Mạng máy tính

Bài tập lớn 1

Trường:	Đại học Bách Khoa
Môn học:	Mạng máy tính
GVHD :	TS Phạm Trần Vũ
Lớp :	L02-B
Năm học:	2016 - 2017

STT	Họ và tên:	MSSV
1	Đỗ Điền Tín	1414014
2	Tô Thành Nhân	1412643
3	Nguyễn Duy Minh	1412279
4	Nguyễn Thiện Nhân	1412632
5	Tôn Thất Lập	1411982

Nhóm Yomost

I. Giới thiệu về hệ thống

1. Mô tả tính năng hệ thống

a) Bên phía server:

- Đây là hệ thống máy chủ cho phép thực hiện các thao tác đơn giản trên một ứng dụng chat giữa nhiều user. Chức năng của server là tạo ra một kênh trung gian (port của ứng dụng) giúp cho các user client kết nối với nhau để sử dụng tính năng của client.
- *Server IP*: Khi người dùng khởi động server thì chương trình sẽ tự động lấy địa chỉ IP của máy tính để làm server IP.
- *Server Status:*
  + Mô tả tình trạng hiện tại của server đã được khởi động hay chưa.
  + Hiển thị "Server ready to serves" khi khởi động.
  + Thông báo cho server biết khi user client tham gia vào hoặc thoát khỏi phòng chat.
- Button "Close": dùng để thoát chương trình.

b) Bên phía client:

- Đây là giao diện mà các user sử dụng để kết nối tới server và có thể dùng các tính năng mà client cho phép.
- *Login:* yêu cầu sử dụng địa chỉ IP của server và User name của người dùng để đăng nhập vào hệ thống chat. Nếu User name bị trùng nhau thì hệ thống sẽ báo lỗi "Your login name has already exist" và yêu cầu đăng nhập lại.
- *Online list:* hiển thị các tên user đã đăng nhập thành công vào phòng chat.
- *Khung chat:* dùng để gửi đoạn hội thoại của một user tới phòng chat.
- *Button "Send" :* dùng để gửi nội dung chat tới Khung hiển thị nội dung chat.
- *Khung hiển thị nội dung chat:*
  + Hiển thị "login successful" khi user đăng nhập thành công vào phòng chat.
  + Hiển thị tất cả các nội dung chat giữa những user có trong phòng đó.
- *Button "Close":* dùng để thoát chương trình và đăng xuất ra khỏi phòng chat.

2. Bộ giao thức dùng trong ứng dụng

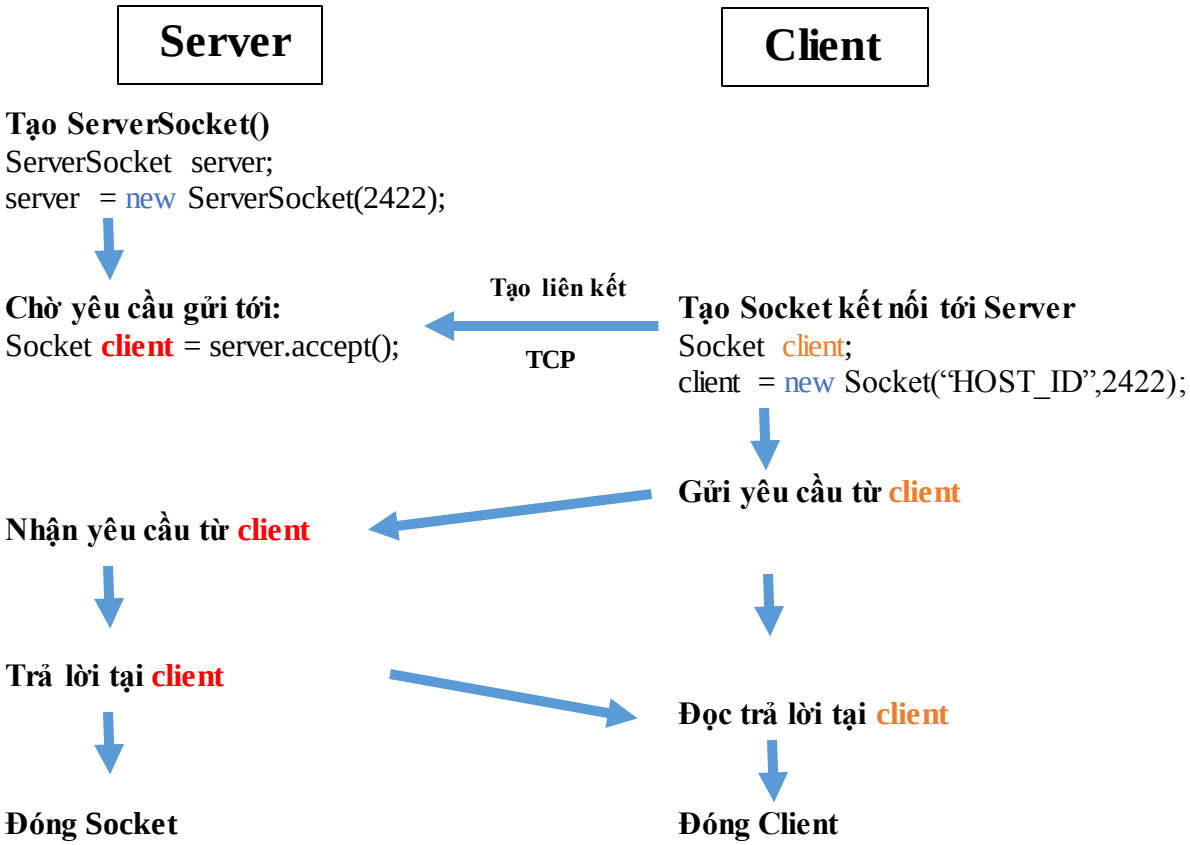
a. Giao thức tương tác giữa Server với Client

- Có 2 giao thức tương tác giữa Client với Server TCP và UDP
- Nhóm chọn giao thức tương tác TCP thông qua Socket trong Java
- Một socket có thể thực hiện các thao tác cơ bản:

+ Kết nối các máy tính từ xa

+ Gửi dữ liệu
+ Nhận dữ liệu
+ Ngắt liên kết
+ Nghe dữ liệu đến

- Lớp Socket của Java được sử dụng bởi cả client và server, có các phương thức tương ứng với các thao tác trên. Các thao tác này được cài đặt bởi lớp ServerSocket. Các socket cho client thường được sử dụng theo mô hình sau:



- Một socket mới được tạo ra bằng cách sử dụng hàm Socket().
- Socket cố gắng liên kết với một host ở xa thông qua HOST_ID và port
- Mỗi khi liên kết được thiết lập, các host ở xa nhận các luồng vào và luồng ra từ socket, và sử dụng các luồng này để gửi dữ liệu cho nhau. Kiểu liên kết này được gọi là song công (full-duplex)-các host có thể nhận và gửi dữ liệu đồng thời. Ý nghĩa của dữ liệu phụ thuộc vào giao thức.
- Khi việc truyền dữ liệu hoàn thành, một hoặc cả hai phía ngắt liên kết. Một số giao thức, như HTTP, đòi hỏi mỗi liên kết phải bị đóng sau mỗi khi yêu cầu được phục vụ. Các giao thức khác, chẳng hạn FTP, cho phép nhiều yêu cầu được xử lý trong một liên kết đơn

b. Dùng Stream để tương tác Server và Client

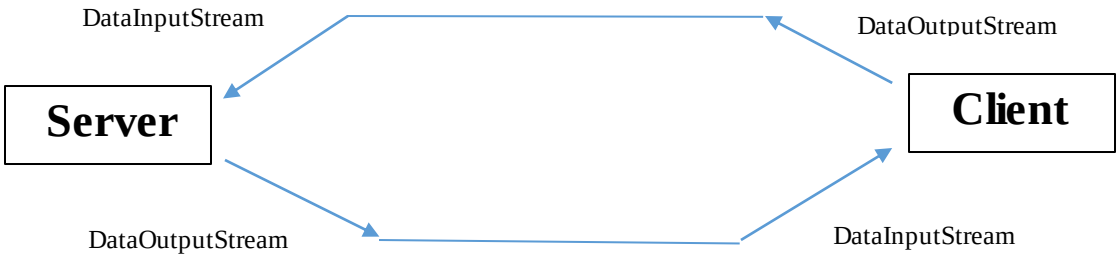
- **public OutputStream getOutputStream() throws IOException:**

Trả về output stream của Socket. Output stream được kết nối tới input stream của Socket từ xa

- **public InputStream getInputStream() throws IOException :**

Trả về input stream của Socket. Input stream được kết nối tới output stream của Socket từ xa

Server và Client tương tác với nhau thông qua DataStream theo mô tả như sơ đồ sau :



II. Xây dựng ứng dụng chat bằng Java

1. Mô tả

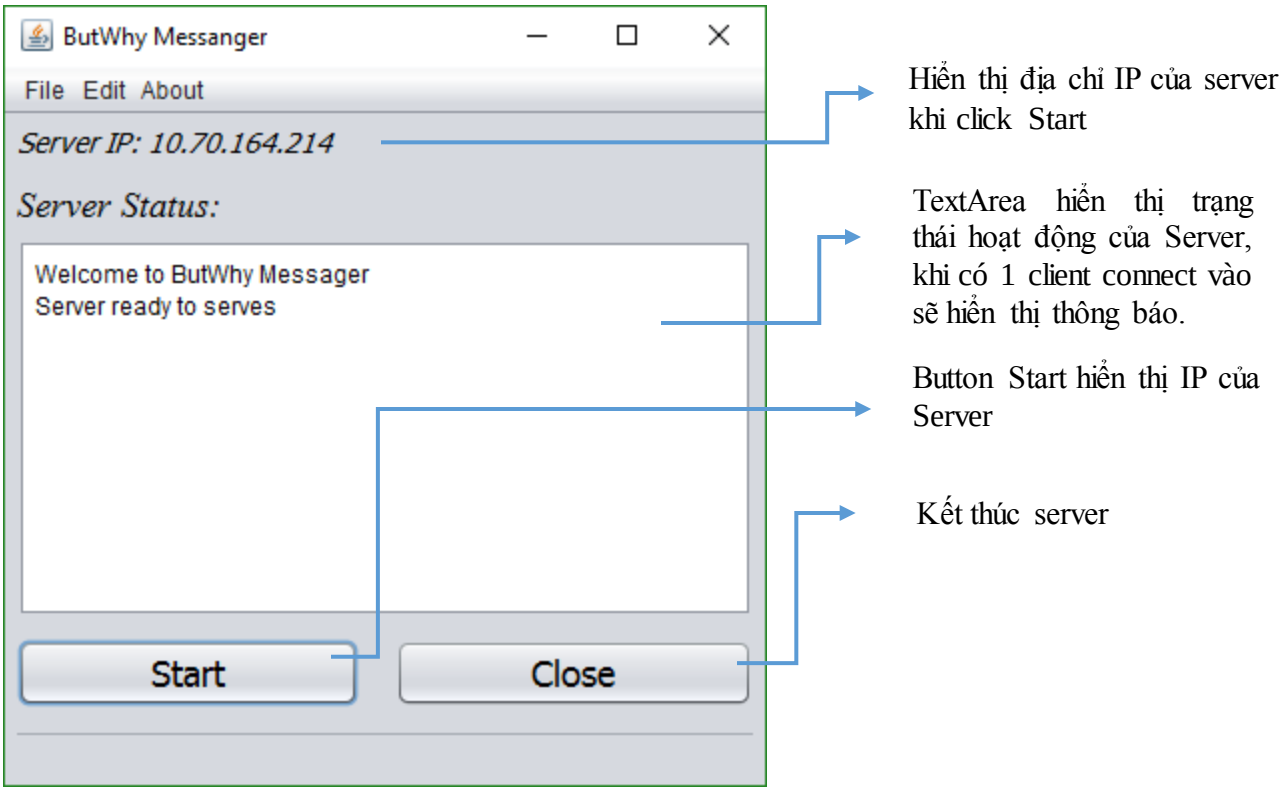
- Ứng dụng cho phép nhiều client ở nhiều máy khác nhau có thể kết nối với nhau.
- Có thể chat, trò chuyện một lúc nhiều người.

2. Hướng dẫn sử dụng

Bước 1:

Tạo Server, chạy file server.jar để khởi động server

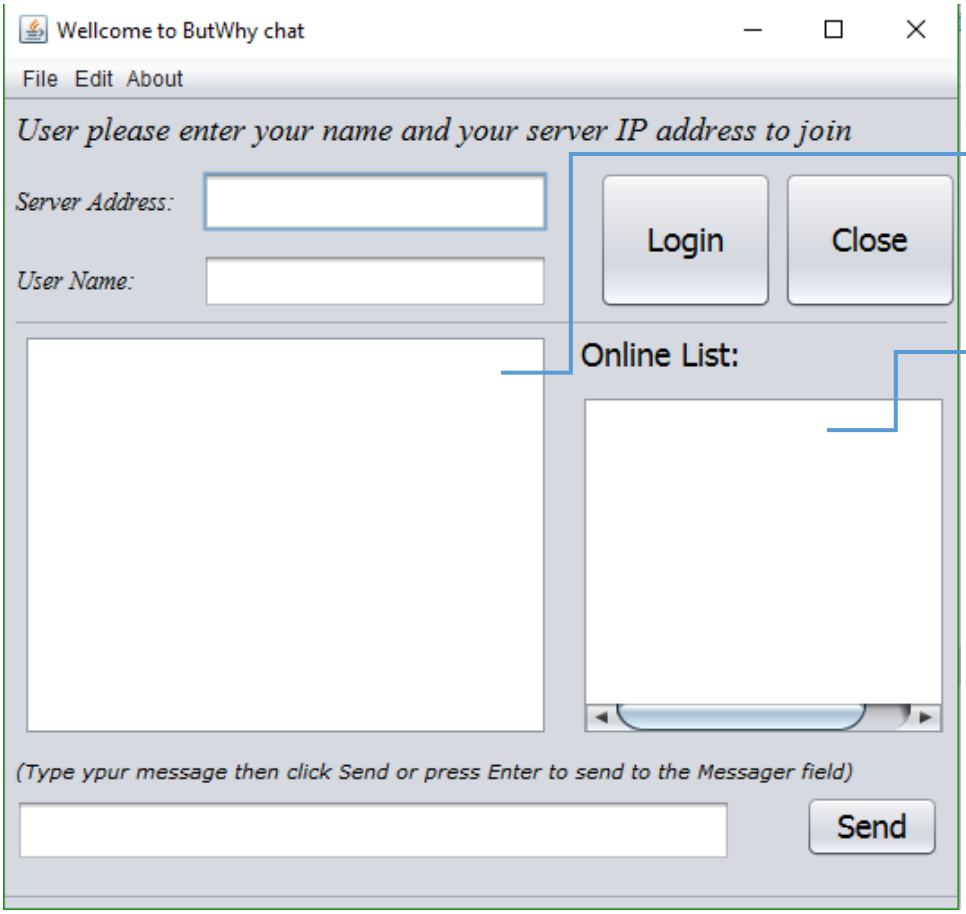
Click Start để lấy IP adress của Server



Hình 1. Server khởi động

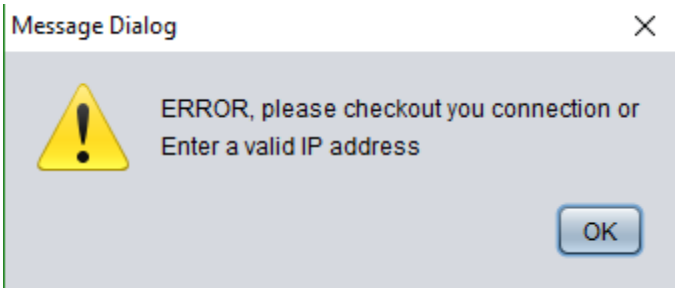
Server sẽ khởi tạo ServerSocket để chờ Client connect tới

Bước 2:
Chạy các client từ các máy tính khác để kết nối với server
Giao diện khởi động sẽ trông như thế này:



Hình 2. Client khởi động

Người dùng lấy IP Address do server cung cấp nhập vào textfield Server Address
Người dùng chọn 1 tên đẻ kết nối với Server sau đó click **Login**
Nếu IP không đúng hoặc máy đang sử dụng không kết nối mạng sẽ hiển thị thông báo lỗi khi kết nối như sau.



Hình 3. Lỗi khi Login

3. Thực hiện xây dựng ứng dụng

a. Phía Server

Bao gồm 2 Class:

- Server.java
- Connected.java

Class Server tiến hành mở Socket và chờ cho các Client kết nối đến
Chứa các đối tượng giao diện Java Swing
Class Connected extends Thread chạy luồn xử lý khi có 1 Client kết nối
Có thể có nhiều Client kết nối tới server do có nhiều luồn thực thi

Class Server chứa các thuộc tính sau :

private ServerSocket server;

Thuộc tính này dùng để tạo ServerSocket

public Hashtable<String, Connected> listUser;

Đây là 1 cấu trúc dữ liệu dùng để lưu tên các Client kết nối với Server có key kiểu String là tên đăng nhập của Client khi Login

Class Server chứa các Phương thức quan trọng sau :

private void client_conected();

Phương thức để mở Socket với port định sẵn

private String getIP();

Phương thức trả về IP Address của Server và hiển thị lên giao diện
Client muốn kết nối đến cần phải biết địa chỉ IP của server

public void sendAll(String from, String msg);

Phương thức gửi gói tin tới tất cả các Client nhận vào 2 String from, msg là tên của Client gửi đến và tin nhắn Client muốn hiện thị cho các Client Khác
Server nhận gói tin từ 1 Client gửi đến thông qua DataStream trình bày trên và gửi lại cho tất cả Client khác để hiển thị lên giao diện

public String getAllName();

Phương thức lấy danh sách tên các Client còn đang kết nối

public void sendAllUpdate(String from);

Phương thức cập nhật danh sách Client còn đang kết nối với Server. Hiển thi trên TextArea của Client

Class Connected chứa các Phương thức quan trọng sau :

private void logout();

Xử lý khi người dùng đăng nhập trùng tên với 1 tên khác đã đăng nhập vào Server
Đóng các dòng DataStream

private void exit();

Xử lý khi người dùng A thoát khỏi ứng dụng
Phương thức này sẽ gửi thông báo đến server và server gửi lại thông cho tất cả Client là người dùng A đã thoát ra

private boolean checkNick(String nick);

Kiểm tra xem tên đăng nhập đã có trong sanh sách hay chưa

```
private void sendMSG(String data) {
    try {
        dos.writeUTF(data);
        dos.flush();
    } catch (IOException ex) {
        Logger.getLogger(Connected.class.getName()).log(Level.SEVERE, null,
ex);
    }
}
```

Gửi thông tin đến Client thông qua đối tượng DataOutputStream bằng phương thức writeUTF(data)

```
public void sendMSG(String msg1, String msg2) {
    sendMSG(msg1);
    sendMSG(msg2);
}
```

msg1 là thông báo để biết msg2 là loại tin nhắn nào, Client dùng cấu trúc switch case để điều hướng hiện thị cho msg2

```
private String getMSG() {
    String data = null;
    try {
        data = dis.readUTF();
    } catch (IOException ex) {
        Logger.getLogger(Connected.class.getName()).log(Level.SEVERE, null,
ex);
    }
    return data;
}
```

Phương thức nhận thông tin từ Client gửi đến thông qua đối tượng DataInputStream

b. Phía Client

Bao gồm 2 Class:

- Clientlog.java
- DataStream.java

Class Clientlog tạo kết nối đến Socket thông qua IP và port mà Server cung cấp
Chứa các đối tượng giao diện Java Swing
Class DataStream cung cấp các phương thức xử lý DataStream

Class Clientlog chứa các Phương thức quan trọng sau:

```
public void go(String SVaddress) {
    try {
        client = new Socket(SVaddress,2422);
        dos = new DataOutputStream(client.getOutputStream());
        dis = new DataInputStream(client.getInputStream());

    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "ERROR, please checkout you
connection or \nEnter a valid IP address", "Message Dialog",
JOptionPane.WARNING_MESSAGE);
        System.exit(0);
    }
}
```

Phương thức khởi tạo Socket kết nối và tạo các dòng DataStream tương tác lẫn nhau

```
private void sendMSG(String data) {
    try {
        dos.writeUTF(data);
        dos.flush();
    } catch (IOException ex) {
        Logger.getLogger(Clientlog.class.getName()).log(Level.SEVERE, null,
ex);
    }
}
```

Phương thức gửi thông tin đến Server thông qua đối tượng DataOutputStream

```
private String getMSG() {
    String data = null;
    try {
        data = dis.readUTF();
    } catch (IOException ex) {
        Logger.getLogger(Clientlog.class.getName()).log(Level.SEVERE, null,
ex);
    }
    return data;
}
```

Phương thức nhận thông tin từ Server gửi đến thông qua đối tượng DataInputStream

```
public void getMSG(String msg1, String msg2) {
    int stt = Integer.parseInt(msg1);
    switch (stt) {
        case 3:
            this.taCom.append(msg2);
            break;
        case 4:
            this.taOnline.setText(msg2);
            break;
        case 5:
            dataStream.stopThread();
            exit();
            break;
        default:
            break;
    }
}
```

Phương thức xử lý tín hiệu và hiển thị thông tin

4. Đánh giá

a. Ưu điểm

- Giao diện ưa nhìn
- Dễ sử dụng
- Có thể giao tiếp nhiều Client với nhau

b. Chưa thực hiện được

- Chưa hiện thực được chức năng gửi nhà nhận File như mong muốn
- Nhóm hoạt động chưa tốt trễ nải tiến độ hoàng thành