ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN: MẠNG MÁY TÍNH

CHATROOM APPLICATION

Giảng viên: PGS.TS. Nguyễn Hoài Sơn

Lớp: INT2213 3

Thành viên:

1. Phạm Bảo Phúc 19020083
2. Phạm Gia Tâm 19020429

MỤC LỤC

I. Các thuật ngữ trong bài:

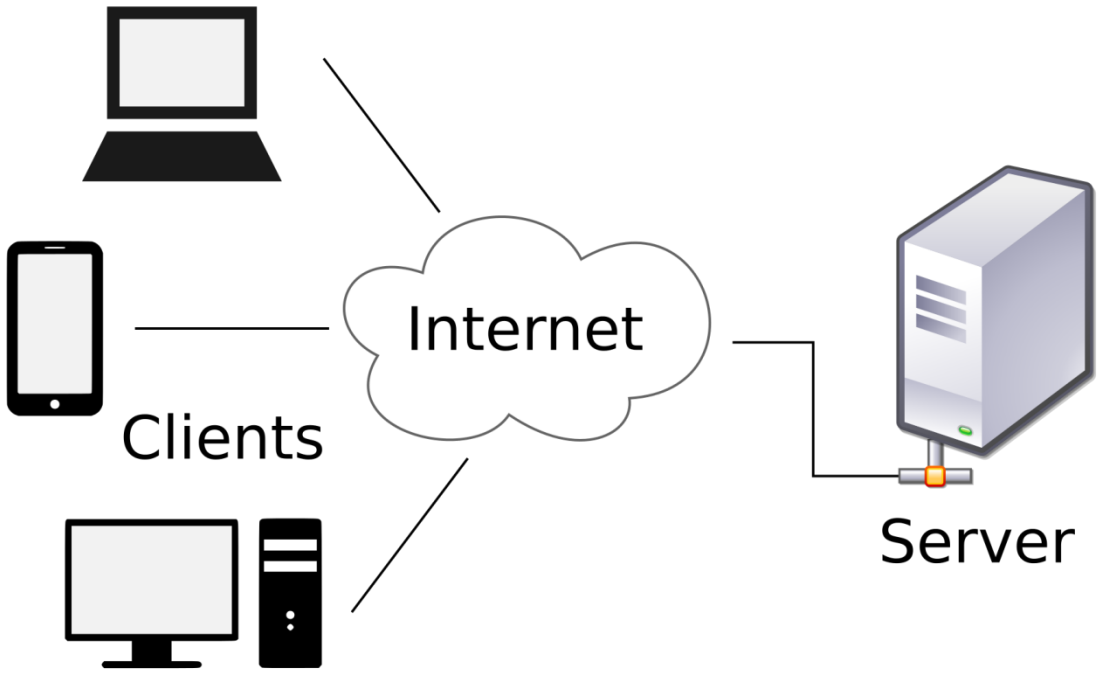
1. Chatroom	Phòng chat chung cho các người dùng
2. User	Người dùng
3. Username	Tên người dùng
4. Userlist	Danh sách người dùng đang hoạt động trong hệ thống

II. Giới thiệu về socket, phương thức TCP và mô hình Client – Server:

1. Giới thiệu mô hình Client-Server

a. Client: hay chính là máy khách, máy trạm – là nơi gửi yêu cầu đến server. Nó tổ chức giao tiếp với người dùng, server và môi trường bên ngoài tại trạm làm việc. Client tiếp nhận yêu cầu của người dùng sau đó thành lập các query string để gửi cho server. Khi nhận được kết quả từ server, client sẽ tổ chức và trình diễn những kết quả đó.

b. Server: xử lý yêu cầu gửi đến từ client. Sau khi xử lý xong, server sẽ gửi trả lại kết quả, client có thể tiếp tục xử lý các kết quả này để phục vụ người dùng. Server giao tiếp với môi trường bên ngoài và client tại server, tiếp nhận yêu cầu dưới dạng query string (xâu ký tự). Khi phân tích xong các xâu ký tự, server sẽ xử lý dữ liệu và gửi kết quả về cho client.



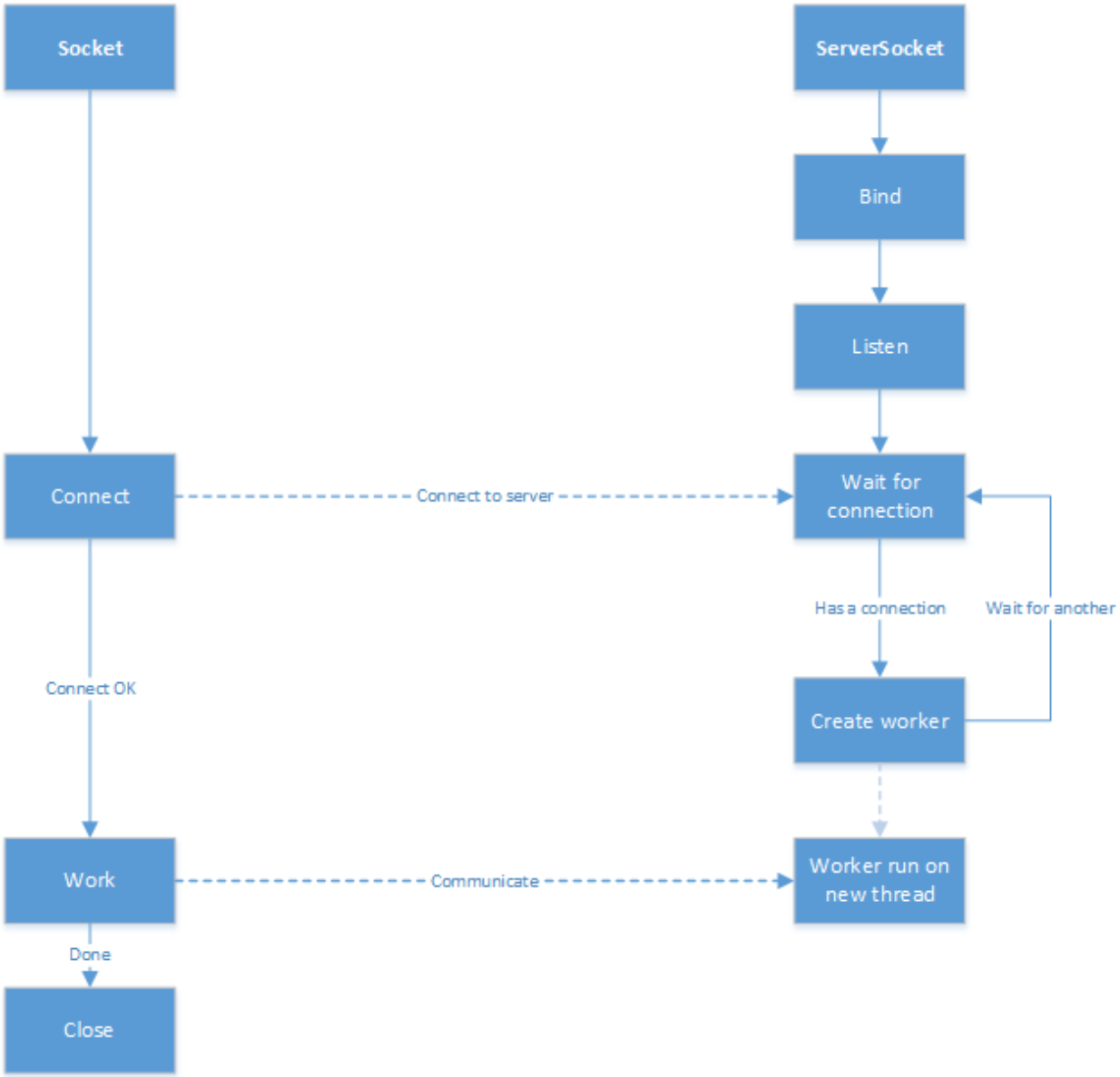
c. Sử dụng TCP socket trong mô hình client-server:

Client sẻ sử dụng một socket để làm việc với server theo 3 bước:

- Kết nối tới server.
- Trao đổi dữ liệu với server.
- Đóng kết nối

Server sẽ làm việc với client theo 5 bước:

- Bind tới một endpoint(một địa chỉ IP và port) trên server.
- Bắt đầu lắng nghe kết nối.
- Đợi các kết nối.
- Tạo một worker khi có kết nối mới.
- Quay lại bước 3.



Một hình ảnh minh họa sơ bộ gần đúng cơ chế hoạt động của bài

d. Worker: ở đây sẽ là một lớp wrapper có nhiệm vụ làm việc trực tiếp với client và nó sẽ được thực thi trong một thread khác để đảm bảo server có thể phục vụ được nhiều client cùng lúc. Thực ra khi server chấp nhận một kết nối, nó sẽ sinh ra một socket để ta làm việc với client và socket này sẽ được Worker bao bọc lại cùng với nó là một thread mới được sinh ra. Sau khi Worker được sinh ra thì server sẽ quay lại lắng nghe tiếp các kết nối khác, vòng lặp như thế sẽ diễn ra mãi mãi cho đến khi ta đóng server bằng phương thức close hoặc có một lỗi xảy ra trong lúc lắng nghe. Thông thường thì server sẽ cần phải hoạt động 24/7 nên ít khi xảy ra trường hợp cần phải đóng server.

2. Socket

a. Trong hệ thống mạng máy tính tồn tại những mô hình tham chiếu có kiến trúc phần tầng (OSI, TCP/IP…) nhằm hỗ trợ chức năng trao đôi thông tin giữa các ứng dụng ở nhiều máy tính khác nhau.

b. Dữ liệu bên gửi sẽ được đóng gói (Encapsulation) từ tầng trên đến tầng cuối là tầng vật lí (Physical Layer), sau đó nhờ tầng vật lí này chuyển dữ liệu đến tầng vật lí máy bên nhận, bên nhận tiến hành giải mã (decapsulation) gói dữ kiện từ tầng dưới lên tầng trên cùng, là tầng ứng dụng (application layer).

c. Ở đây, Socket chính là cửa giao tiếp giữa tầng ứng dụng và tầng giao vận (Transport layer). Nói cách khác, Socket là giao diện do ứng dụng tạo ra trên máy trạm, quản lí bởi hệ điều hành qua đó các ứng dụng có thể gửi/nhận thông điệp đến/từ các ứng dụng khác. Ở đó, Socket sẽ được ràng buộc với một mã số cổng (Port Number) để giúp tầng giao vận định danh được ứng dụng nhận/gửi thông điệp.

d. Socket được hỗ trợ trên nhiều ngôn ngữ như C, Java, Pearl, Python,…. Bài làm này sử dụng ngôn ngữ Java

III. Đặc tả chức năng: Các chức năng của ứng dụng:

Ứng dụng sử dụng phương thức TCP

1. Client:

- Ứng dụng hoạt động dựa trên cơ chế chatroom, một room chung được tạo ra cho phép các user tham gia vào và chat với nhau trong room chung này. User có thể lựa chọn một username ưa thích tự đặt cho mình, từ đó bước vào chatroom chung với những user trong hệ thống
- Danh sách các user hoạt động (Userlist) được hiển thị ngay bên cạnh màn hình hiển thị tin nhắn, giúp user theo dõi và nắm được thông tin về những user khác
- User có thể gửi tin nhắn dựa theo nhập từ bàn phím vào text input hoặc sử dụng file txt đã soạn sẵn nội dung muốn gửi. Sau khi gửi nội dung gửi sẽ được hiển thị trên màn hình của người gửi cũng như toàn bộ user còn lại trong hệ thống.
- Khi có user mới tham gia hoặc rời khỏi chatroom, thông tin này sẽ được thông báo hiển thị trên màn hình chat, đồng thời userlist sẽ được cập nhật lại.
- User có thể rời chatroom bất cứ khi nào.

2. Server:

- Mọi thông tin về khởi tạo server, user tham gia/rời chatroom hay các hoạt động đều được hiển thị trên log hoạt động.
- Theo dõi và quan sát toàn bộ user trong chatroom. Nói cách khác server cũng có một userlist riêng.

IV. Các thành phần:

1. ChatClient:

a. Giao diện chính của Client:

b. Các chú thích:

Khu vực	Giải thích
1	Text input area nhập nickname/username
2	Màn hình hiển thị các tin nhắn trong chatroom
3	Text input area nhập tin nhắn để gửi đi
4	Active userlist
5	Các buttons gửi tin nhắn, gửi tin nhắn từ file txt và thoát, ngắt kết nối

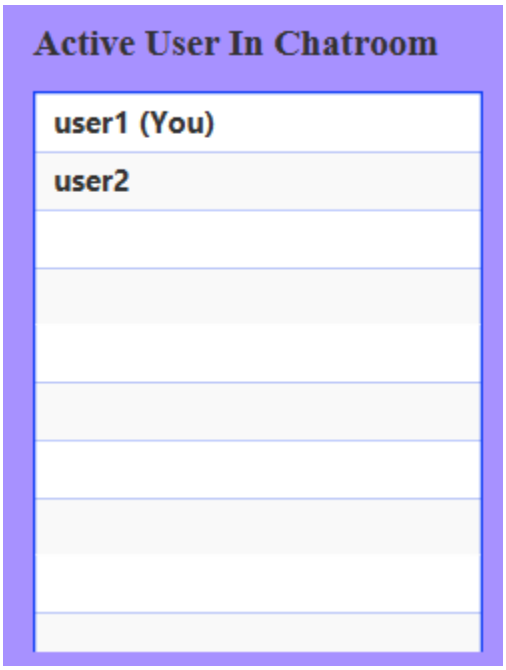
c. Các thao tác:

- Nickname ở đây chính là username mà user phải nhập vào text input ở trên. Chỉ khi nào người dùng nhập thành công nickname mà không trùng với bất kỳ user khác đang có trong hệ thống thì việc join mới thành công.

- Ban đầu, khi nhập nickname userlist sẽ chưa hiển thị các active user. Chỉ khi tham gia thành công vào chatroom (được hiểu là đã nhập username và không bị trùng) thì userlist mới được hiển thị, và user mới có thể sử dụng tính năng gửi tin nhắn lên server và màn hình hiển thị.

- User có 2 lựa chọn gửi tin nhắn, đó là thông qua việc nhập từ bàn phím vào text input area gửi tin nhắn rồi ấn "Send". Lựa chọn thứ hai là gửi thông qua file txt tin nhắn đã soạn sẵn trong file, và văn bản sẽ được gửi thẳng và hiển thị lên màn hình. Để thao tác lựa chọn này, chọn "File txt" rồi chọn file txt đã soạn sẵn gửi đi.

- Khi tham gia thành công chatroom, userlist sẽ hiển thị danh sách các active user, bao gồm cả chính user đó. Dưới đây là màn hình hiển thị userlist sau khi tham gia thành công:



2. Thread:

- Đây là thành phần xử lý các kết nối từ nhiều user đến server. Giúp hỗ trợ nhiều client user cùng một thời điểm.

- Nó là 1 class tích hợp trong class ChatServer, và không có giao diện

- Xử lý từng client connection đến server tại cùng một thời điểm, mỗi một kết nối như vậy lại có một luồng Thread riêng.

3. ChatServer:

a. Giao diện:

- Khu vực 1: Các log về kết nối của các client đến server
- Khu vực 2: Active user trong chatroom.

b. Các thông tin hiển thị:

- Thông tin thời gian khởi tạo chạy server
- Thông tin thời gian khởi tạo kết nối từ các tiến trình client đến server
- Đóng kết nối khi một user gặp trục trặc hoặc rời khỏi chatroom.

V. Cơ chế hoạt động:

1. Server:

a. Khởi tạo và chạy:

```
serverSocket = new ServerSocket(SERVER_PORT);
Platform.runLater(() ->
        logItems.add("MultiThreadServer started at " + new Date()));
while (true) {
    Socket socket = serverSocket.accept();
    /*
    Add accepted socket to the socketList.
     */
    socketList.add(socket);
    Platform.runLater(() -> logItems.add("Connection from " + socket + " at " + new Date()));
    DataOutputStream dataOutputStream = new DataOutputStream(socket.getOutputStream());
    // Save output stream to hashtable
    outputStreams.put(socket, dataOutputStream);
    new ServerThread( server: this, socket);
}
```

- Đầu tiên khởi tạo ServerSocket (đã được Java hỗ trợ sẵn đối tượng này), truyền vào tham số là port (kiểu int). Cần lưu ý port này cần tránh các port mặc định của các giao thức khác (80 của http, 25 của smtp hay 20/21 của ftp). Trong ứng dụng này port được tạo là 8000 và lưu trong hằng SERVER_PORT.

- Yêu cầu đặt ra là Server phải trong tình trạng luôn sẵn sàng chờ yêu cầu kết nối từ client, phục vụ cho việc ai cũng có thể tham gia chatroom bất cứ khi nào miễn là server đang chạy. Để đạt được mục đích này, ta sử dụng vòng lặp while(true). Trong vòng lặp này, ta sử dụng phương thức accept() chấp nhận request từ client và sau đó khởi tạo thread cho việc socket tương ứng với request/client này. Cơ chế hoạt động của Thread sẽ được đề cập ở phần sau.

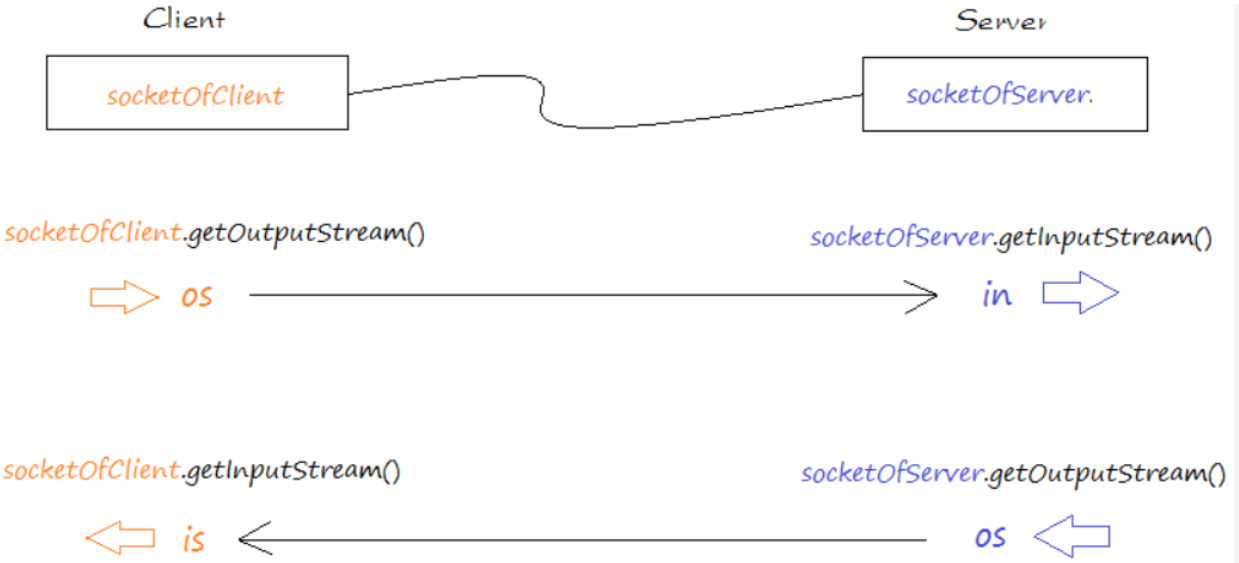
- Với yêu cầu là multi-user chatroom, ta cần thêm hashtable (được đặt tên là outputStreams) outputStream mapping giữa socket được khởi tạo cho client và outputStream tương ứng. Ngoài ra ta lưu các socket đã được khởi tạo vào một list lưu trữ là socketList.

- Cuối cùng khởi tạo thread tương ứng để xử lý với từng client cụ thể.

b. Các khái niệm về DataOutputStream và DataInputStream:

- Client và Server đều có 2 thành phần này. Trong đó DataOutputStream phục vụ cho việc gửi tin nhắn đi. Như vậy, khi client gửi tin nhắn đến server thì sẽ đưa tin nhắn vào DataOutputStream của Client. Và tin nhắn này sẽ gửi đến DataInputStream của Server.

- Ngược lại cũng tương ứng khi Server gửi tin nhắn đến Client.



c. Toàn bộ các thao tác còn lại: sẽ do Thread quản lý và xử lý, bao gồm việc nhận tin nhắn, nhận danh sách active user, broadcast tin nhắn và userlist đến toàn bộ Client, cũng như đóng kết nối.

2. Thread:

a. Khởi tạo:

- Thread có hai đối tượng là socket kết nối và chatServer. Mỗi khi một client request connection được gửi đến server, server sẽ khởi tạo một luồng thread mới, truyền vào chính Server và socket đã được accept ở trên (`serverSocket.accept()`)vào trong luồng Thread. Nói cách khác, server thì tất cả luồng Thread có chung 1 chat server, nhưng socket ở đây thì riêng cho từng client.

```
public class ServerThread extends Thread {
    private ChatServer server;
    private Socket socket;
    String userName;
    boolean userJoined;

    public ServerThread(ChatServer server, Socket socket) {
        this.socket = socket;
        this.server = server;
        start();
    }
```

b. Xử lý user tham gia chatroom:

```
DataInputStream dataInputStream = new DataInputStream(socket.getInputStream());
DataOutputStream dataOutputStream = new DataOutputStream(socket.getOutputStream());
while (true) {
    /*
    Check if already joined
    1. if not then add them
    2. if already joined then broadcast the message from server to all users
    */
    if(!userJoined){
        userName = dataInputStream.readUTF();
        if(userList.contains(userName)){
            dataOutputStream.writeUTF(userName);
            System.out.println(userName + " already exist.");
        }
        else{
            userList.add(userName);
            dataOutputStream.writeUTF( str: "Accepted");
            server.updateUserlist();
            System.out.println(userName +" joined the chat room");
            userJoined = true;
            String userNotification = userName + " joined the chat room.";
            Platform.runLater(() ->
                    logItems.add(userName + " joined the chat room."));
            server.sendUsersInSystem(socket, userNotification);
            userItems.clear();
            userItems.addAll(userList);
        }
    }
```

- Khi một user nhập tên từ GUI của phía client, tên đó sẽ được chuyển về server kiểm tra trùng lặp trong userlist không. Nếu không trùng thì sẽ thực hiện câu lệnh trong khối else. Server thêm username mới vào userlist, và gửi thông tin chấp nhận (Accepted) username này cho client để Client thực hiện bắt đầu cho phép gửi tin nhắn. Thread cũng tiến hành gửi và cập nhật lại userlist cho các client (do có client mới tham gia chatroom) và hiển thị lên giao diện userlist và log của server.

- Nếu username không được chấp nhận thì khối câu lệnh if được thực thi. Thread sẽ gửi đi chính lại cái tên đó báo hiệu cho client là username này không được chấp nhận để phía Client yêu cầu user nhập lại.

c. Broadcast:

```
else if(userJoined){
    String string = dataInputStream.readUTF();
    System.out.println(string);

    server.sendToAll(string);
    server.updateUserlist();

    Platform.runLater(() -> logItems.add(string));

}
```

- Sau khi hoàn tất việc tham gia chatroom, Thread sẽ chờ nhận tin nhắn user gửi về, và sử dụng cơ chế broadcast, gửi tin nhắn đó từ server đến tất cả các user còn lại trong chatroom (`server.sendToAll(string)`) để hiển thị lên giao diện. Ngoài ra nếu có thay đổi về thêm/bớt các user thì cũng tiến hành cập nhật lại userlist cho tất cả client user luôn (`server.updateUserlist()`).

d. Exception:

```
catch(IOException ex) {
    System.out.println("Connection Closed for " + userName);
    ex.printStackTrace();
    Platform.runLater(() ->
            logItems.add("Connection Closed for " + userName));

    if( !userName.equals(null)){
        userList.remove(userName);
    }
    outputStreams.remove(socket);
    server.updateUserlist();
    if ( !userName.equals(null)){
        server.sendToAll(userName + " has left the chat room.");
    }
    Platform.runLater(() ->{
        userItems.clear();
        userItems.addAll(userList);
    });
}
```

- Khi client dừng kết nối, rời chatroom, exception ngoại lệ sẽ được ném ra ở phía Thread khi kết nối không còn. Phần xử lý này được để trong câu lệnh catch để không phải dừng toàn bộ chatroom, mà chỉ hoàn tất việc ngắt kết nối hoàn toàn với client cụ thể đó. Việc này sẽ gồm loại bỏ username khỏi userlist và cập nhật lại cho toàn bộ hệ thống chatroom, và loại bỏ kết nối với client này.

3. Client:

a. Khởi tạo kết nối server:

```
try {
    socket = new Socket(ChatServer.SERVER_IP, ChatServer.SERVER_PORT);
    dataInputStream = new DataInputStream(socket.getInputStream());
    dataOutputStream = new DataOutputStream(socket.getOutputStream());
    new Thread(() -> receiveMessages()).start();
}
// Providing feedback to user to notify connection issues.
catch (IOException ex) {
    errorLabel.setTextFill(Color.RED);
    errorLabel.setText("Unable to establish connection");
    System.err.println("Connection refused");
}
```

- Việc khởi tạo, rồi từ đó tạo các dataInputStream và OutputStream cho client được thực hiện như trên ảnh. Ở đây, SERVER_IP được đặt mặc định là localhost hoặc 127.0.0.1.

b. Nhận tin nhắn, userlist và cập nhật:

- Đầu tiên, khi tham gia và hệ thống chatroom, nickname mà user nhập sẽ được kiểm tra. Quy ước ở đây nếu server trả lại "Accepted" thì là thành công và giao diện sẽ hiển thị cho phép chat, còn không thì nếu trả lại y hệt username đã nhập thì hiểu rằng username bị nhập là lỗi, phải nhập lại.

- Sau khi tham gia thành công client chờ tin nhắn từ server gửi về input stream. Nếu như tin nhắn có bắt đầu bằng "[" thì đó là array list của userlist và phục vụ mục đích cập nhật hiển thị active userlist. Còn không thì đó là tin nhắn hiển thị lên giao diện.

c. Gửi tin nhắn – gửi file:

- Cơ chế gửi tin nhắn giống với server đã trình bày ở trên là đưa vào output stream gửi về server, điểm khác biệt là tin nhắn này do user nhập và gửi về server để broadcast.

- Với cơ chế file txt, có sự khác biệt:

```
dataOutputStream.writeInt(fileNameBytes.length);
/*
 Send the file name.
 */
dataOutputStream.write(fileNameBytes);
/*
 Send the length of the byte array so the server knows when to stop readi
 */
dataOutputStream.writeInt(fileBytes.length);
/*
 Send the actual file.
 */
dataOutputStream.write(fileBytes);
```

Có 4 yếu tố cần gửi đi, đó là độ dài tên file, tên file, độ dài file thực tế và chính file txt đấy. Có thể thấy trước khi gửi nội dung (tên file và nội dung file) ta phải gửi độ dài của chúng để server biết đọc đến khi nào thì dừng đọc file.

d. Đóng kết nối:

VI. Các thông tin thêm:

1. Phiên bản đồ họa sử dụng là javafx.
2. Các thư viện đã được thêm sẵn vào thông qua Maven, nhưng còn runtime thì không tự import được nên bản nộp sẽ có đính kèm và cả 1 file hướng dẫn chạy javafx runtime để hiển thị chương trình.

VII. Log phân chia công việc:

Thành viên / Thời gian	Phạm Bảo Phúc	Phạm Gia Tâm
21/4 – 25/4	Tiến hành phân chia công việc	
26/4 – 1/5	Tìm hiểu các project và hướng dẫn làm trên Youtube, nghiên cứu những cơ chế cơ bản nhất về Socket, ServerSocket và Thread	Tìm hiểu cơ chế hoạt động của Socket, TCP và mô hình Client – Server trong báo cáo.
2/5 – 8/5	- Lập trình Server, Client và các tính năng như userlist và gửi tin nhắn qua file txt	- Tạo bộ khung báo cáo, hoàn thiện phần giới thiệu về TCP và Socket - Lập trình về Thread xử lý luồng kết nối
8/5 – 15/5	- Bổ sung lại cơ chế hoạt động của Client và Server trong báo cáo	- Kiểm thử chức năng - Hoàn thiện giao diện

		- Hoàn thiện phần GUI và cơ chế hoạt động của Thread trong báo cáo

VIII. Tổng kết - Nhận xét - Đánh giá

1. Chương trình làm còn đơn giản, chưa đáp ứng được nhu cầu thiết thực

2. Báo cáo đã trình bày các kiến thức cơ bản về TCP Socket và ứng dụng của nó. Từ kết quả thực nghiệm cho thấy, chương trình còn nhiều hạn chế. Để chương trình có thể được sử dụng vào trong thực tế, cần giải quyết những nhược điểm mà chương trình còn tồn tại, hoàn thành tiếp các chức năng chưa hoàn thiện, thực hiện các ý tưởng mới nhằm năng cao tốc độ, hiệu suất và độ chính xác của chương trình.

3. Các tính năng hỗ trợ còn chưa cao, chỉ cho phép gửi file txt và tin nhắn cơ bản, chưa hỗ trợ gửi các loại file khác. Ngoài ra đây cũng chỉ là chatroom đơn, không hỗ trợ gửi cá nhân riêng tư 1v1.