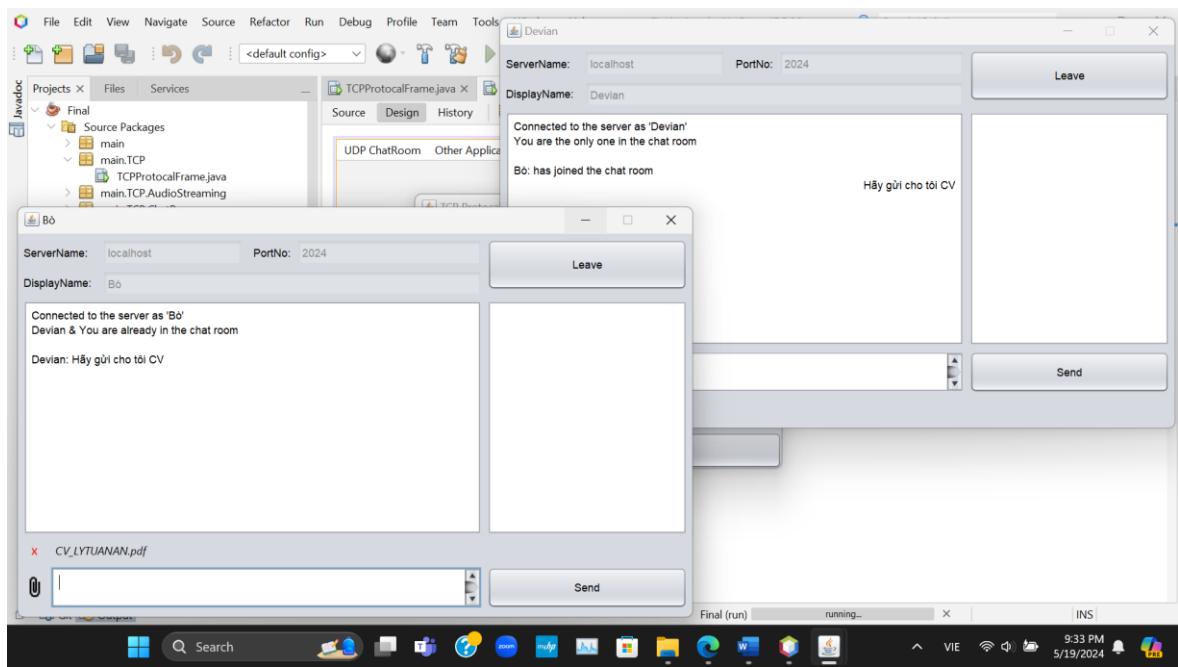
Khi stop Server thì cũng sẽ thông báo đến tất cả Client; Đồng thời ngắt kết nối mọi người



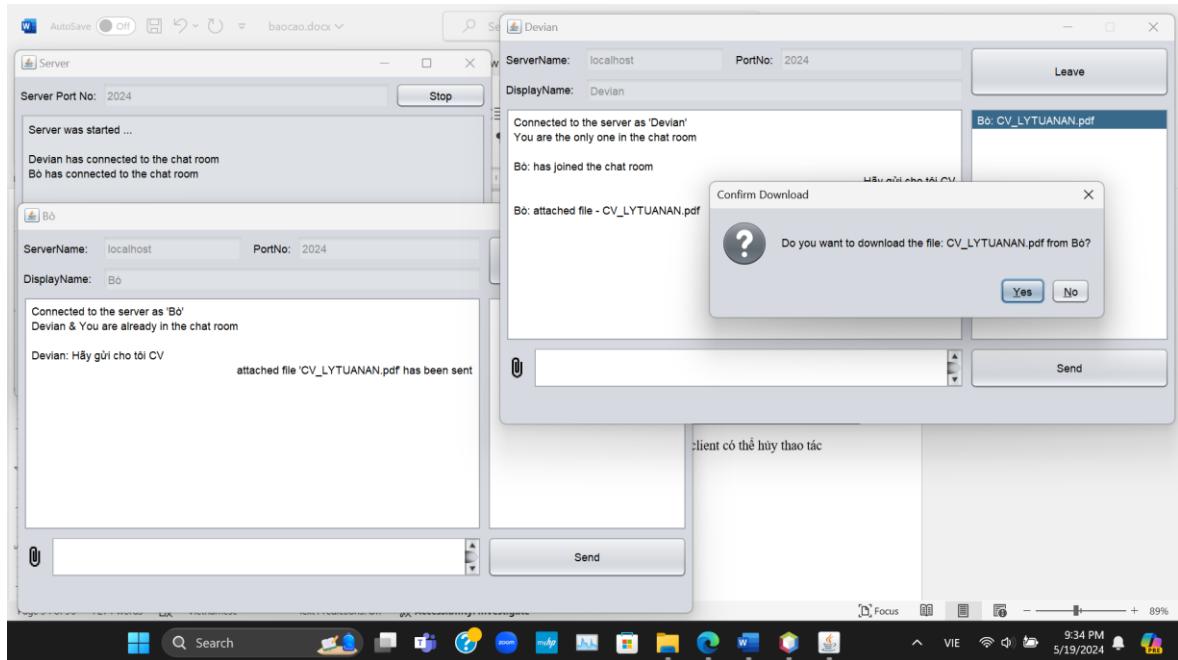
Ứng dụng TCP File Transfer (Truyền file)



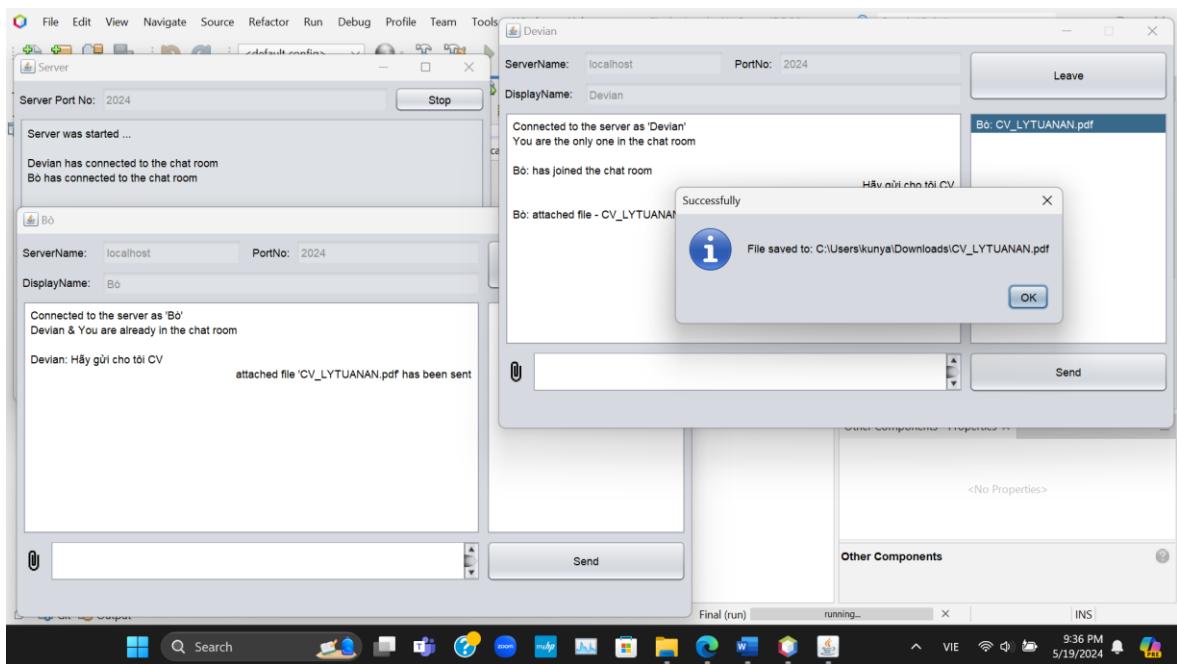
Tương tự ứng dụng Multi Chat Room nhưng bổ sung thêm tính năng chọn và gửi file cho các client trong room



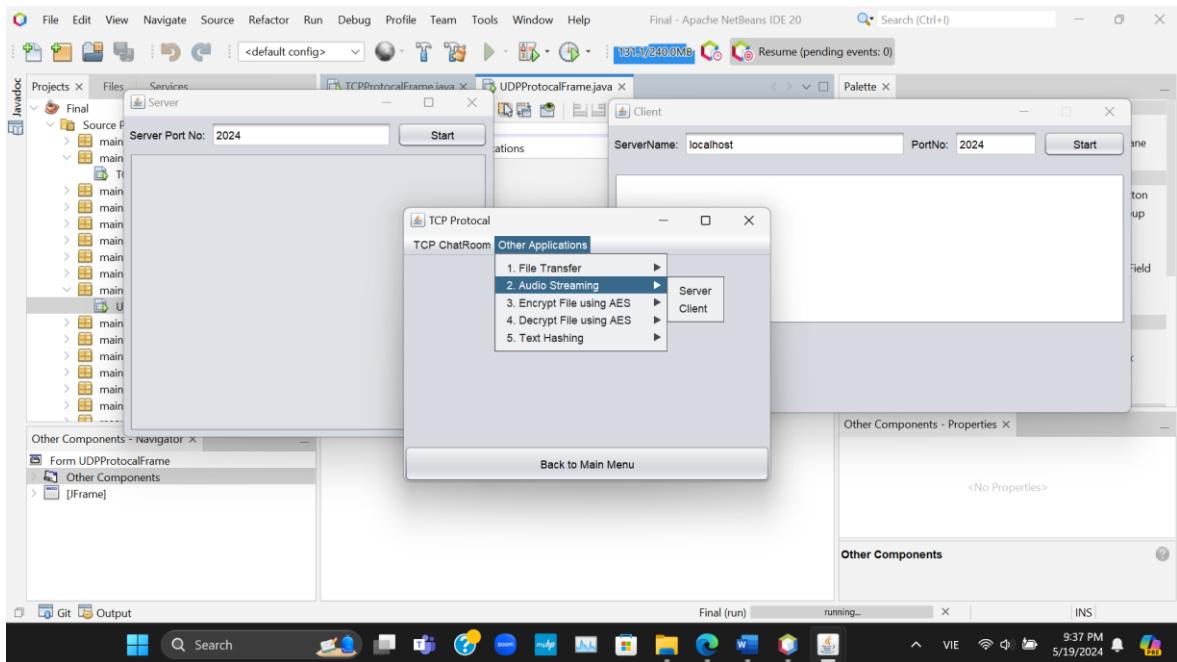
Chọn file thành công sẽ hiển thị tên file và nút x để client có thể hủy thao tác



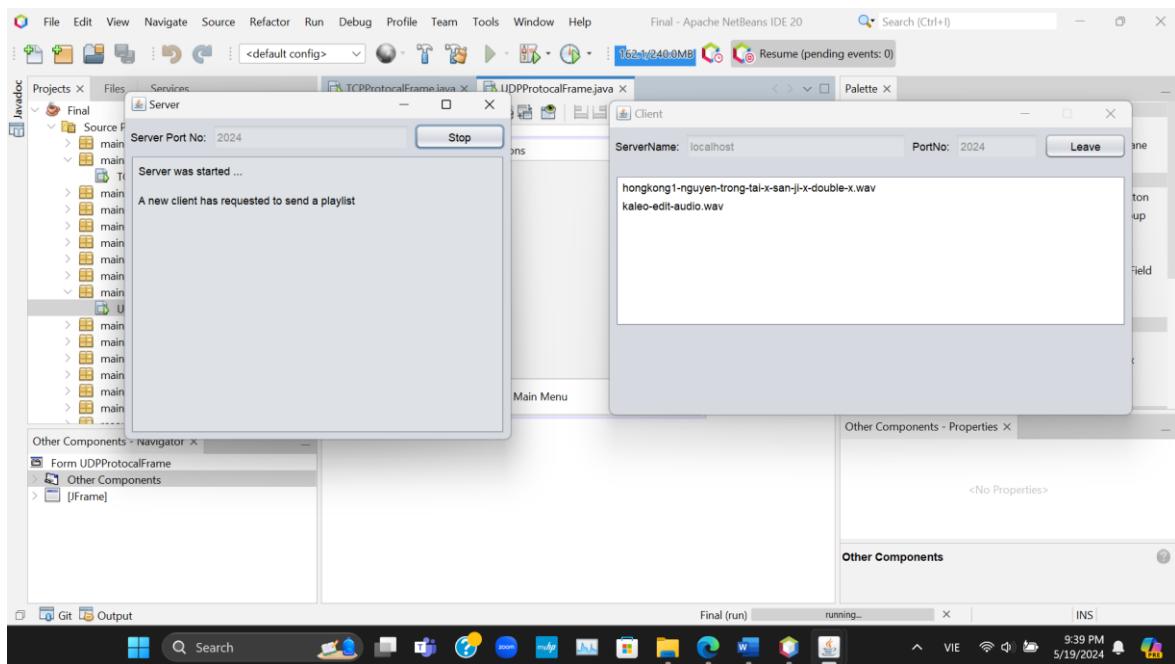
Phía client khác sẽ nhận được file và khi click vào item tương ứng trên JList sẽ có thể thực hiện download file



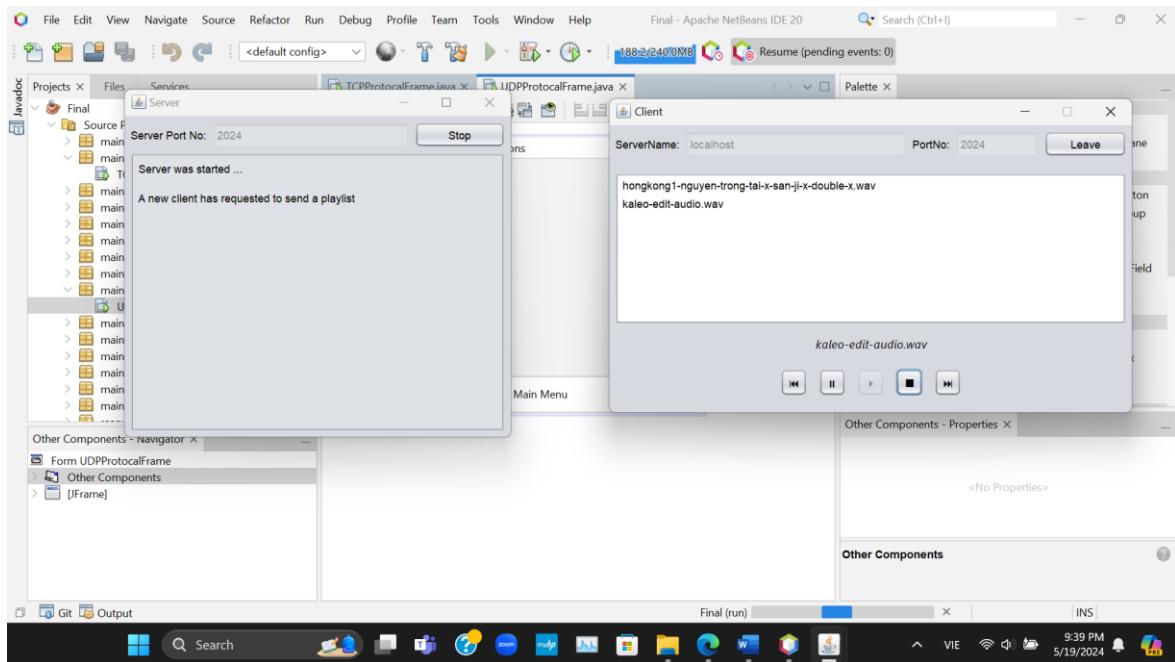
Download thành công vào thư mục Downloads của máy tính



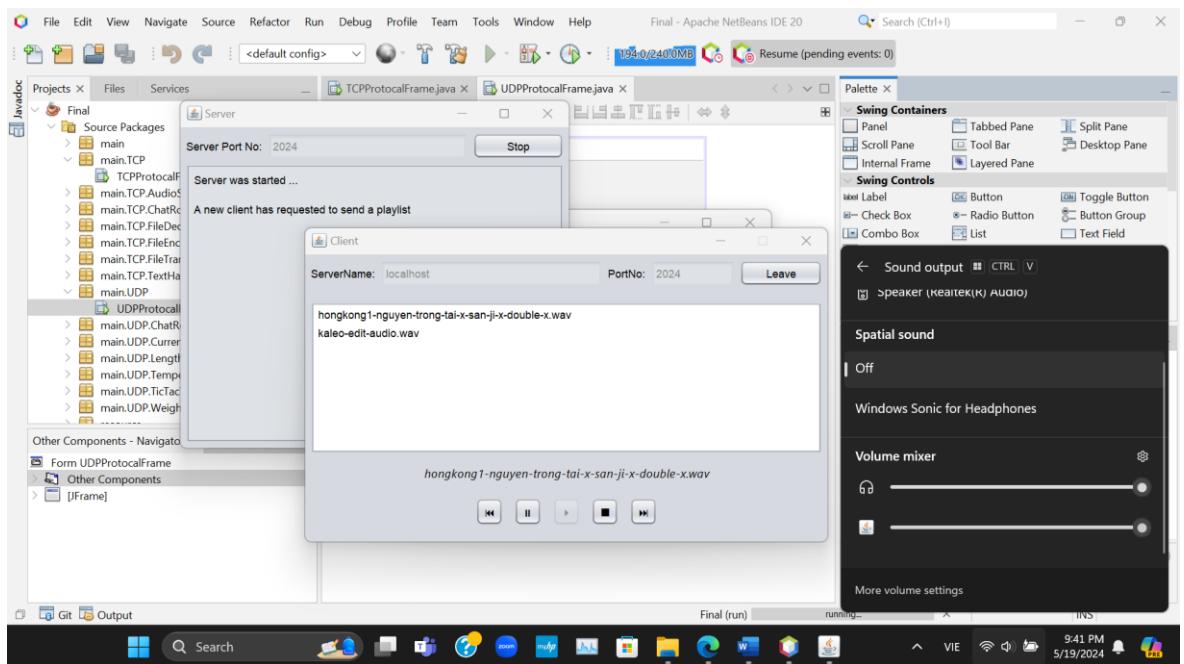
Ứng dụng TCP Audio Streaming (Phát thanh, phát nhạc,...)



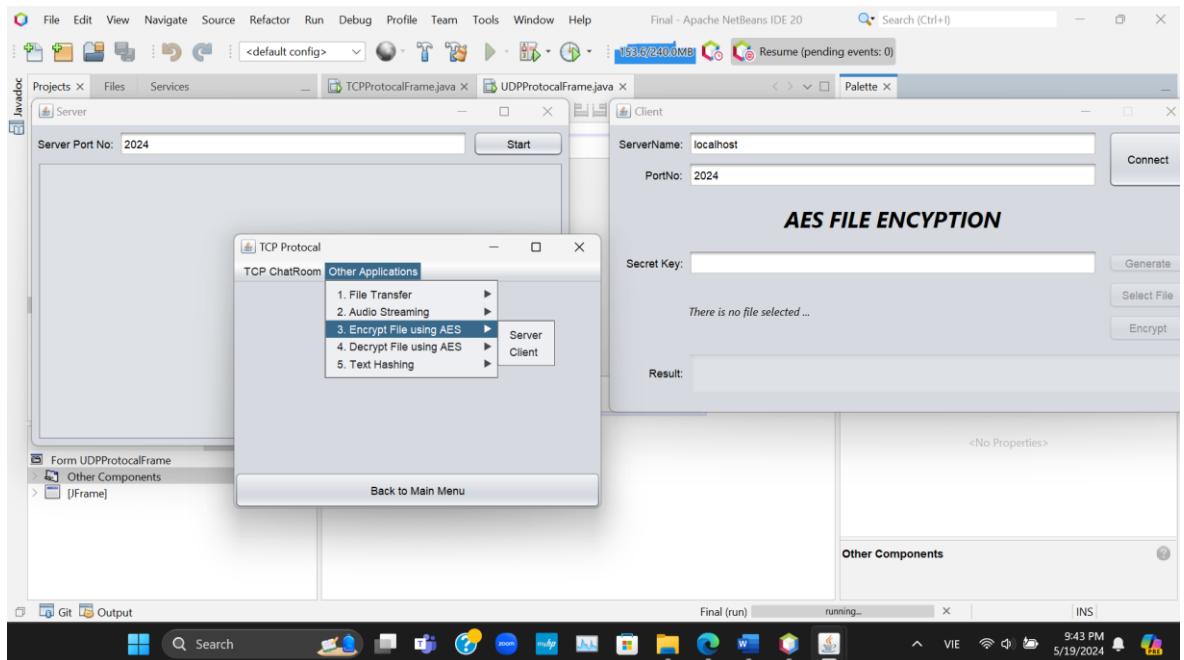
Client kết nối thành công sẽ nhận được danh sách bài hát hiện có từ phía Server



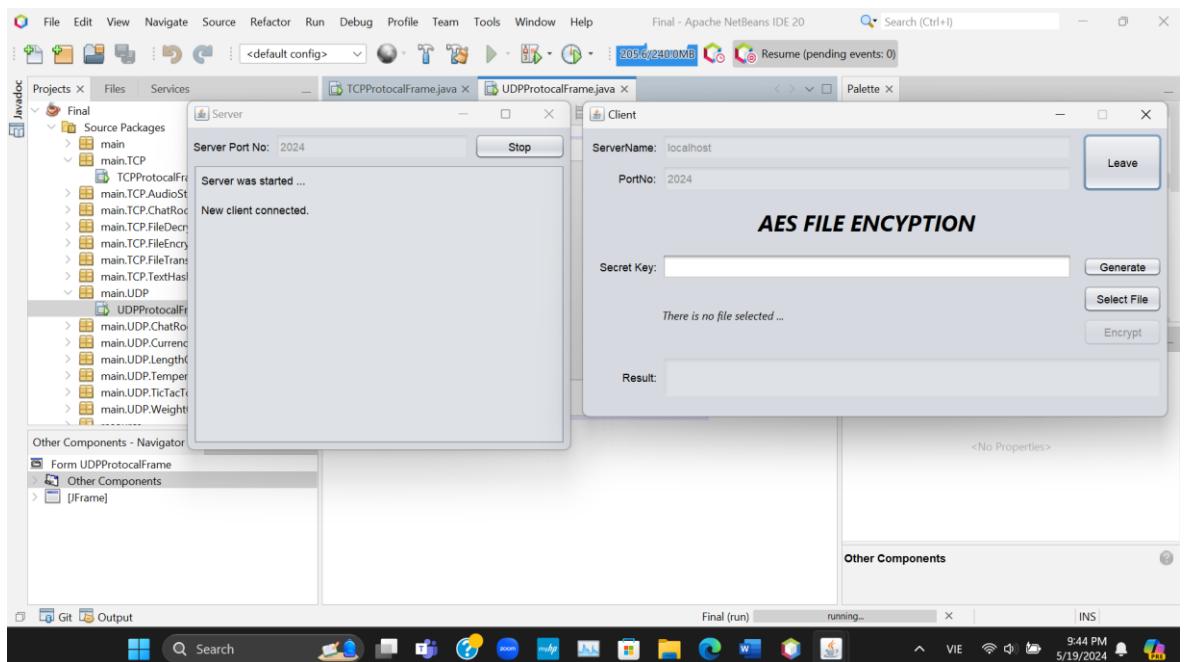
Chọn bài hát và nhấn nút play sẽ nhận được truyền tải dữ liệu của file âm thanh từ phía Server



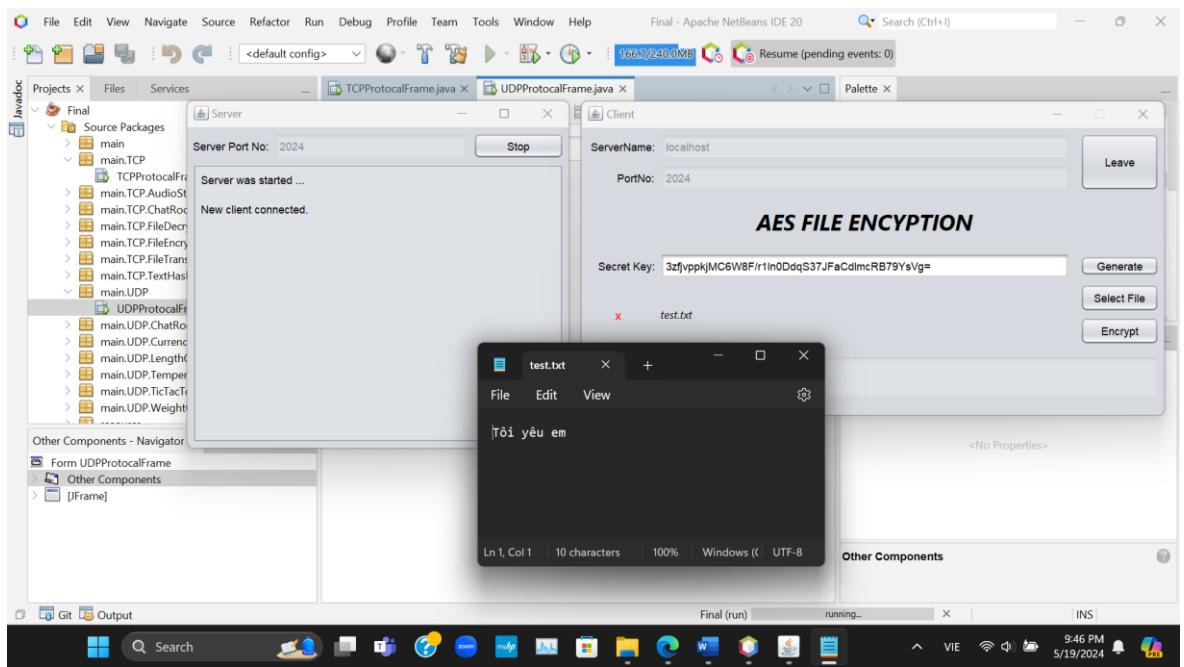
Thanh tăng giảm âm lượng hiện thị trên máy tính khi phát nhạc



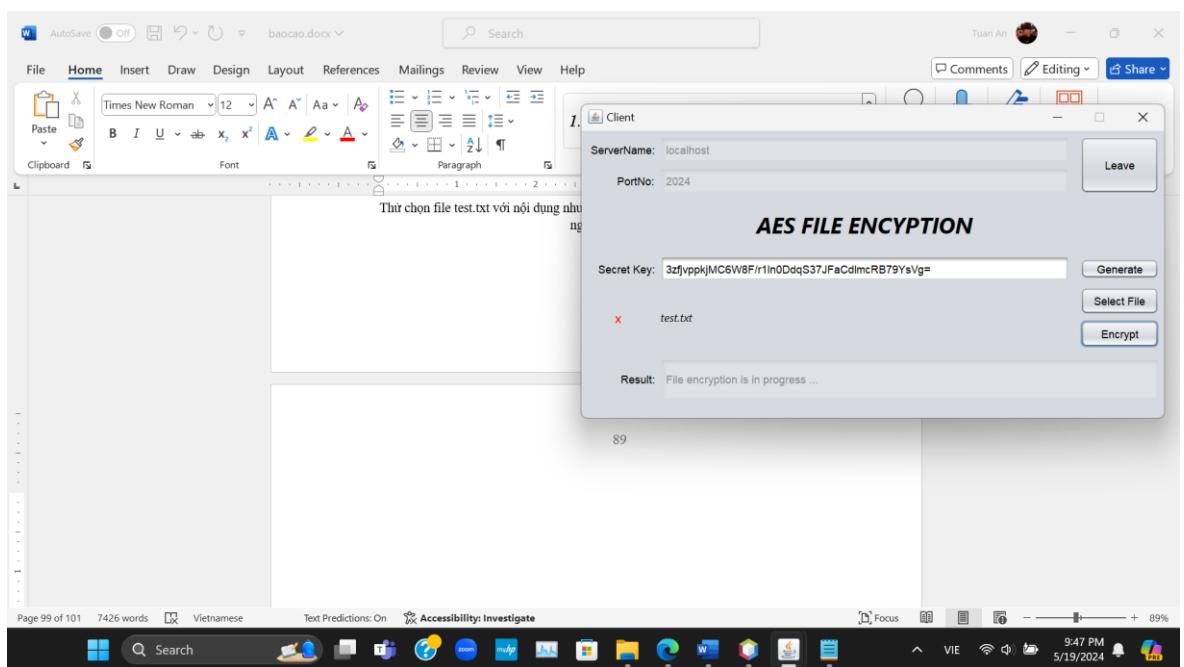
Ứng dụng TCP Encrypt File using AES



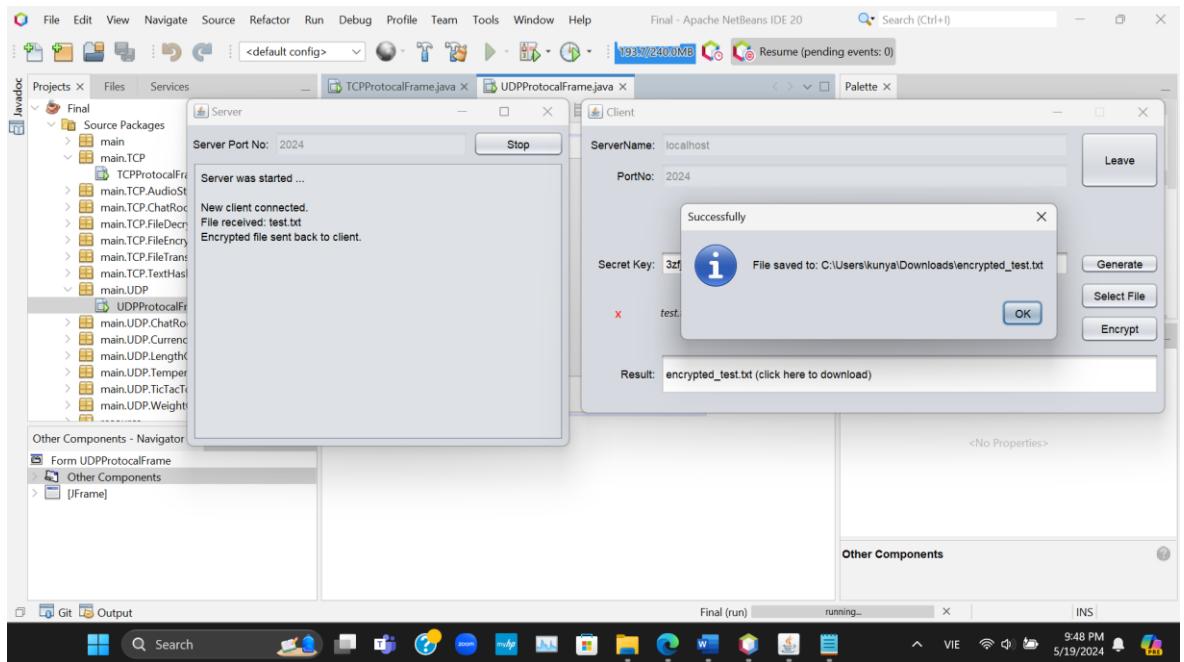
Kết nối đến Server thành công nhưng chưa tạo khóa bí mật và chọn file thì nút gửi dữ liệu Encrypt sẽ bị vô hiệu hóa



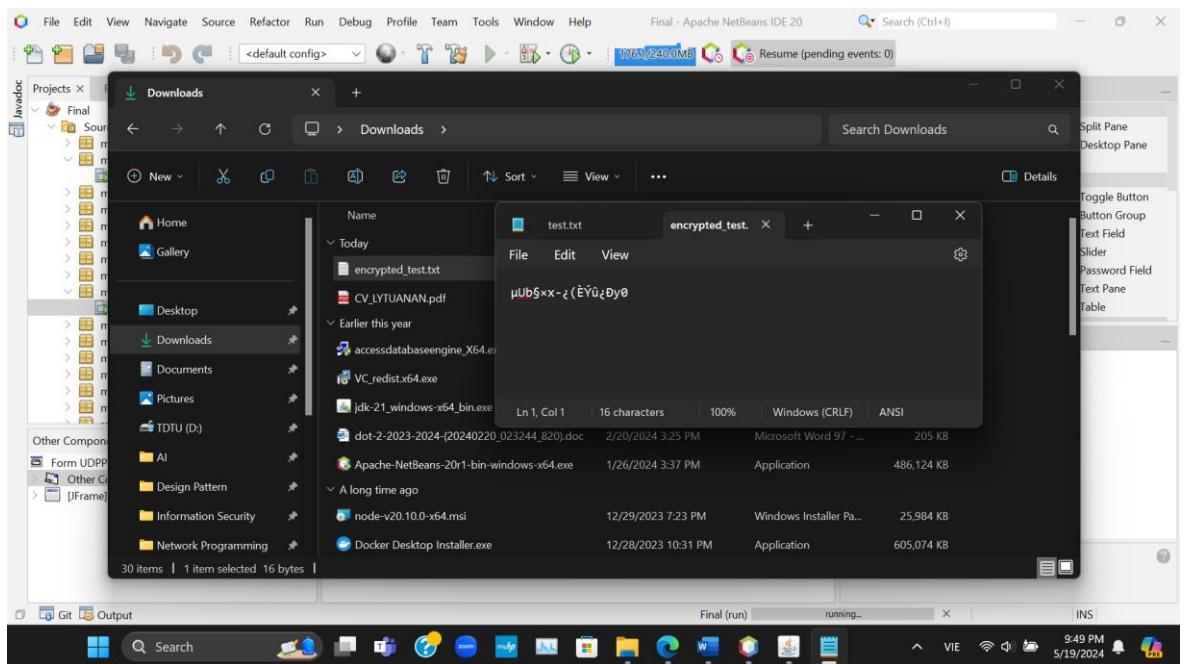
Thứ chọn file test.txt với nội dung như trên để mã hóa; Và lưu lại khóa bí mật để có thể thử nghiệm ở phần sau



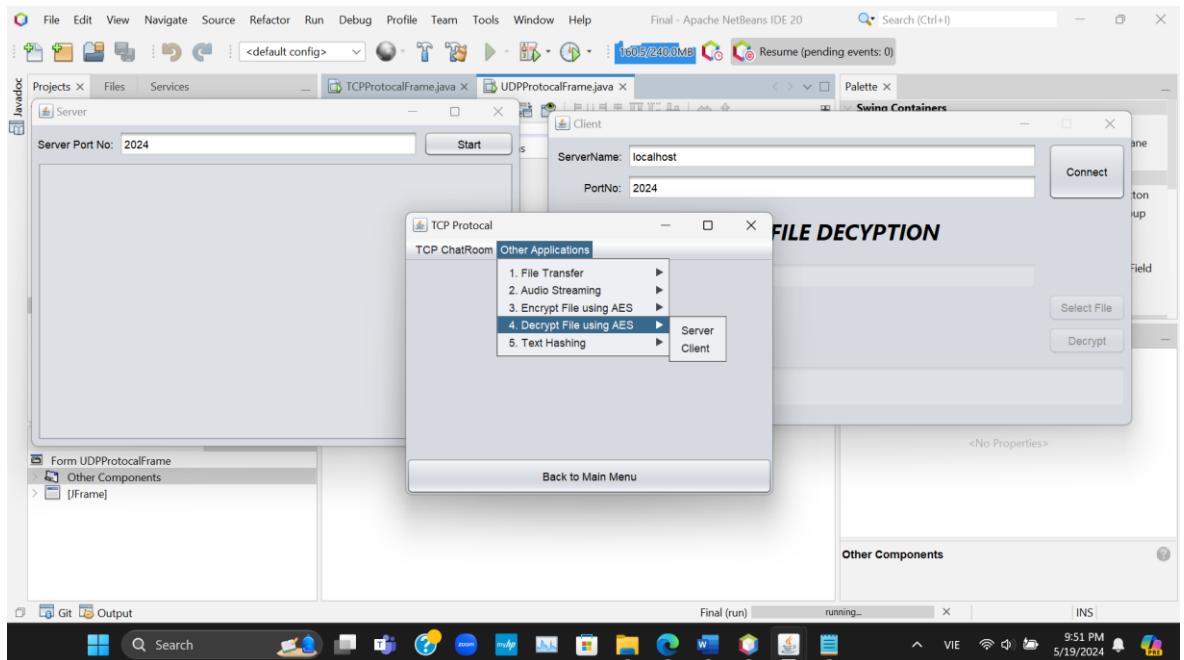
Sau khi nhấn nút Encrypt sẽ cập nhật giao diện phía Client



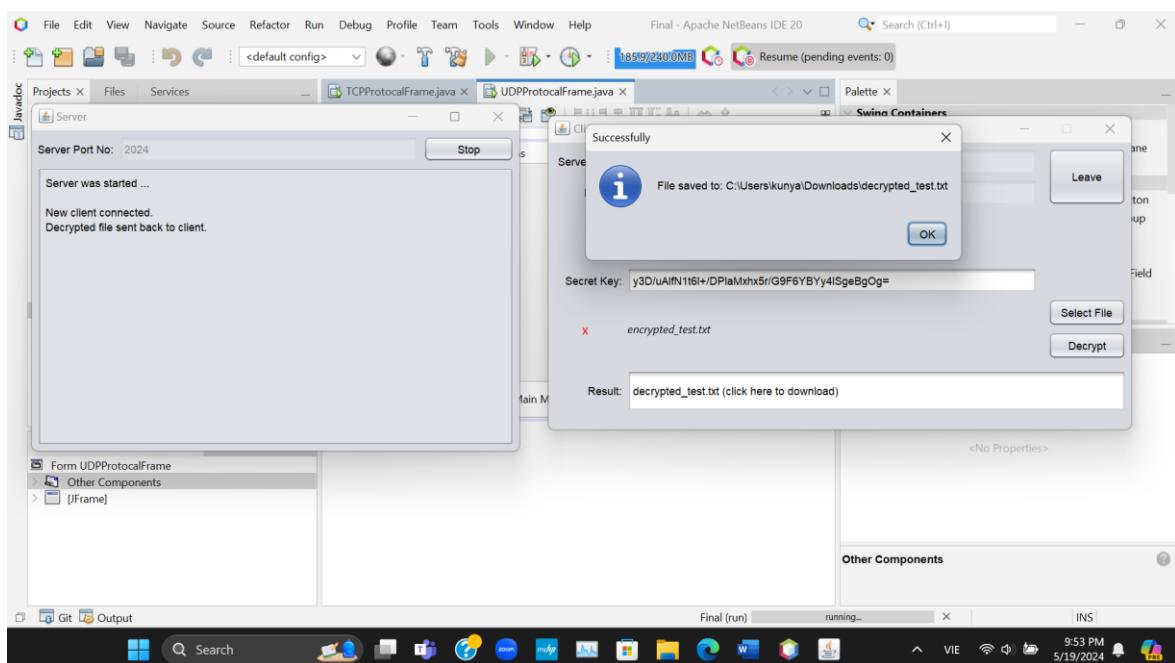
Sau khi đã nhận được file mã hóa từ Server có thể click vào ô Result để thực hiện download file.



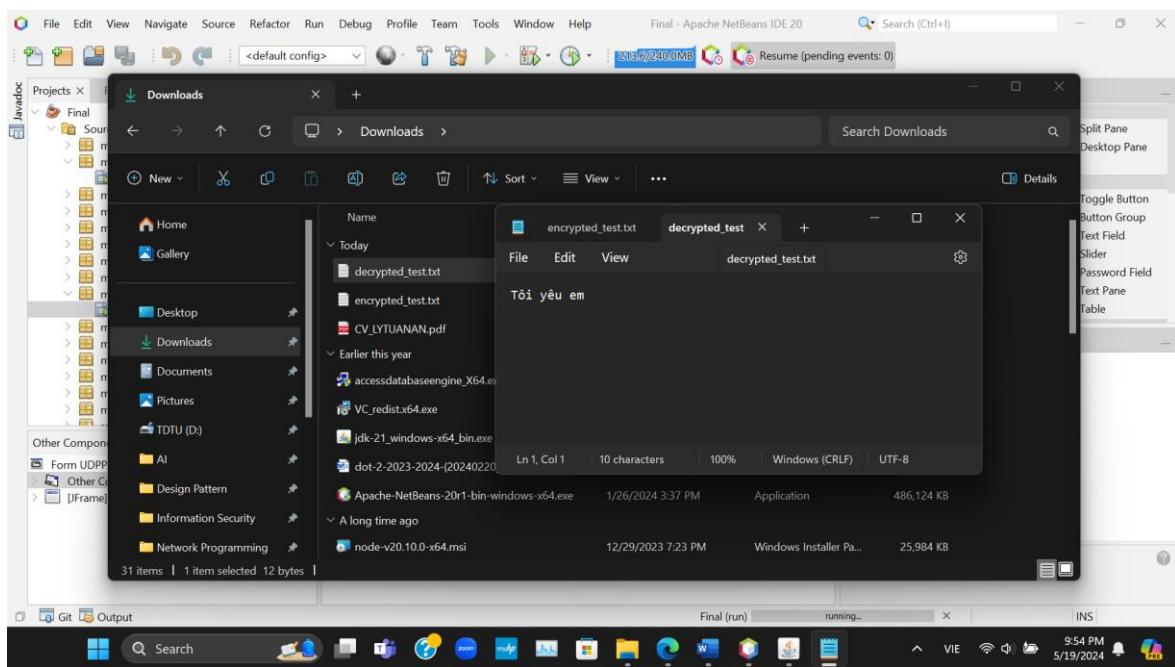
Nội dung file đã được mã hóa



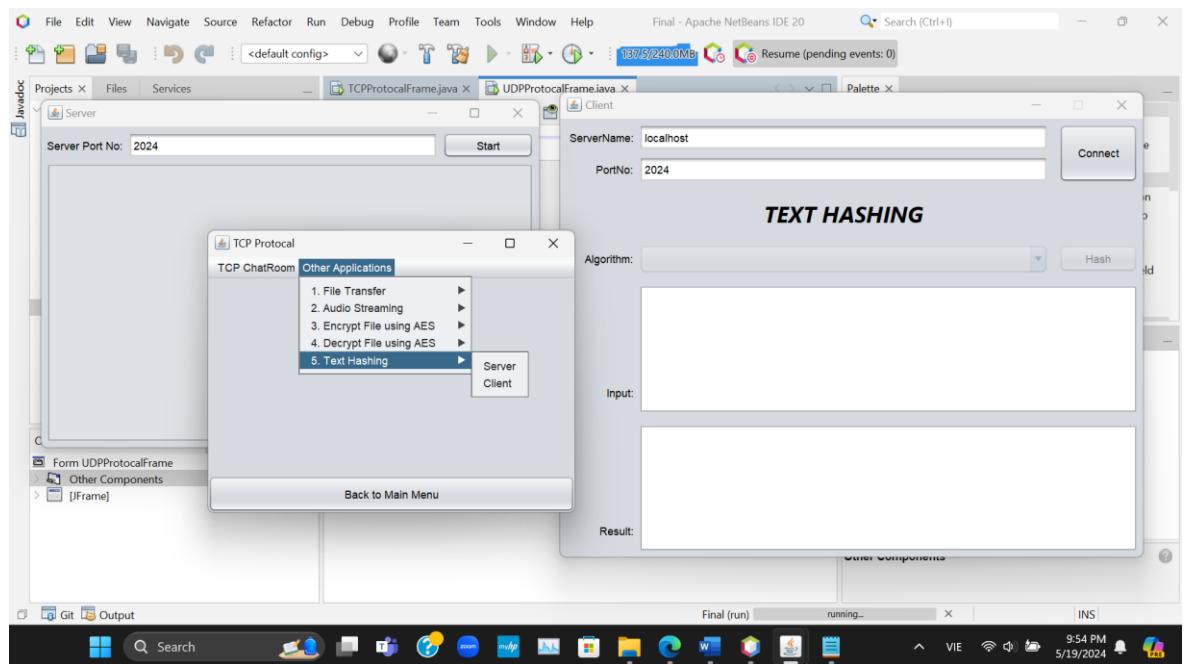
Ứng dụng TCP Decrypt File using AES; Tương tự phần mã hóa, trong ứng dụng này sẽ thực hiện gửi cho Server khóa bí mật và file bị mã hóa để thực hiện giải mã



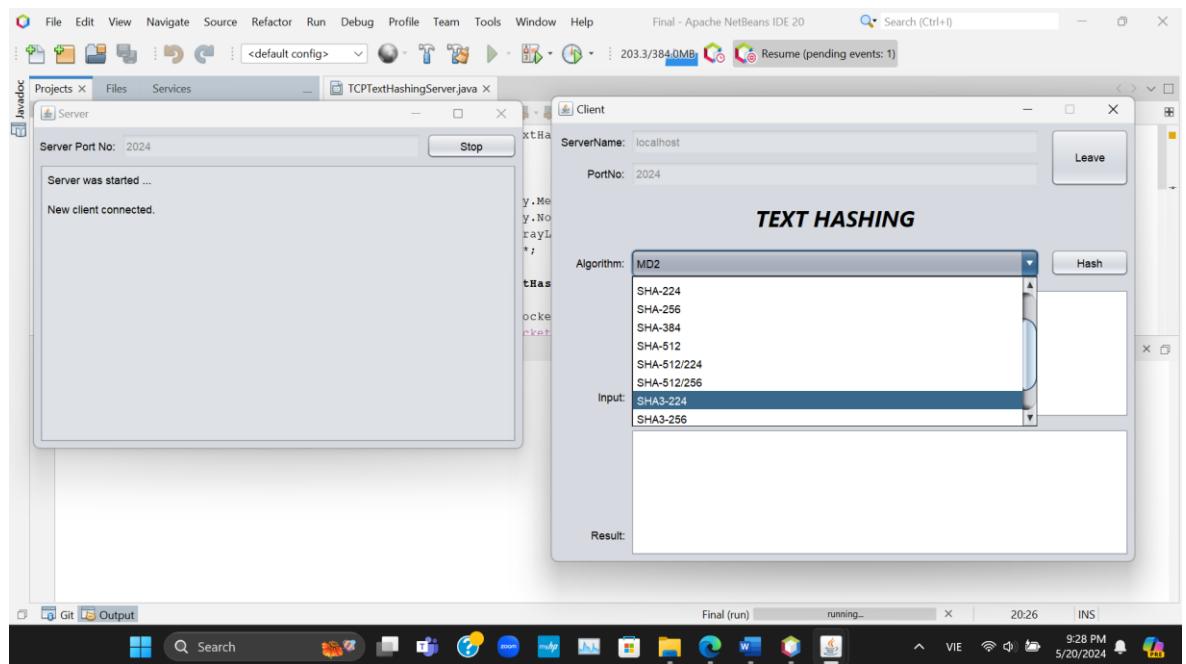
Thực hiện gửi khóa bí mật khi này và file bị mã hóa; Sau đó download file đã giải mã



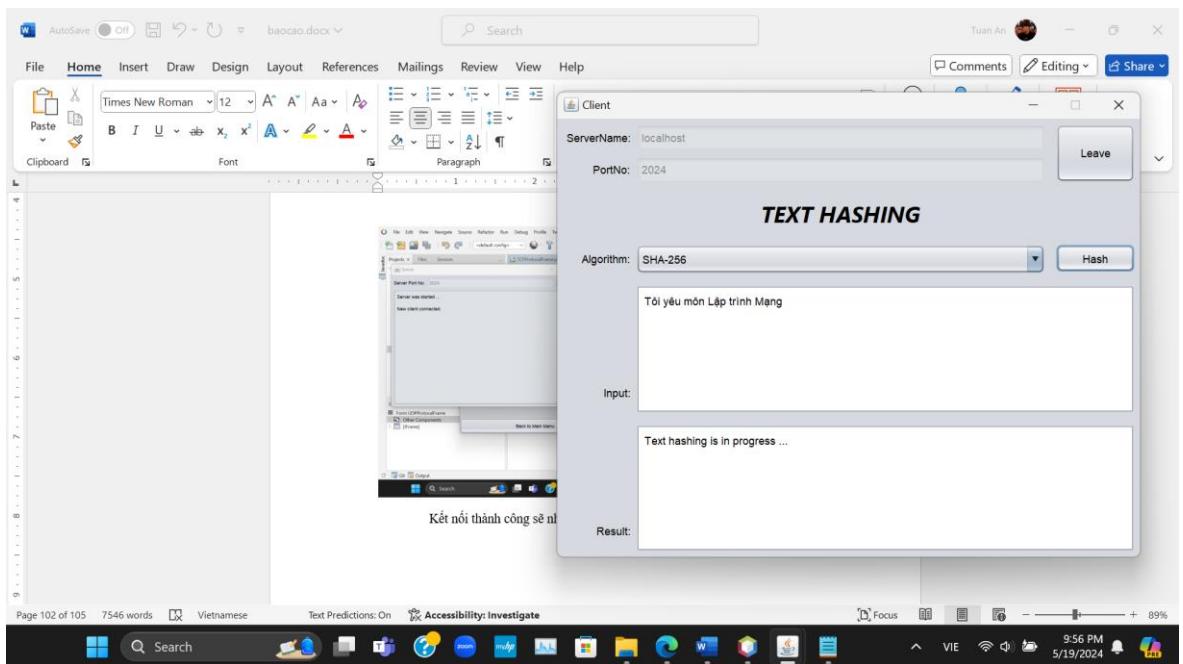
File đã được giải mã



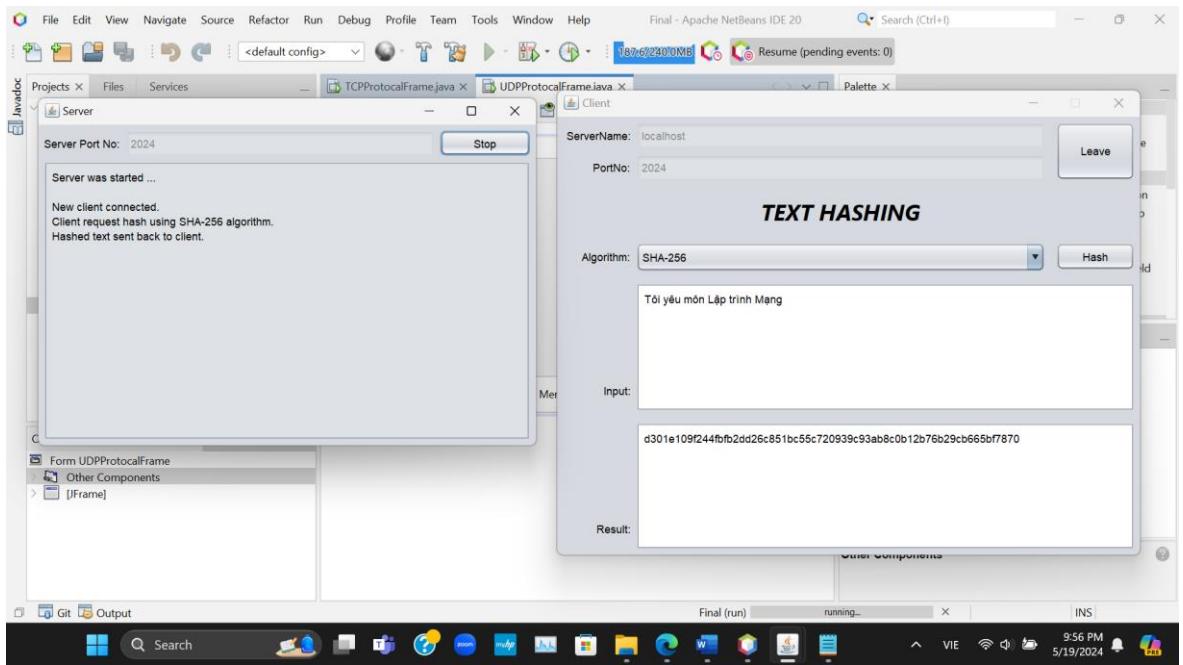
Ứng dụng TCP Text Hashing



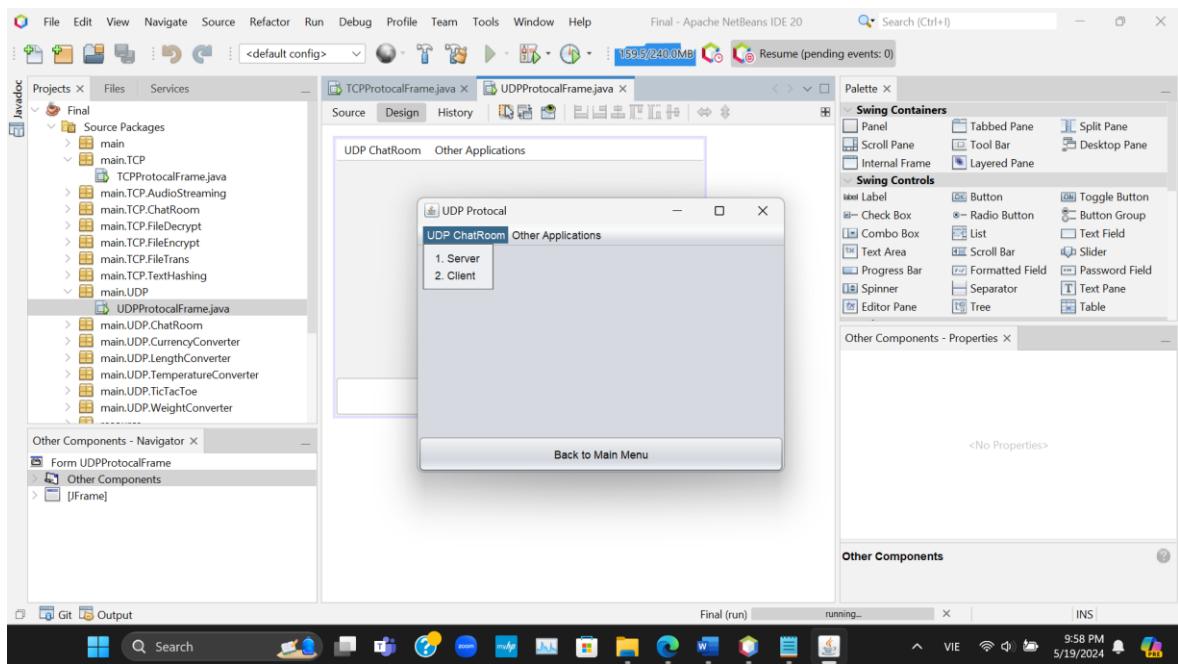
Kết nối thành công sẽ nhận được danh sách thuật toán mà Server hỗ trợ;



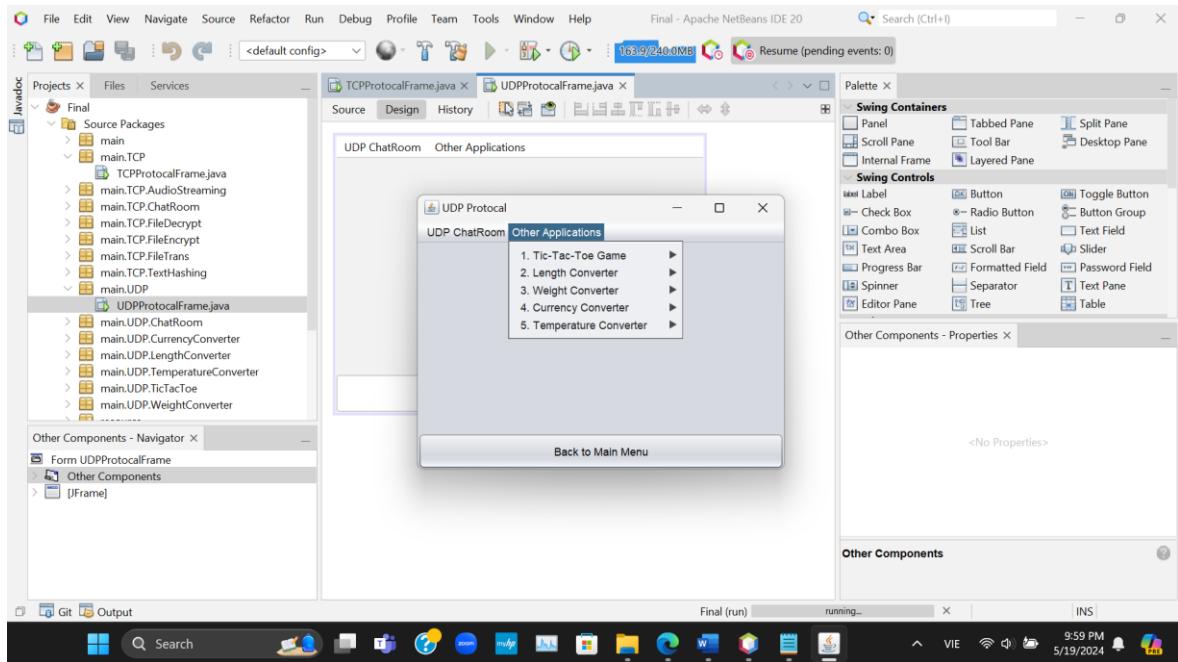
Thực hiện hash đoạn văn bản; Và chờ kết quả từ phía Server



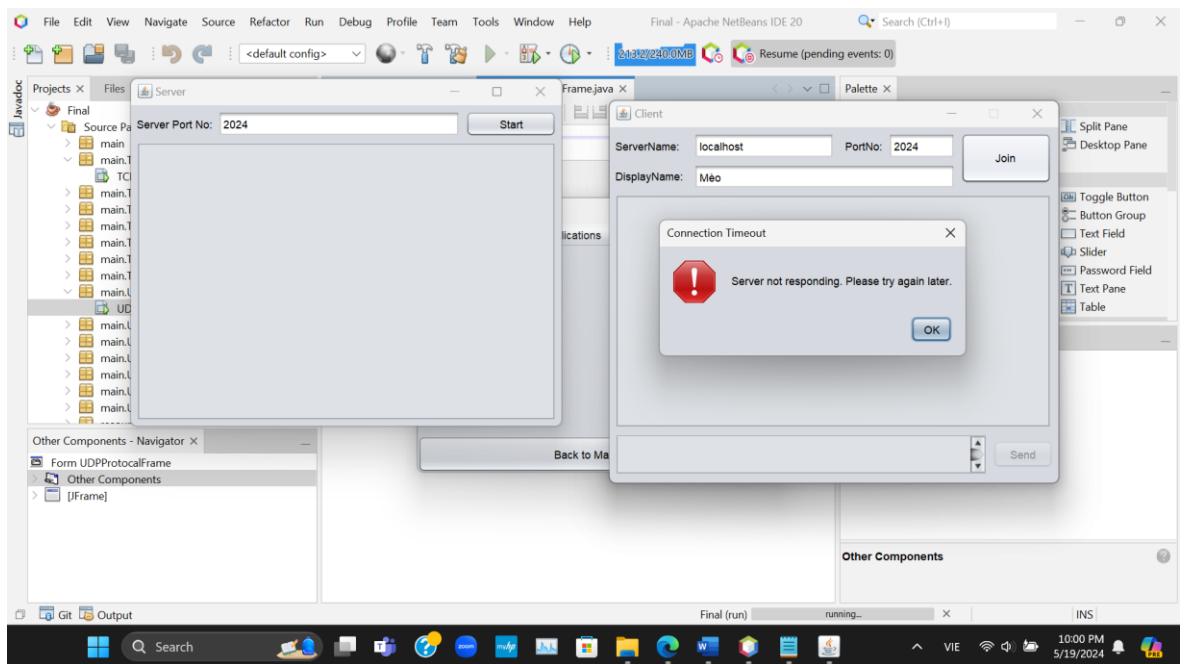
Nhận được kết quả sau khi hash đoạn văn bản



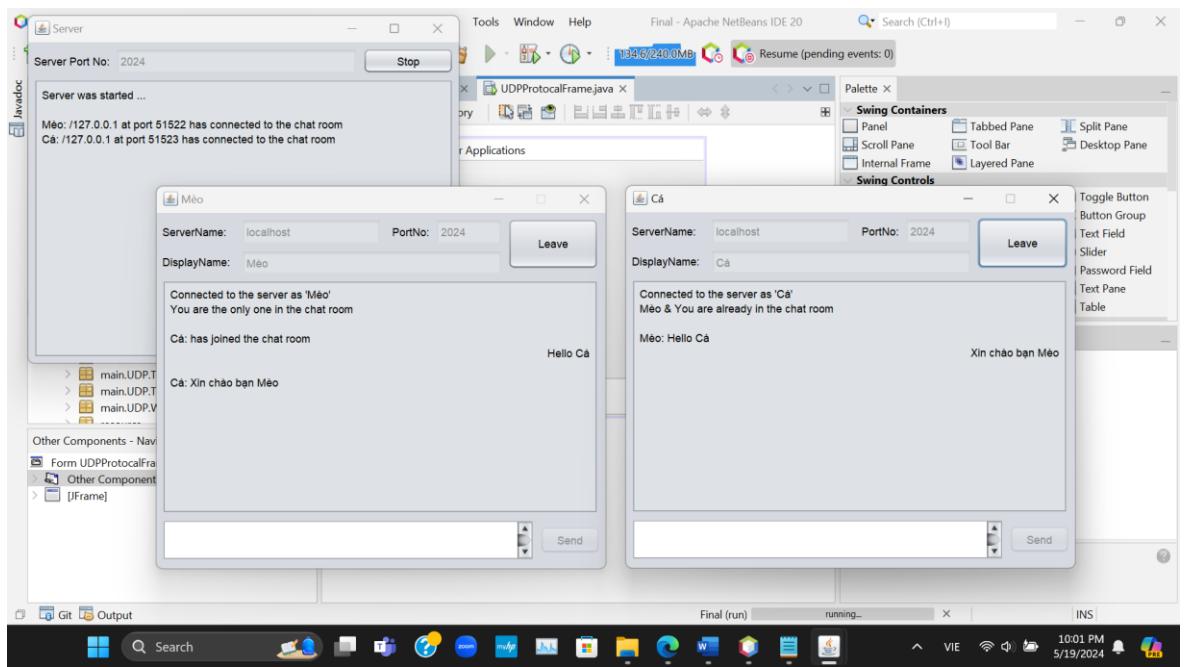
UDP Protocol Menu Multi Chat Room



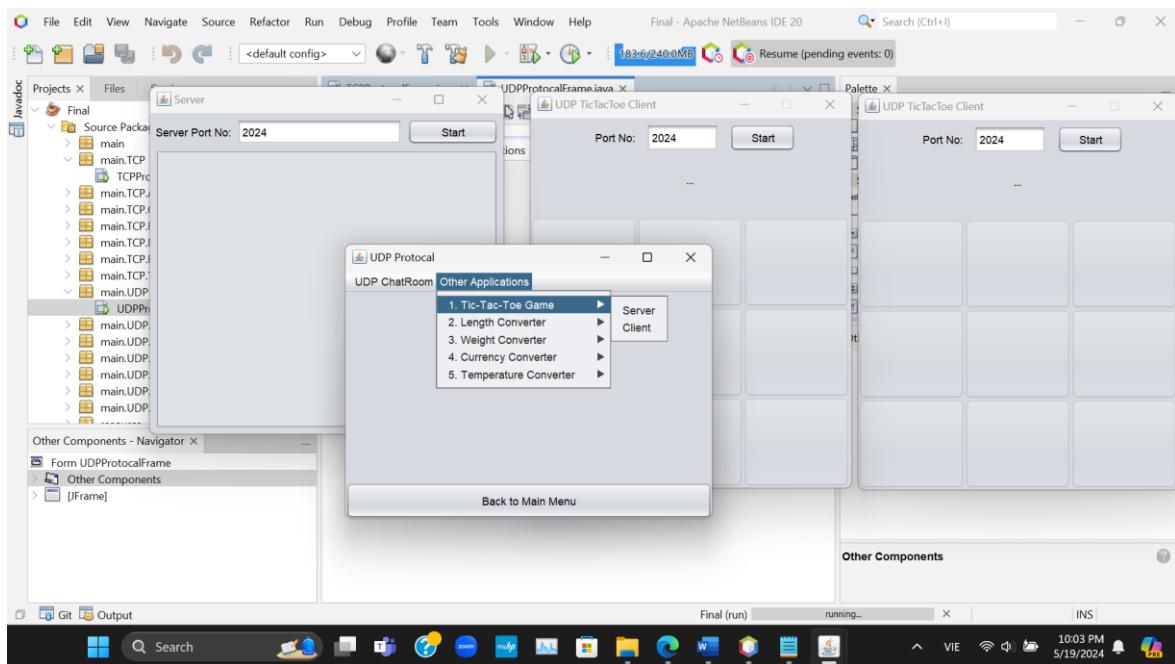
UDP Protocol Menu Other Applications



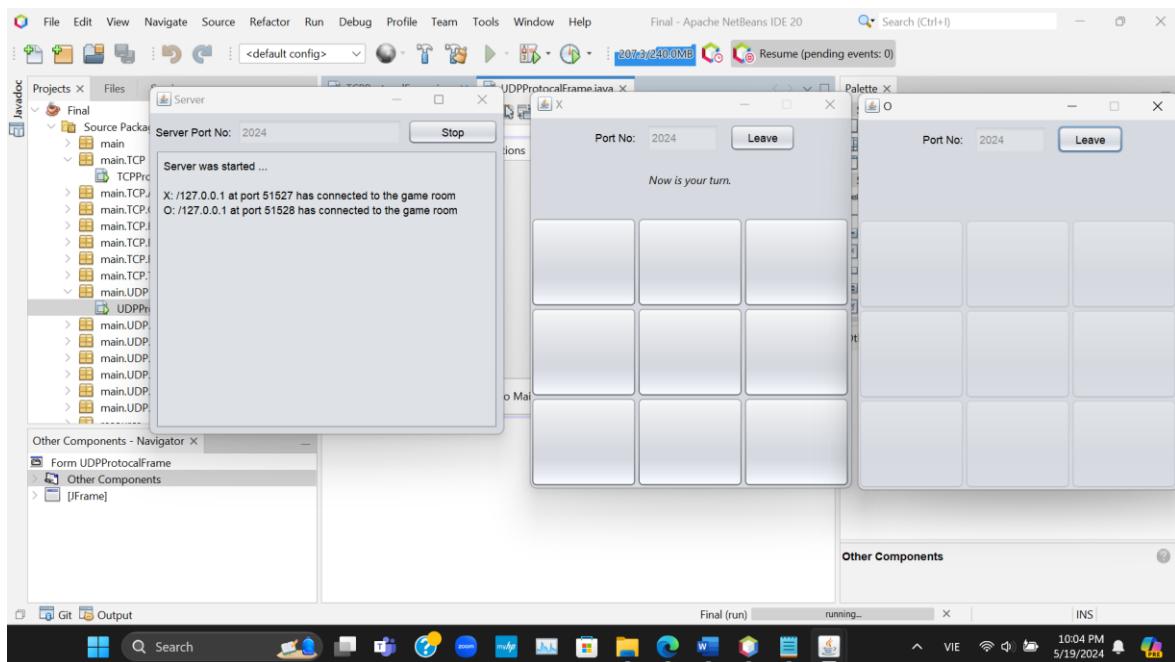
Ứng dụng UDP Multi Chat Room; Báo lỗi khi Client bấm Join mà phía Server chưa start do không có phản hồi



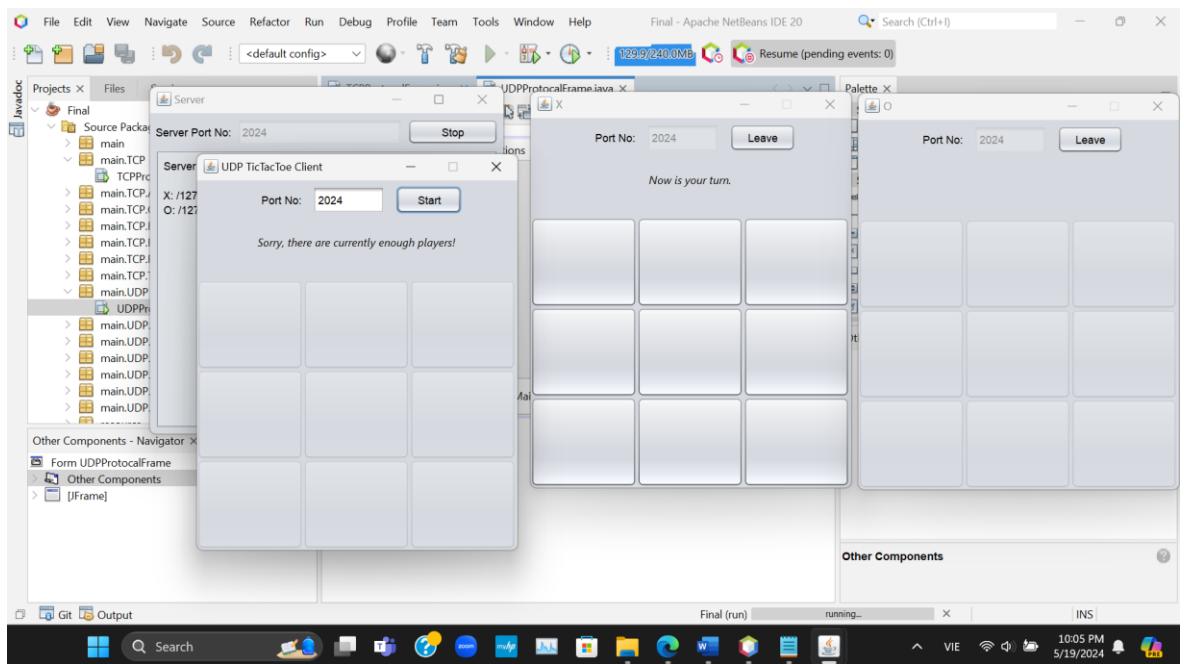
Tương tự ứng dụng Multi Chat Room của TCP; UDP cũng có thể thực hiện truyền tin cho các Client đang kết nối đến cùng một Server cùng Port



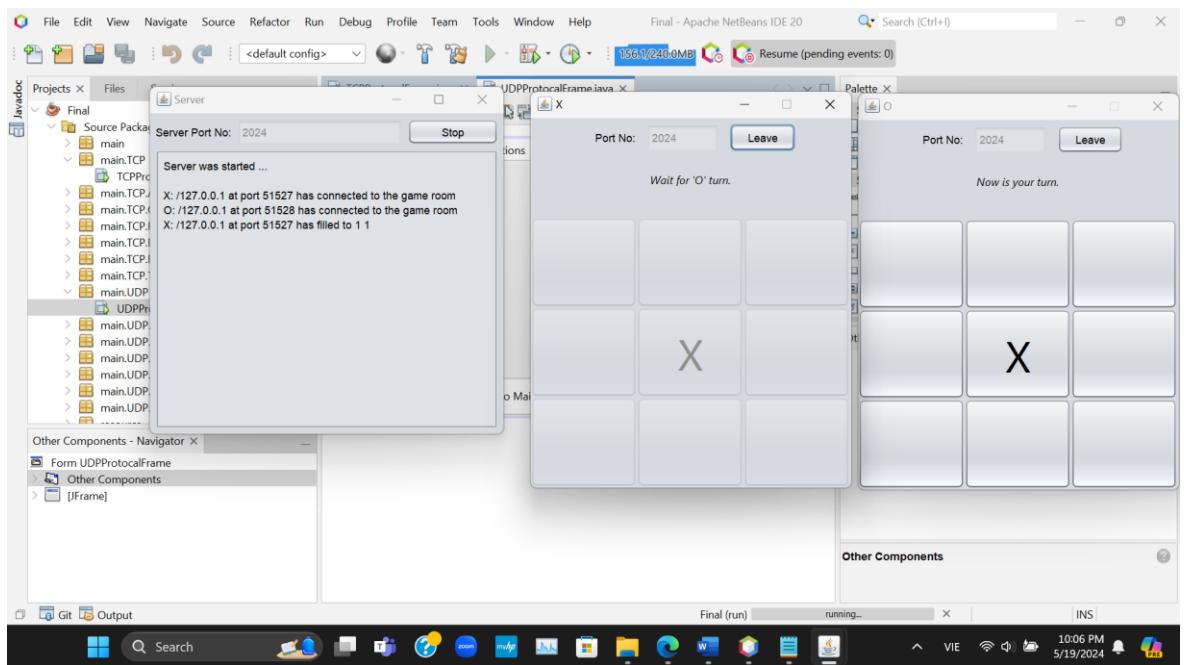
Ứng dụng UDP TicTacToe Game



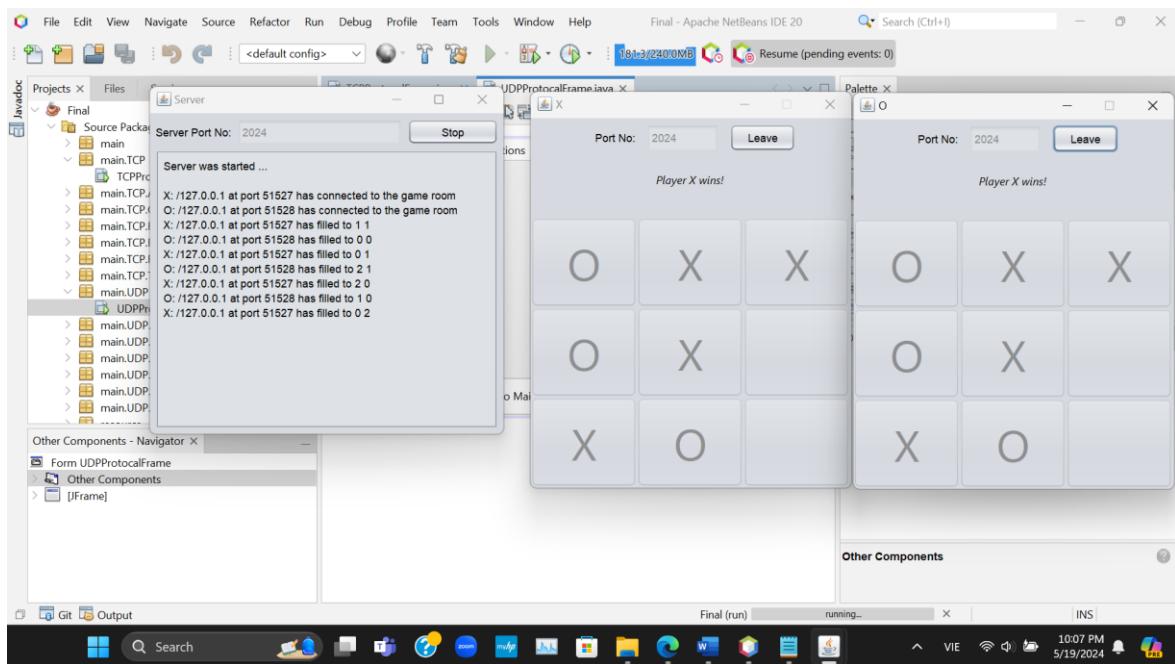
Client nào Join vào phòng trước sẽ được quyền đi trước với quân cờ X; còn lại sẽ là O và phải chờ lượt đánh của X



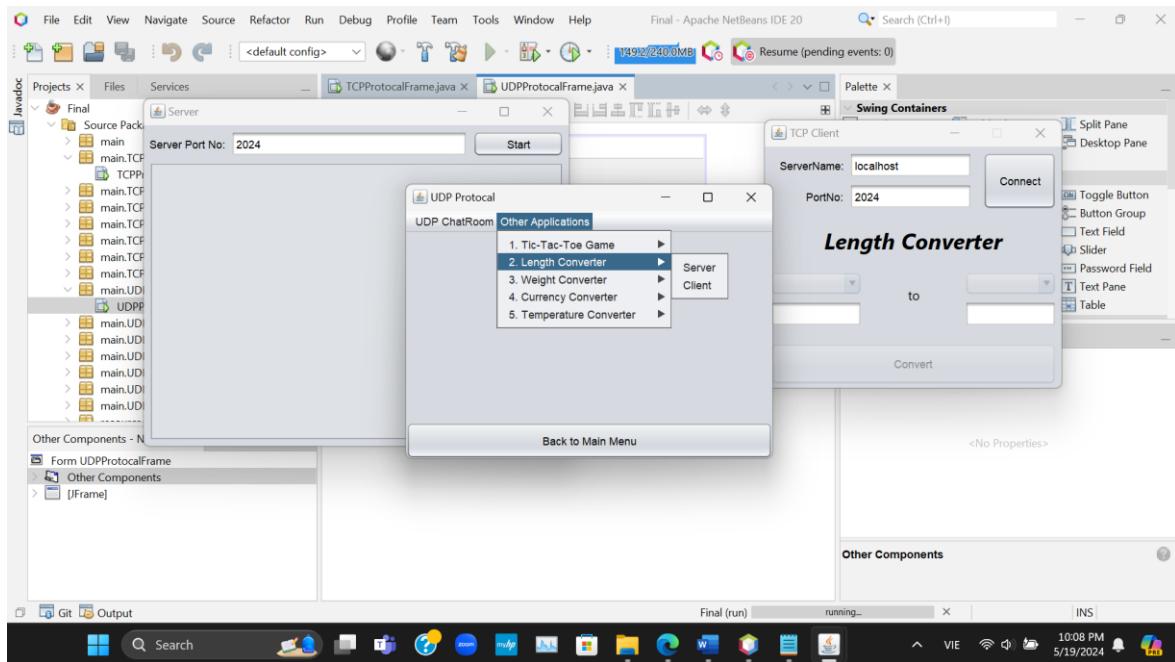
Client thứ 3 sẽ không thể tham gia vào phòng game vì đã đủ số lượng người chơi



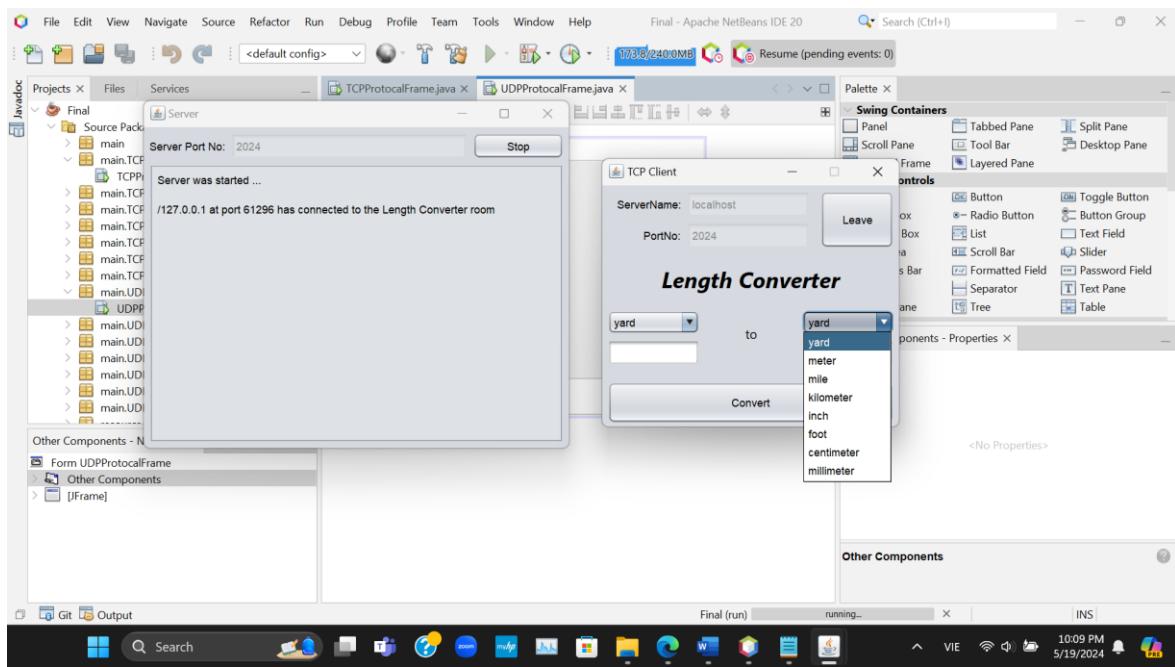
Khi thực hiện nước đi xong thì sẽ chuyển lượt đánh đến đối thủ; đồng thời ở Server cũng có hiện thông tin Client thực hiện nước đi nào



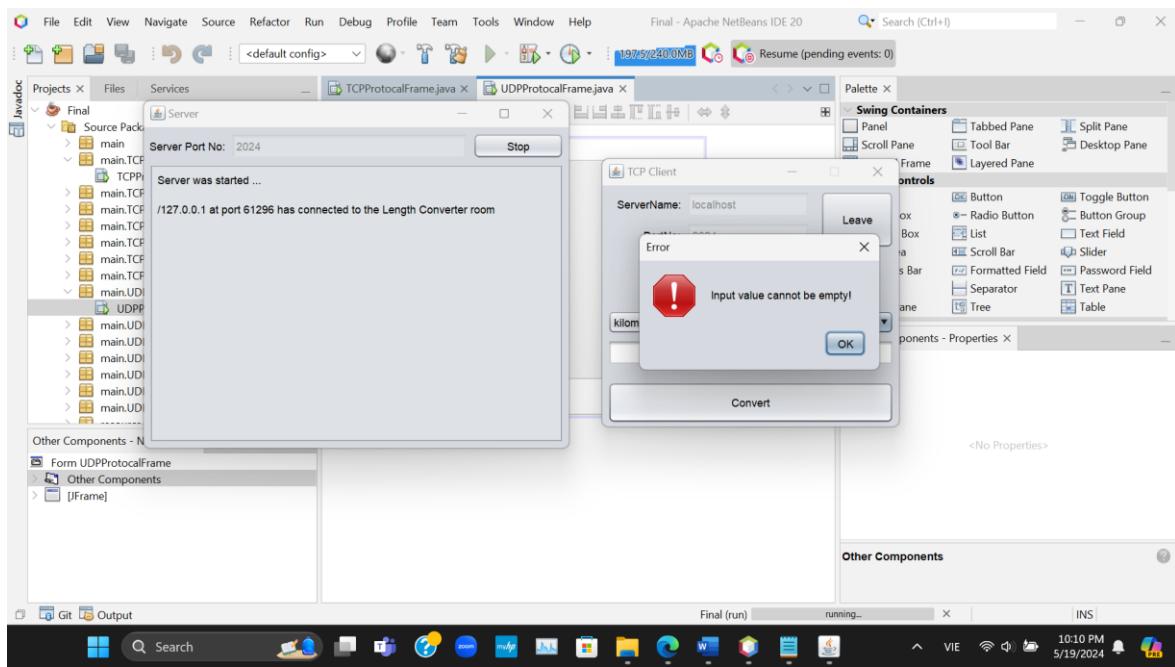
Trò chơi kết thúc khi có 1 trong 2 thắng hoặc cả 2 đều hòa



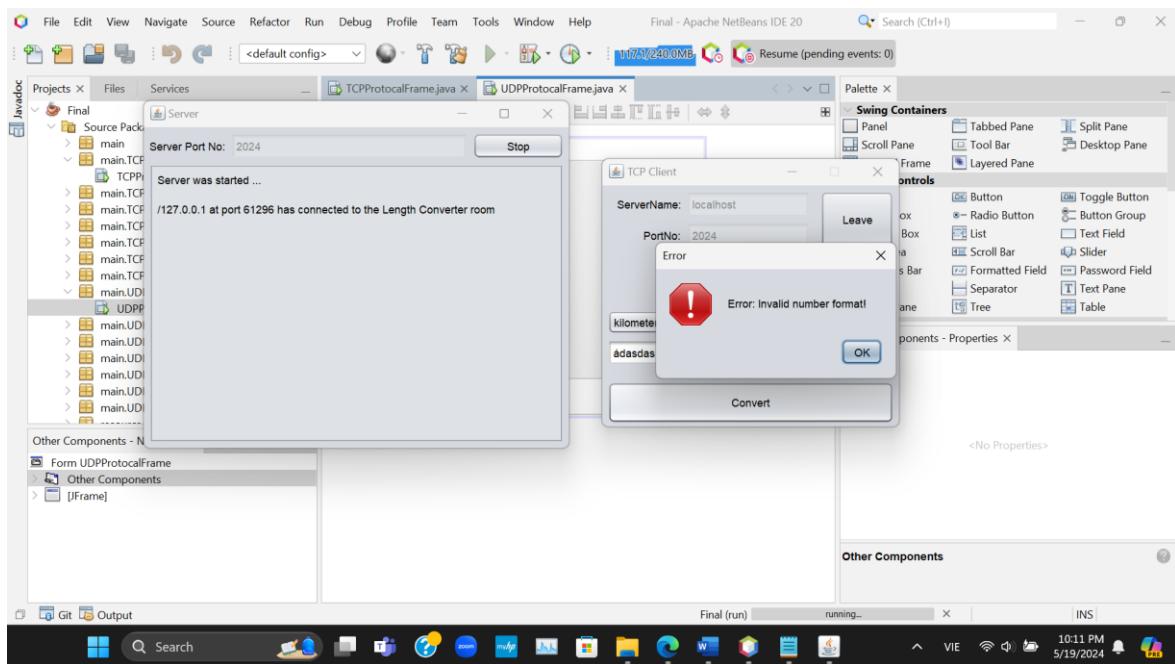
Ứng dụng UDP Length Converter (Chuyển đổi đơn vị độ dài)



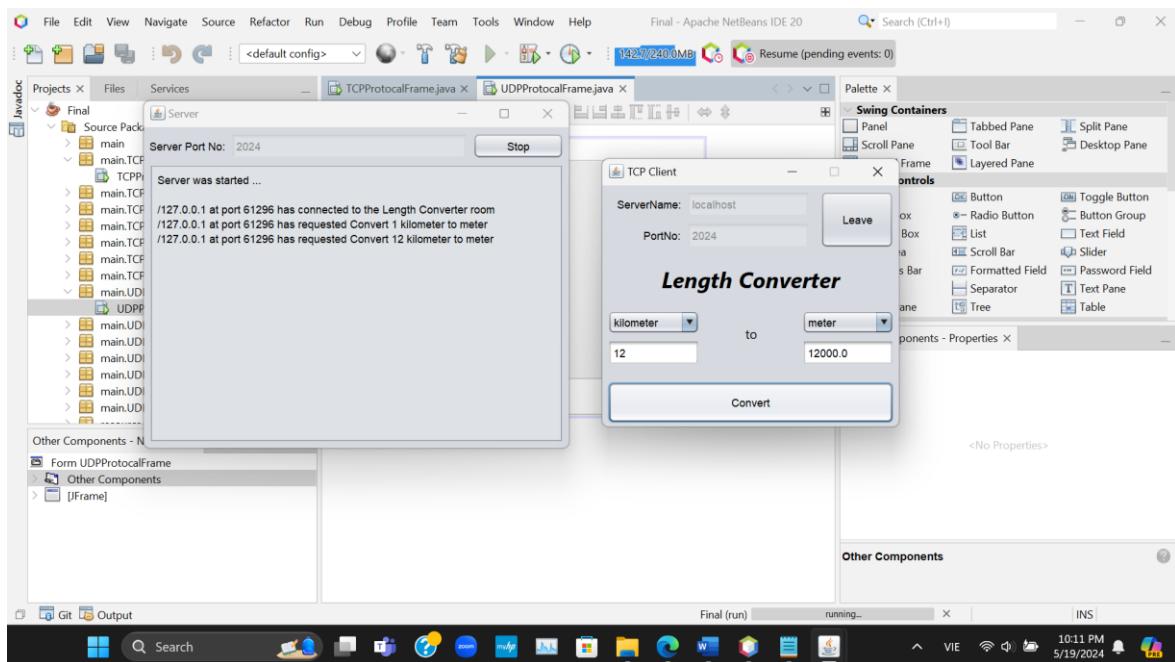
Khi kết nối đến Server thành công sẽ nhận được danh sách đơn vị mà Server hỗ trợ chuyển đổi như trên



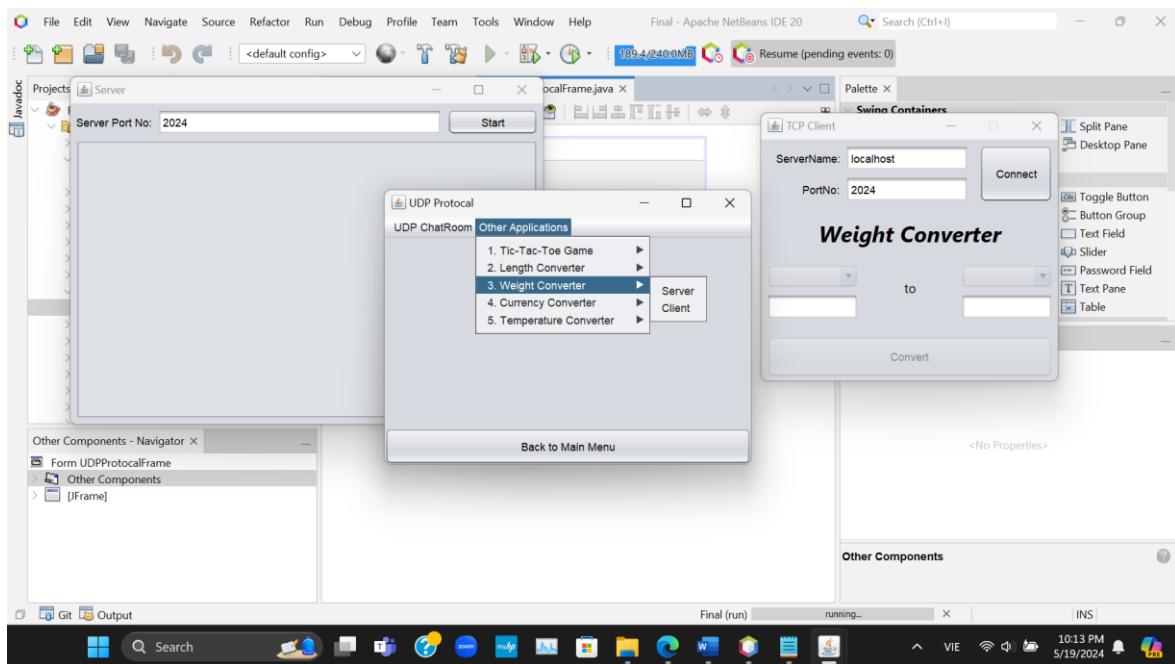
Báo lỗi khi để trống input



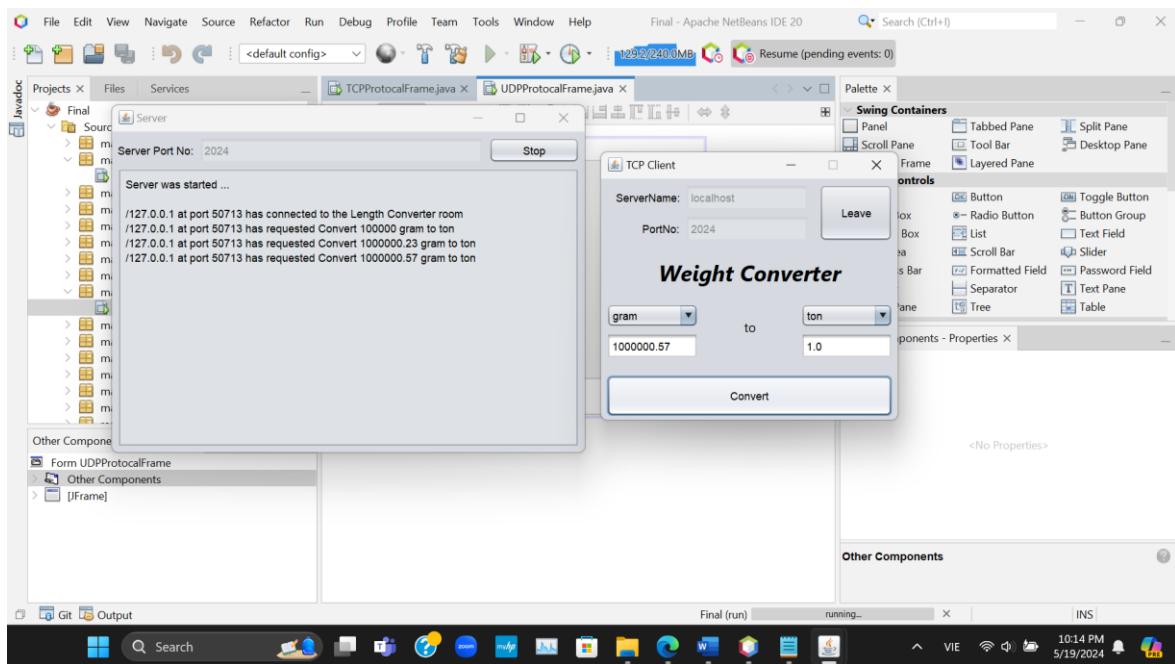
Báo lỗi khi input không phải là dạng số



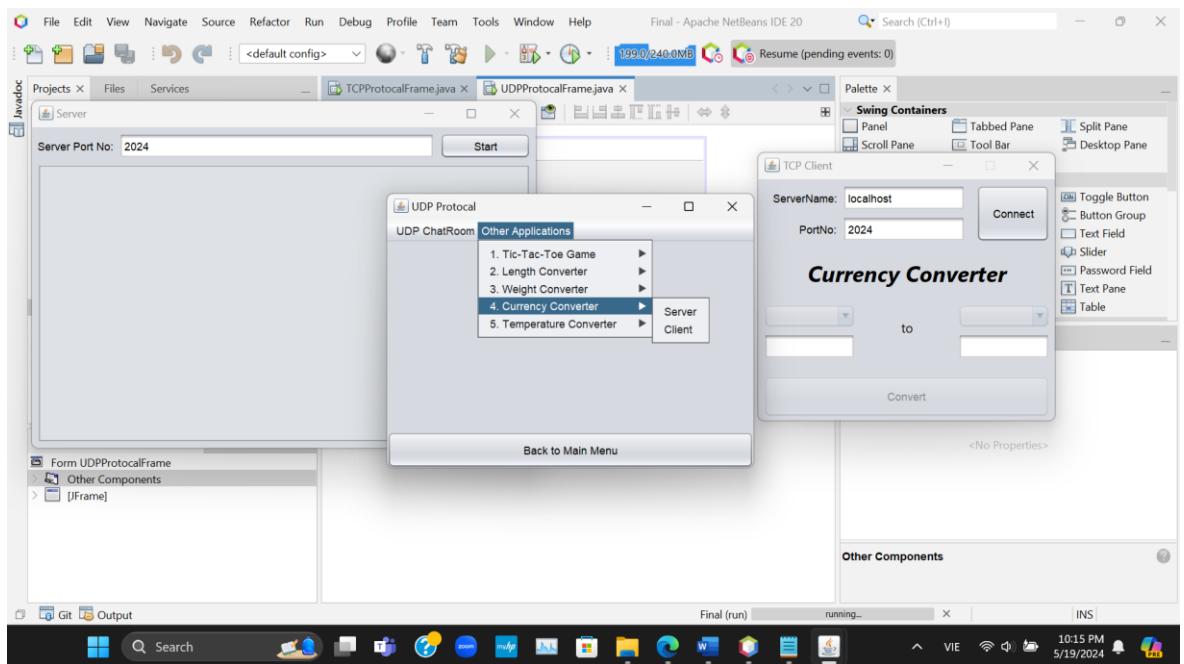
Thực hiện chuyển đổi đơn vị và nhận kết quả từ Server; phía Server đồng thời ghi lại yêu cầu từ Client



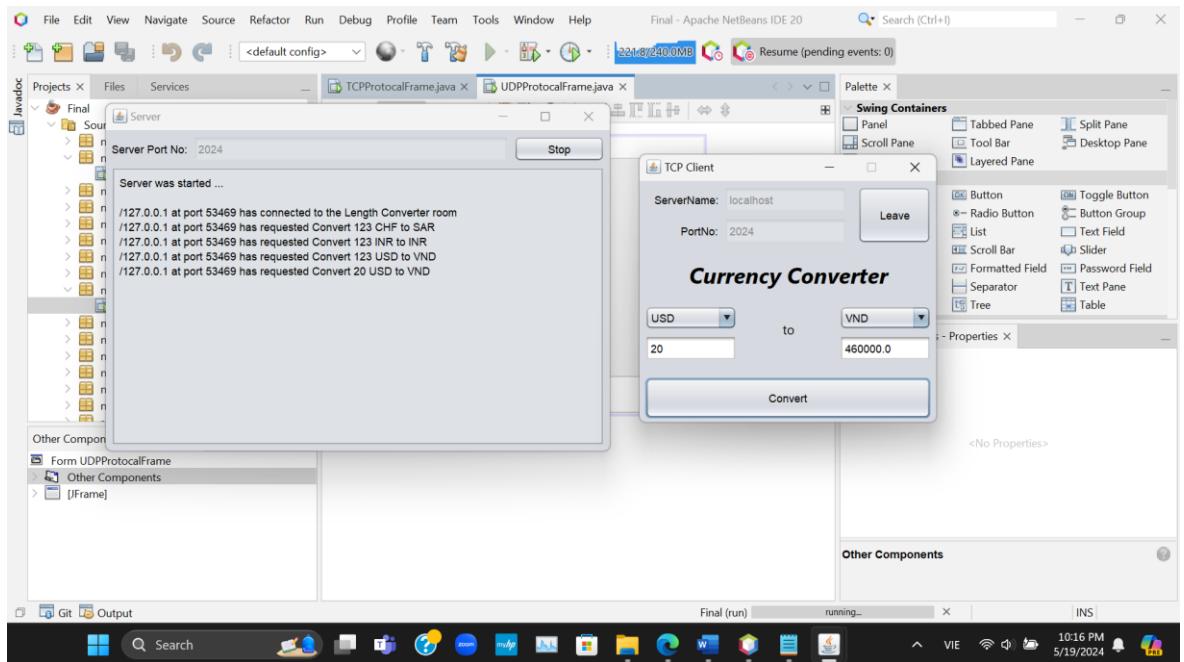
Ứng dụng UDP Weight Converter (Chuyển đổi đơn vị cân nặng)



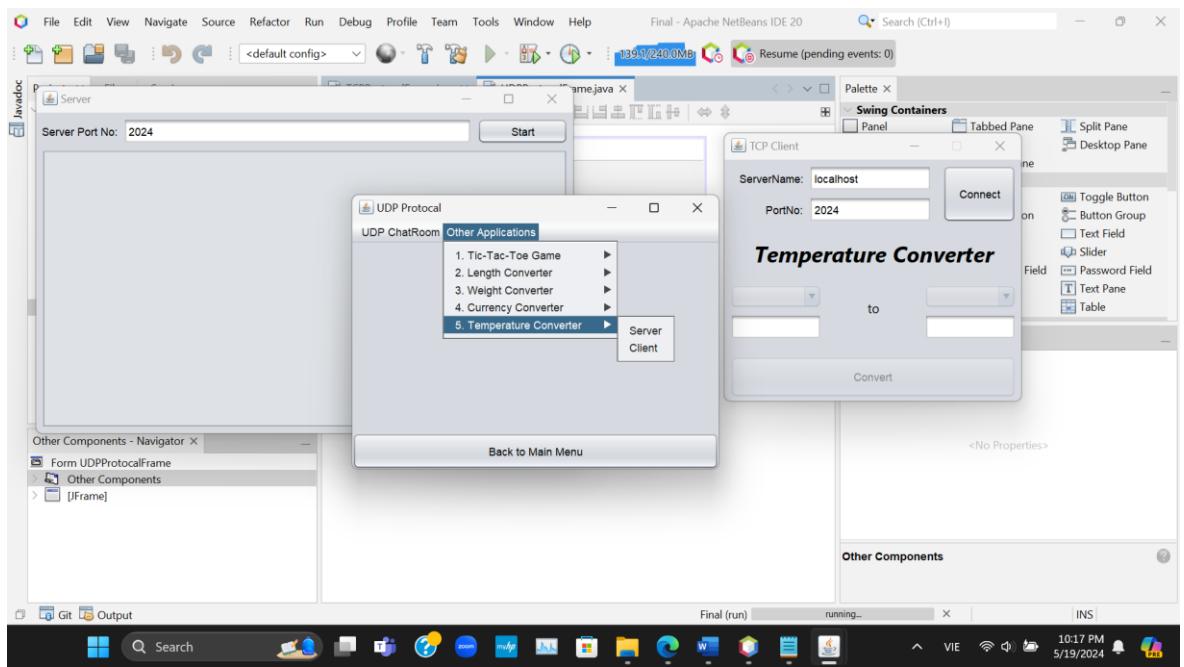
Tương tự với ứng dụng chuyển đổi đơn vị trước đó



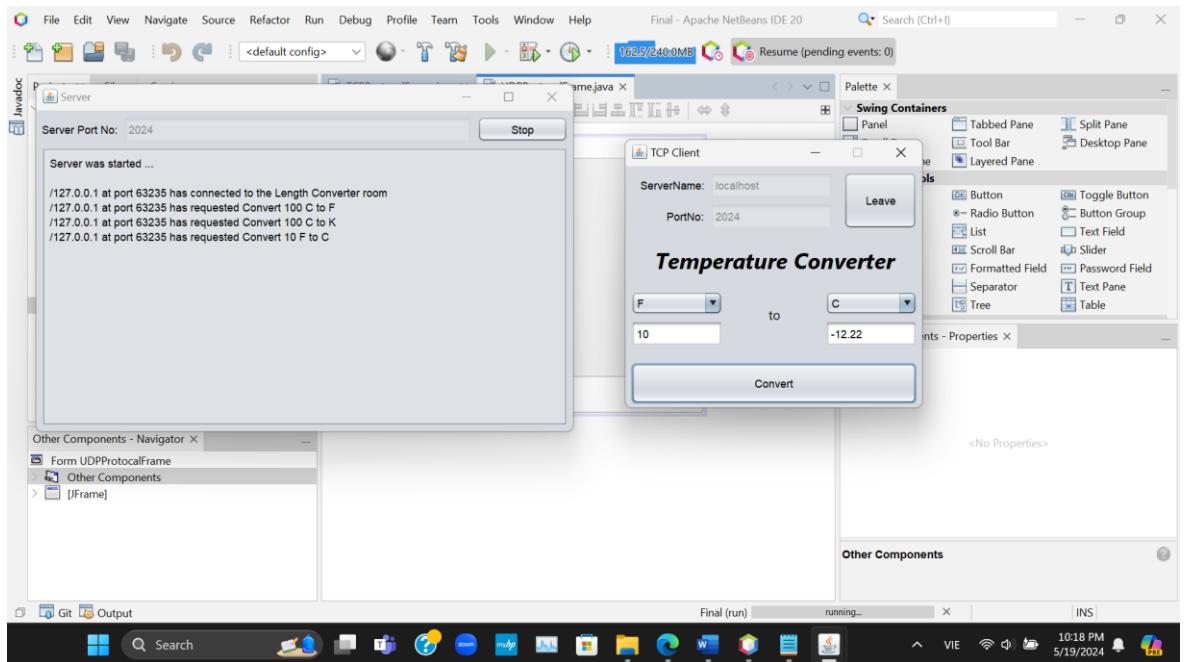
Ứng dụng UDP Currency Converter (Chuyển đổi đơn vị tiền tệ)



Tương tự với ứng dụng chuyển đổi đơn vị trước đó



Ứng dụng UDP Temperature Converter (Chuyển đổi đơn vị nhiệt độ)



Tương tự với ứng dụng chuyển đổi đơn vị trước đó

CHƯƠNG 4. KẾT LUẬN

4.1 Những kết quả đạt được

Trong quá trình phát triển ứng dụng TCP và UD, em đã đạt được một số kết quả đáng kể:

Hoàn Thiện Hệ Thống Giao Tiếp Client - Server: Em đã xây dựng một hệ thống đa dạng cho các loại ứng dụng khác nhau sử dụng 2 phương thức TCP và UDP. Hệ thống này có khả năng giao tiếp giữa các Client và Server một cách chính xác và linh hoạt.

Tích Hợp Giao Diện Người Dùng: Chúng tôi đã thiết kế giao diện người dùng trực quan và dễ sử dụng bằng cách sử dụng Swing trong Java. Giao diện giúp người dùng dễ dàng sử dụng.

Xử Lý Ngoại Lệ và Tình Huống Bất Thường: Hệ thống đã được cải thiện để xử lý các tình huống bất thường như dữ liệu không hợp lệ hoặc mất kết nối với máy chủ.

4.2 Những hạn chế và hướng phát triển

Mặc dù đã đạt được một số kết quả tích cực, nhưng ứng dụng vẫn còn một số hạn chế và cơ hội để phát triển thêm:

Bảo Mật và Xác Thực: Hiện tại, hệ thống chưa có cơ chế bảo mật hoặc xác thực, điều này có thể tạo ra lỗ hổng bảo mật. Cần phát triển cơ chế bảo mật để đảm bảo an toàn cho dữ liệu và thông tin người dùng.

Tích Hợp API Tỷ Giá: Để cải thiện tính linh hoạt và chính xác của chuyển đổi tiền tệ, có thể tích hợp API từ các nguồn tin cậy như Ngân hàng Trung Ương hoặc dịch vụ tài chính trực tuyến để cập nhật tỷ giá hối đoái.

Giao Diện Người Dùng Tùy Biến: Cải thiện giao diện người dùng để cho phép người dùng tùy chỉnh các cài đặt và hiển thị thông tin phản hồi một cách rõ ràng hơn.

Tối Ưu Hiệu Năng và Độ Ổn Định: Tiếp tục tối ưu hiệu năng và độ ổn định của hệ thống, đặc biệt là trong các môi trường mạng không ổn định hoặc có tải cao.

Kiểm Thủ và Sửa Lỗi: Cân tiến hành kiểm thử kỹ lưỡng và sửa lỗi để đảm bảo ứng dụng hoạt động một cách chính xác và mượt mà trong mọi điều kiện.

Hỗ Trợ Đa Ngôn Ngữ: Để mở rộng phạm vi sử dụng, có thể thêm tính năng hỗ trợ đa ngôn ngữ để cho phép người dùng chọn ngôn ngữ ứng dụng.

TÀI LIỆU THAM KHẢO

Tiếng Việt

- Slide bài giảng
- Slide thực hành