TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH



ĐỒ ÁN MÔN NGÔN NGỮ LẬP TRÌNH JAVA

**ĐỀ TÀI: ỨNG DỤNG LAN CHÁT**

Lớp: SE330.H21

**GVLT: Huỳnh Tuấn Anh**

**GVTH: Thái Thị Hoàng Uyên**

**NHÓM SINH VIÊN THỰC HIỆN:**

1. Hoàng Nhật Bảo - MSSV: 14520050
2. Diệp Quang Nhân - MSSV: 14520619
3. Trần Thị Ngọc Hân - MSSV: 14520251

TP. HỒ CHÍ MINH, THÁNG 06 NĂM 2017

Mục lục

[CHƯƠNG 1. TỔNG QUAN 4](#_Toc485437397)

[1.1 Giới thiệu chung 4](#_Toc485437398)

[1.2 Giới thiệu đề tài 4](#_Toc485437399)

[CHƯƠNG 2. Ý TƯỞNG THIẾT KẾ 5](#_Toc485437400)

[2.1 Cách vận hành chương trình 5](#_Toc485437401)

[2.2 Cấu trúc dữ liệu 5](#_Toc485437402)

[CHƯƠNG 3. PHÂN TÍCH VẤN ĐỀ VÀ HƯỚNG GIẢI QUYẾT 6](#_Toc485437403)

[3.1 Xử lý mở kết nối giữa client và server 6](#_Toc485437404)

[3.1.1 Socket 6](#_Toc485437405)

[3.1.2 Thiết lập kết nối 6](#_Toc485437406)

[3.1.3 Cài đặt 6](#_Toc485437407)

[3.2 Xử lý tạo server 7](#_Toc485437408)

[3.3 Xử lý load và tạo phòng chat 9](#_Toc485437409)

[3.4 Xử lý gửi thông tin giữa client và server 9](#_Toc485437410)

[3.4.1 Ý tưởng 9](#_Toc485437411)

[3.4.2 Cài đặt 10](#_Toc485437412)

[3.5 Xử lý lưu thông tin đăng nhập / đăng ký 12](#_Toc485437413)

[3.5.1 Ý tưởng 12](#_Toc485437414)

[3.5.2 Cài đặt 12](#_Toc485437415)

[} 15](#_Toc485437416)

[3.6 Xử lý voice chat 15](#_Toc485437417)

[} 16](#_Toc485437418)

[3.7 Một số chức năng khác 16](#_Toc485437419)

[3.7.1 Đổi font chữ 16](#_Toc485437420)

[3.7.2 Gửi ảnh 17](#_Toc485437421)

[CHƯƠNG 4. HƯỚNG DẪN SỬ DỤNG VÀ ĐÁNH GIÁ CHƯƠNG TRÌNH 18](#_Toc485437422)

[4.1 Hướng dẫn sử dụng 18](#_Toc485437423)

[4.1.1 Cài đặt 18](#_Toc485437424)

[4.1.2 Sử dụng 18](#_Toc485437425)

[4.2 Đánh giá ưu khuyết điểm chương trình 25](#_Toc485437426)

[4.2.1 Ưu điểm 25](#_Toc485437427)

[4.2.2 Khuyết điểm 25](#_Toc485437428)

DANH MỤC HÌNH ẢNH

[Figure 4.1‑1 Giao diện khởi động chương trình chính](#_Toc485420000) 19

[Figure 4.1‑2 Cửa sổ quản lý server 20](#_Toc485420001)

[Figure 4.1‑3 Cửa sổ chọn phòng 2](#_Toc485420002)1

[Figure 4.1‑4 Cửa sổ đăng nhập 2](#_Toc485420003)2

[Figure 4.1‑5 Cửa sổ đăng nhập 2](#_Toc485420003)3

[Figure 4.1‑6 Cửa sổ chat 2](#_Toc485420004)3

[Figure 4.1‑7 Cửa sổ chương trình chat hoạt động 2](#_Toc485420005)4

[Figure 4.1‑8 Giao diện khung chat khi gửi / nhận ảnh. 2](#_Toc485420006)4

[Figure 4.1‑9 Giao diện khi có cuộc gọi tới 2](#_Toc485420007)5

# TỔNG QUAN

## Giới thiệu chung

Ngày nay, trong sự phát triễn của các mạng nội bộ ngày càng rộng lớn như: mạng nội bộ công ty, trường học, tòa nhà, cơ quan hành chính,.. thì nhu cầu sử dụng một hệ thống chat nội bộ, thuận tiện để kết nối mọi người trong công việc ngày càng tăng cao. Với lý do đó, ngày càng nhiều chương trình chat dùng cho mạng Lan ra đời. Điều này giúp ích rất nhiều cho các tổ chức, nội bộ trong việc trao đổi công việc, gửi nhận file, làm việc nhóm.

Ứng dụng Lan Chat sử dụng những kiến thức cơ bản trong lập trình socket, các giao thức để truyền tín hiệu giữa máy khách thông qua máy chủ. Đây cũng là những kiến thức nền tảng cho sinh viên ngành công nghệ thông tin, cần phải biết để trau dồi các kỹ năng chuyên sâu cho ngành công nghệ thông tin.

Ngoài ra, xây dựng ứng dụng lan chat trên nền tảng java là một cách tốt để các cá nhân hoàn thiện kỹ năng lập trình. Java còn là ngôn ngữ đang được thế giới đánh giá cao về tính ứng dụng và tính mới, vì vậy, sử dụng thành thạo java là lợi thế cho lập trình viên.

## Giới thiệu đề tài

Tên đề tài: xây dựng ứng dụng lan chat trên nền tảng ngôn ngữ java.

Chức năng chính của ứng dụng lan chat:

* Đăng ký tài khoàn / đăng nhập (thông tin đăng nhập được lưu trữ trên máy chủ).
* Tạo phòng chat cho từng nhóm riêng biệt.
* Gửi text.
* Gửi file.
* Voice chat.

# Ý TƯỞNG THIẾT KẾ

## Cách vận hành chương trình

Người dùng sau khi khởi động chương trình sẽ xuất hiện các tùy chọn:

* Tạo phòng mới, lúc này máy của người dùng hiện hành sẽ trở thành máy chủ, cho phép các máy khách thông qua nó mà truyền tín hiệu cho nhau. Khi máy này đóng, đồng thời phòng chat đã được tạo bị mất.
* Đăng nhập. Khi chọn chức năng đăng nhập, danh sách các phòng chat đã được tạo trước đó sẽ xuất hiện. Người dùng chọn phòng muốn tham gia, cửa sổ đăng nhập xuất hiện để bắt đầu tiến trình đăng nhập hoặc đăng ký tài khoản mới.
* Thoát. Tắt chương trình hiện hành.

Sau khi đã đăng nhập vào phòng chat, người dùng có thể thực hiện:

* Xem danh sách các người dùng khác đang online trong phòng
* Gửi text cho các máy khác cùng chung một phòng. Text nhận được từ các máy khách sẽ được định danh bằng nickname của người gửi.
* Chọn font chữ cho văn bản mình gởi (chưa gởi được kiểu font xang các máy khách)
* Gửi file, khi một máy khách muốn chia sẽ file, họ upload một file lên server, tại các máy khách còn lại, một thông báo xuất hiện, nếu người dùng đồng ý nhận file thì chọn chỗ lưu file, rồi tải file từ máy chủ về.
* Voice chat, khi một máy khách bắt đầu cuộc gọi nhóm, tại các máy khách khác, cửa sổ gọi xuất hiện, người dùng có thể bắt máy hoặc bỏ qua. Khi bắt máy người dùng có thể trò chuyện với người gọi đến khi một trong các bên gác máy.

## Cấu trúc dữ liệu

Chương trình sử dụng một database lưu trữ thông tin đăng ký tài khoán của người dùng, bao gồm: nick name, password đã được mã hóa md5, ngày đăng ký.

Tín hiệu truyền giữa các máy khách là các DataStream và FileStream. Có sử dụng bộ đệm trong quá trình truyền nhận file và voice chat.

Để phân biệt các loại tín hiệu truyển giữa máy chủ và máy khách, chương trình dùng tập hợp các ký tự số.

Ngoài ra, còn một số cấu trúc dữ liệu đặc biệt để quản lý các thông tin như: danh sách online, danh sách phòng được tạo như: list, bảng băm.

# PHÂN TÍCH VẤN ĐỀ VÀ HƯỚNG GIẢI QUYẾT

## Xử lý mở kết nối giữa client và server

### Socket

Socket là một model của Java, dùng để tạo ra các giao tiếp giữa các máy tính với nhau. Nó là điểm giao tiếp (endpoint) khi thực hiện quá trình liên kết truyền nhận dữ liệu

Socket có thể có 2 luồng: một luồng dữ liệu vào (input stream), và 1 luồng dữ liệu ra (output stream). Máy này gửi dữ liệu cho máy khác thông qua đường mạng, tạo một output stream “đính” với cái socket, và máy tính nhận thì cần input stream với socket của nó để nhận dữ liệu.

### Thiết lập kết nối

Thiết lập giao tiếp giữa các hệ thống không ở cùng một chỗ bằng cách sử dụng mô hình client/server. Máy server sẽ “lắng nghe” yêu cầu (request) từ phía client bằng con đường network hoặc có thể client và server nằm trên cùng 1 máy. Client sẽ kết nối đến server qua địa chỉ IP và port number. Dựa trên kết nối này, server đọc request của từng client và trả lời (respond) các request đó.

Một trong những kiểu kết nối để tạo giao tiếp là giao thức TCP/IP. TCP (Transmission Control Protocol) tạo một kết nối giữa hai endpoints nghĩa là thiết lập socket. Socket này phải ở tình trạng “mở” trong suốt quá trình truyền nhận dữ liệu. Kết nối TCP bao gồm các thông tin sau:

* IP của client.
* IP của server.
* Port của client.
* Port của server.
* Giao thức truyền (protocol) : TCP/IP.

### Cài đặt

Mỗi khi có 1 client kết nối đến server thì sẽ tạo ra 1 Thread để xử lý cho Client đó. Như vậy nếu có nhiều client cùng muốn kết nối vs server thì cũng cũng không gây xung đột cho server.

Yêu cầu kết nối ở client:

Socket client**;**

DataOutputStream dos**;**

DataInputStream dis**;**

client **=** **new** Socket**(**ip**,** port**);**

dos**=new** DataOutputStream**(**client**.**getOutputStream**());**

dis**=new** DataInputStream**(**client**.**getInputStream**());**

Khi server tiếp nhận yêu cầu kết nối của client:

server **=** **new** ServerSocket**(this.**port**);**

Socket client **=** server**.**accept**();**

**new** ServerThread**(this,**client**);**

ServerThread**:**

DataOutputStream dos**;**

DataInputStream dis**;**

dos**=** **new** DataOutputStream**(**client**.**getOutputStream**());**

dis**=** **new** DataInputStream**(**client**.**getInputStream**());**

## Xử lý tạo server

Khi người dùng chọn tạo Server trong màn hình chính của ứng dụng 1 luồng Thread sẽ được tạo ra, luồng này chạy liên tục trong vòng lặp While( true) để gởi địa chỉ IP của máy người dùng đi tất cả các địa chỉ IP khác trong cùng mạng nội bộ. Việc gởi IP thươc hiện bằng DatagramSocket (truyền tin theo UDP) và được thực hiện trên port cố định, mỗi server được chạy sẽ tự động tìm một port riêng ( từ 1026 đến 65025) hỗ trợ một máy có thể tạo nhiều server cùng một lúc. Giới hạn của ứng dụng chỉ dừng lại ở mạng lớp C 192.168.x.x vì vòng lặp phải gởi IP đến tất cả IP trong cùng mạng nên nếu làm lớn hơn dễ gây treo máy, nhưng với giới hạn đó cơ bản cũng đủ mạnh để kết nối tối đa 255 người dùng, và mỗi người dùng trong trường hợp mỗi người đều tạo full server có thể kết nối vào 16581375 server cùng một lúc. Có thể phát triển thêm việc tự nhập địa chỉ IP để tránh chạy vòng lặp, gây tốn bộ nhớ máy.

//send my ip

Thread sendip **=** **new** Thread**(new** Runnable**()** **{**

@Override

public void run**()** **{**

DatagramPacket dp**;**

String host**;**

int i**;**

**try** **{**

InetAddress localHost **=** InetAddress**.**getLocalHost**();**

NetworkInterface networkInterface **=** NetworkInterface**.**getByInetAddress**(** localHost**);**

DatagramSocket ds **=** **new** DatagramSocket**();**

String MessString **=** findIPconnect**()** **+** "+" **+** Integer**.**toString**(**newport**)** **+** "+" **+** NameServer**.**getText**();**

String ia **=** findIPconnect**();**

byte**[]** MessByte **=** MessString**.**getBytes**();**

int Length **=** MessString**.**length**();**

**if(**networkInterface**.**getInterfaceAddresses**().**get**(**0**).**getNetworkPrefixLength**()** **==** 24**)**

**{**

**for(**i**=**ia**.**length**()-**1**;** i**>**0 **;**i**--)**

**{**

**if(**ia**.**charAt**(**i**)** **==** '.'**)**

**{**

ia**=**ia**.**substring**(**0**,** i**+**1**);**

i**=**0**;**

**}**

**}**

**while(true)**

**{**

**for(**i**=**1 **;**i**<**255 **;**i**++){**

host **=** ia **+** i**;**

dp **=** **new** DatagramPacket**(**MessByte**,** Length**,** InetAddress**.**getByName**(**host**),** 1025**);**

ds**.**send**(**dp**);**

**}**

**}**

**}**

**else{**

/\*\*

\*Mở rông thêm đồ án cho mạng lớp A và B hố trợ nhiều \*subnet \*Mark hơn

\*/

**}**

**}** **catch** **(**IOException ex**)** **{**

System**.**out**.**println**(**ex**);**

**}**

**}**

**});**

sendip**.**start**();**

Việc gởi IP đến tất cả các IP khả dụng trong mạng nội bộ, được thực hiện trên port cố định 1025 vì tất cả các port khả dụng trong một máy là khoảng 64000 (trừ port của hệ thống) port, nên mỗi lần tạo port là mỗi lần phải sử dụng một địa chỉ mới chạy từ 1026 đến 65025. Địa chỉ port riêng cho mỗi server được tạo cũng giúp việc chat rõ ràng hơn, vì các thành viên chat ở mỗi server khác nhau là mỗi port khác nhau nên không gây mâu thuẫn dữ liệu dù một máy có chạy nhiều server

Sau khi gởi IP của mình đến cho các client kết nối, server sẽ tạo một Socket để lắng nghe các client. Socket này sẽ lắng nghe trên địa chỉ IP đã kết nối 2 máy và port của IP đang chạy. Vì phải liên tục lắng nghe nên server phải được chạy trong một luồng riêng.

## Xử lý load và tạo phòng chat

Vì server khi được tạo sẽ gởi địa chỉ của mình một cách liên tục theo giao thức truyền tin UDP, nên khi người dùng load room để tìm danh sách phòng chat khả dụng để có thể đăng nhập vào thì ứng dụng sẽ tạo một DatagramSocket để nhận vào trên port cố định 1025, việc lắng nghe được thực hiện trong 10000 miliseccond, trong trường hợp bắt được tín hiệu từ server đến IP của client setsoTimeout(10000) được đặt lại, và lại tiếp tục lắng nghe cho tới khi không còn nhận được IP mới hoặc load được.

Sau khi load được danh sách các phòng có thể đăng nhập người dùng có thể chọn 1 phòng theo tên phòng để tiến hành giao tiếp với các người dùng khác thông qua server. Một button được load trong danh sách phòng chứa thông tin về tên phòng, bên trong đó là IP và port của server đã tạo phòng đó giúp client nhận biết và kết nối tới server đó.

## Xử lý gửi thông tin giữa client và server

### Ý tưởng

Khi các client đã kết nối tới server, server sẽ đóng vai trò truyền tín hiệu từ một máy client bất kì cho các máy còn lại trong cùng một room.

Khi client muốn gửi gửi tin nhắn đến cho các client còn lại, nó gửi cho server một tín hiệu đầu tiên thống báo loại dữ liệu à nó sắp gửi đến. Tín hiệu này là một con số, mỗi con số đại diện cho một loại dữ liệu riêng biệt, trong chương trình sử dụng các tín hiệu số sau đây để biểu diễn thông tin gửi từ client cho server:

* 11: Client đăng nhập. Sau khi server nhận tín hiệu này, nó đọc thêm 2 thông tin từ DataInputStream bao gồm: nickname và password đã được mã hóa.
* 10: Client đăng ký một tài khoản mới. Tương tự như 11, server cũng đọc thêm 2 thông tin là nickname và password để đăng ký của client.
* 0: Client đã thoát khỏi room.
* 1: Client vừa gửi một tin nhắn thường (text), server có nhiệm vụ gửi cho tất cả client còn lại trong room.
* 2: client vừa gửi một file, server sẽ đọc tiếp trong DataInputStream thông tin file size và file name. Rồi truy cập vào FileOutputStream để đọc file lưu lại vào máy chủ, sau đó gửi cho các máy khách còn lại trong room.
* 20: Client vừa bắt đầu một voice chat, server sẽ có nhiệm vụ thông báo tới các client còn lại trong room.
* 40: client vừa gửi một file ảnh cho server.

Tương tự như cách trên, server cũng dùng tín hiệu số để thông báo cho client các thông tin mà nó sắp gửi cho client, đại diện như sau:

* 3: server gửi tin nhắn text thứ máy một máy khách bất kì trong room, client lúc này phải đọc tin nhắn đó trong DataInputStream rồi biểu diễn lại nó ra giao diện.
* 6: server sắp gửi một file từ máy khách khác trong room, tiếp theo, client xuất thông báo cho người dùng có muốn lưu file không.
* 21: server gửi thông tin có người vừa bắt đầu voice chat, client sẽ hiện thông báo để người dùng có tiếp nhận cuộc gọi hay không.
* 41: server vừa gửi một ảnh từ một client cho nhóm, ảnh sẽ xuất hiện trên khung chat.

### Cài đặt

Server sẽ luôn trong tình trạng lắng nghe client:

**while(**run**){**

//tiếp nhận thông tin

**}**

Xây dựng một hàm đọc tín hiệu từ client, trả về thông tin đọc được từ DataInputStream của server:

String getMSG**(){**

String data**=null;**

**try** **{**

data**=**dis**.**readUTF**();**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**return** data**;**

**}**

Trong trường hợp dữ liệu là file:

private void getFile**(){**

**try{**

FileOutputStream fos **=** **new** FileOutputStream**(**fileName**);**

byte**[]** buffer **=** **new** byte**[**1024**];**

int read **=** 0**;**

long remaining **=** fileSize**;**

**while((**read **=** dis**.**read**(**buffer**,** 0**,** Math**.**min**(**buffer**.**length**,** **(**int**)**remaining**)))** **>** 0**)** **{**

remaining **-=** read**;**

fos**.**write**(**buffer**,** 0**,** read**);**

**}**

fos**.**close**();**

**}**

**catch** **(**Exception e**)**

**{**

e**.**printStackTrace**();**

**}**

**}**

Xây dựng một hàm gửi tín hiệu từ server đến client thông qua DataOutputStream:

public void sendMSG**(**String data**){**

**try** **{**

System**.**out**.**println**(**data**);**

dos**.**writeUTF**(**data**);**

System**.**out**.**println**(**dos**);**

dos**.**flush**();**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

Hàm này có thể overwrite nhiều kiểu dữ liệu cho phù hợp với nhu cầu của chương trình. Trong trường hợp gửi file:

public void sendFile**(**String fileName**,** long fileSize**){**

**try{**

File file **=** **new** File**(**fileName**);**

FileInputStream fis **=** **new** FileInputStream**(**file**);**

byte**[]** buffer **=** **new** byte**[**1024**];**

**while** **(**fis**.**read**(**buffer**)** **>** 0**)** **{**

dos**.**write**(**buffer**);**

**}**

fis**.**close**();**

**}**

**catch(**Exception e**){**

e**.**printStackTrace**();**

**}**

**}**

Hàm getMSG, sendMSG, getFile, sendFile được thiết lập tương tự ở client side.

## Xử lý lưu thông tin đăng nhập / đăng ký

### Ý tưởng

Ngoài cung cấp ip và port cho server để mở một kết nối, chương trình yêu cầu người dùng gửi đến một định danh cho cuộc trò chuyện và mật khẩu đã đăng ký.

Để thực hiện đăng ký, người dùng phải cung cấp nick name, password, việc cung cấp thông tin này giúp những người trong cùng cuộc hội thoại dễ dàng nhận ra nhau.

Server phải lưu trữ được các thông tin do người dùng cung cấp. Đảm bảo rằng tài khoản đã tạo là hợp lệ, tức là:

* Nickname phải là duy nhất.
* Mật khẩu tối thiểu phải có 8 ký tự.

Sau khi đã có một tài khoản, vào lần sử dụng chương trình tiếp theo, người dùng chỉ cần cung cấp nickname và password hợp lệ, tức là:

* Tài khoản này đã được đăng ký trước đó.
* Chỉ có một người duy nhất được login với 1 tài khoản.

### Cài đặt

Java JDBC là một Java API để kết nối và thực thi truy vấn với cơ sở dữ liệu. JDBC API sử dụng các JDBC Driver để kết nối với cơ sở dữ liệu. JDBC làm việc với Java trên nhiều nền tảng đa dạng như Windows, Mac OS, Unix, …

Chương trình xây dựng một cơ sở dữ liệu trên sqlite, gồm bảng User (id user, nickname, password, date).

Kết nối với database:

Connection conn **=** **null;**

Statement stmt **=** **null;**

**try** **{**

Class**.**forName**(**"org.sqlite.JDBC"**);**

conn **=** DriverManager**.**getConnection**(**"jdbc:sqlite:wechat.db"**);**

stmt **=** conn**.**createStatement**();**

System**.**out**.**println**(**"Connection Success"**);**

**}** **catch** **(**ClassNotFoundException **|** SQLException **|** HeadlessException e**)** **{**

System**.**out**.**println**(**"Connection failed: "**+**e**);**

JOptionPane**.**showMessageDialog**(this,** "Faile to connect database"**,** "Message Dialog"**,** JOptionPane**.**WARNING\_MESSAGE**);**

**}**

Khi có thông báo client muốn đăng ký tài khoản, server sẽ kiểm tra xem tên đang nhập của họp có lợp lý hay không:

boolean checkUserName**(**String nickName**){**

**try{**

String sql **=** "SELECT \* FROM user WHERE nickName=?"**;**

pstmt **=** conn**.**prepareStatement**(**sql**);**

pstmt**.**setString**(**1**,** nickName**);**

ResultSet rs **=** pstmt**.**executeQuery**();**

System**.**out**.**println**(**rs**.**getFetchSize**());**

**if(**rs**.**next**())** **return** **false;**

**return** **true;**

**}catch(**Exception ex**){**

ex**.**printStackTrace**();**

**}**

**return** **false;**

**}**

Nếu kiểm tra hợp lệ, thì thực hiện thêm vào database:

boolean insertUser**(**String nickName**,** String passWord**){**

**if(**checkUserName**(**nickName**)){**

**try{**

System**.**out**.**println**(**passWord**);**

DateFormat dateFormat **=** **new** SimpleDateFormat**(**"YYYY-MM-dd"**);**

java**.**util**.**Date date **=** **new** java**.**util**.**Date**();**

String sql **=** "INSERT INTO user(nickName, passWord, date) VALUES(?,?,?) "**;**

pstmt **=** conn**.**prepareStatement**(**sql**);**

pstmt**.**setString**(**1**,** nickName**);**

pstmt**.**setString**(**2**,** passWord**);** pstmt**.**setDate**(**3**,**Date**.**valueOf**(**dateFormat**.**format**(**date**)));**

pstmt**.**executeUpdate**();**

**return** **true;**

**}catch** **(**SQLException e**){**

System**.**out**.**println**(**"Insert to user failed: "**+**e**.**getMessage**());**

**}**

**}**

**return** **false;**

**}**

Khi có client yêu cầu đăng nhập, server sẽ kiểm tra tính hợp lệ của tài khoản:

boolean checkUser**(**String nickName**,** String passWord**){**

**try{**

String sql **=** "SELECT \* FROM user WHERE nickName=? and passWord=?"**;**

pstmt **=** conn**.**prepareStatement**(**sql**);**

pstmt**.**setString**(**1**,** nickName**);**

pstmt**.**setString**(**2**,** passWord**);**

ResultSet rs **=** pstmt**.**executeQuery**();**

**while** **(**rs**.**next**())** **{**

**return** **true;**

**}**

**return** **false;**

**}catch(**Exception ex**){**

ex**.**printStackTrace**();**

**}**

**return** **false;**

## }

## Xử lý voice chat

Khi thực hiện chức năng voice chat client thực hiện gửi tín hiệu “20” lên sever và bật audio. Sever gửi tín hiệu “21” đến các client thông báo có cuộc voice chat nếu chấp nhận cuộc gọi thì bật audio.

Audio:

public static AudioFormat getaudioformat**(){**

float sampleRate **=** 8000.0F**;**

int sampleSizeInbits **=** 16**;**

int channel **=** 2**;**

boolean signed **=** **true;**

boolean bigEndian **=** **false;**

**return** **new** AudioFormat**(**sampleRate**,** sampleSizeInbits**,** channel**,** signed**,** bigEndian**);**

**}**

Bật audio lấy dữ liệu (nói):

public void init\_audio**(){**

**try** **{**

AudioFormat format **=** getaudioformat**();**

DataLine**.**Info info **=** **new** DataLine**.**Info**(**TargetDataLine**.**class**,** format**);**

**if(!**AudioSystem**.**isLineSupported**(**info**)){**

System**.**out**.**println**(**"not suport"**);**

System**.**exit**(**0**);**

**}**

audio\_in **=** **(**TargetDataLine**)** AudioSystem**.**getLine**(**info**);**

audio\_in**.**open**(**format**);**

audio\_in**.**start**();**

Client\_CV r **=** **new** Client\_CV**();**

InetAddress inet **=** InetAddress**.**getByName**(**add\_server**);**

r**.**audio\_in **=** audio\_in**;**

r**.**dout **=** **new** DatagramSocket**();**

r**.**server\_ip **=** inet**;**

r**.**server\_port **=** port**;**

runCalling **=** **true;**

r**.**start**();**

VoiceChat**.**setEnabled**(false);**

ENDVC**.**setEnabled**(true);**

**}** **catch** **(**LineUnavailableException **|** UnknownHostException **|** SocketException ex**)** **{**

**}**

**}**

Bật audio xuất dữ liệu (nghe):

public void init\_audio1**(){**

**try** **{**

AudioFormat format **=** getaudioformat**();**

DataLine**.**Info info\_out **=** **new** DataLine**.**Info**(**SourceDataLine**.**class**,** format**);**

**if(!**AudioSystem**.**isLineSupported**(**info\_out**)){**

System**.**out**.**println**(**"not suport"**);**

System**.**exit**(**0**);**

**}**

audio\_out **=** **(**SourceDataLine**)**AudioSystem**.**getLine**(**info\_out**);**

audio\_out**.**open**(**format**);**

audio\_out**.**start**();**

Client\_CV\_Out p **=** **new** Client\_CV\_Out**();**

p**.**din **=** **new** DatagramSocket**(**port**);**

p**.**audio\_out **=** audio\_out**;**

runCalling **=** **true;**

p**.**start**();**

**}** **catch** **(**LineUnavailableException **|** SocketException ex**)** **{**

**}**

## }

## Một số chức năng khác

### Đổi font chữ

Jfontchooser là một thư viện mã nguồn mở, cung cấp các chức năng thay đổi format của text trong các control hiển thị text của java swing. Để sử dụng Jfontchooser, cần download jfontchooser.jar, thêm nó vào project hiện hành.

**import** javax**.**swing**.**JFileChooser**;**

**import** javax**.**swing**.**filechooser**.**FileNameExtensionFilter**;**

Khi người dùng nhấn chọn chức năng đổi font, khởi tạo một đối tượng JfontChooser:

JFontChooser fontChooser **=** **new** JFontChooser**();**

Sau đó xử lý bắt sự kiện đóng hộp thoại của người dùng:

int result **=** fontChooser**.**showDialog**(this);**

**if** **(**result **==** JFontChooser**.**OK\_OPTION**)**

**{**

Font font **=** fontChooser**.**getSelectedFont**();**

**this.**txtMSG**.**setFont**(**font**);**

**}**

### Gửi ảnh

Chức năng gửi ảnh tương tự như chức năng truyền file, tuy nhiên, người dùng có thể xem trực tiếp trên cửa sổ chat.

Chọn ảnh trong một dialog chỉ cho phép người dùng chọn file ảnh, chỉ chấp nhận cái file ảnh nhỏ hơn 2.5M:

JFileChooser j **=** **new** JFileChooser**();**

j**.**setFileSelectionMode**(**JFileChooser**.**FILES\_ONLY**);**

j**.**setFileFilter**(new** FileNameExtensionFilter**(**"Images"**,** "jpg"**,** "png"**,** "gif"**,** "bmp"**));**

int opt **=** j**.**showOpenDialog**(this);**

**if(**opt **==** JFileChooser**.**APPROVE\_OPTION**){**

String savePath**=**j**.**getSelectedFile**().**getPath**();**

File m**=new** File**(**savePath**);**

**if(**m**.**length**()<=**2500**){**

**this.**getImage**(**savePath**);**

**this.**sendMSG**(**"40"**);**

**this.**sendImage**(**m**);**

**}**

**else** JOptionPane**.**showMessageDialog**(this,**"Ảnh quá lớn, vui lòng chọn ảnh có kích thước nhỏ hơn 2.5M"**,**"Message Dialog"**,**JOptionPane**.**WARNING\_MESSAGE**);**

**}**

Để biểu thị ảnh này vừa đủ xem với khung hình, cần scale bằng các phép biến đổi Affine:

public BufferedImage scale**(**BufferedImage sbi**,** int imageType**,** int dWidth**,** int dHeight**,** double fWidth**,** double fHeight**)** **{**

BufferedImage dbi **=** **null;**

**if(**sbi **!=** **null)** **{**

dbi **=** **new** BufferedImage**(**dWidth**,** dHeight**,** imageType**);**

Graphics2D g **=** dbi**.**createGraphics**();**

AffineTransform at **=** AffineTransform**.**getScaleInstance**(**fWidth**,** fHeight**);**

g**.**drawRenderedImage**(**sbi**,** at**);**

**}**

**return** dbi**;**

**}**

Biểu diễn nó lên JtextPane:

StyledDocument doc **=** txtMSG**.**getStyledDocument**();**

SimpleAttributeSet style **=** **new** SimpleAttributeSet**();**

StyleConstants**.**setIcon**(**style**,** **new** ImageIcon**(**img**));**

StyleConstants**.**setAlignment**(**style**,** StyleConstants**.**ALIGN\_RIGHT**);**

doc**.**insertString**(**doc**.**getLength**(),** "\n"**,** style**);**

Một số control được sử dụng

Control hiển thị tin nhắn: JtextPane.

Control hiển thị hộp thoại chọn file/path: JfileChooser.

Một số control phổ biến: JtextFiled, Buttom, Label, JPanel

# HƯỚNG DẪN SỬ DỤNG VÀ ĐÁNH GIÁ CHƯƠNG TRÌNH

## Hướng dẫn sử dụng

### Cài đặt

WeChat là ứng dụng portable, yêu cầu hệ thống để chạy chương trình này:

* Java phiên bản tối thiểu:

Để khởi động chương trình:

* Window: chạy file wechat.exe
* Ubuntu:
* Với máy bất kì có cài java:

### Sử dụng

Giao diện chương trình chính:



Figure 4.1‑1 Giao diện khởi động chương trình chính

Khi khởi tạo một phòng chat, cửa sổ theo dõi server xuất hiện:

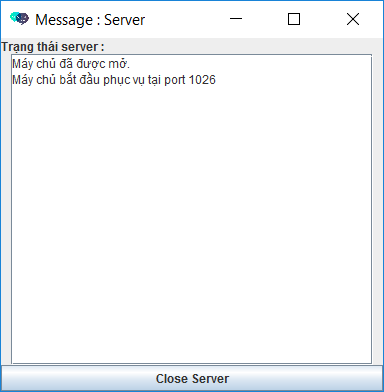


Figure 4.1‑2 Cửa sổ quản lý server

Muốn tắt 1 phòng chat, đóng cửa sổ này quản lý server.

Khi chọn vào chức năng Login, cửa sổ danh sách các phòng xuất hiện cho người dùng chọn.

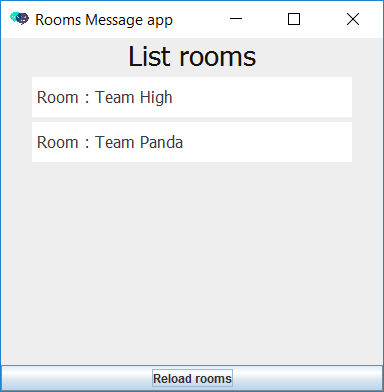


Figure 4.1‑3 Cửa sổ chọn phòng

Khi chọn được phòng thích hợp để login, cửa sổ login xuất hiện:

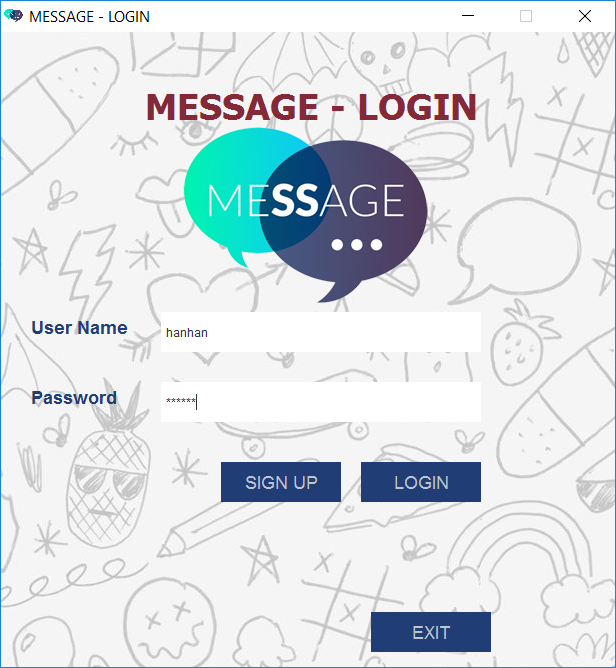


Figure 4.1‑4 Cửa sổ đăng nhập

Người dùng cũng có thể đăng ký tài khoảng mới khi click vào button SIGN UP trên cửa sổ đăng nhập. Một cửa sổ mới sẽ hiện ra cho phép người dùng nhập vào tên tài khoảng và mật khẩu muốn đăng ký



Figure 4.1‑5 Cửa sổ đăng ký

Sau khi đăng nhập hoặc đăng ký thành công, tài khoảng của người dùng sẽ được tham gia vào phòng chat.

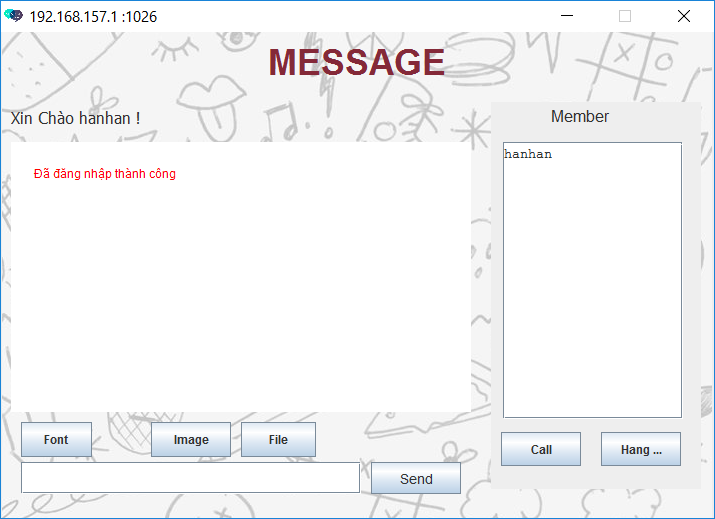


Figure 4.1‑6 Cửa sổ chat

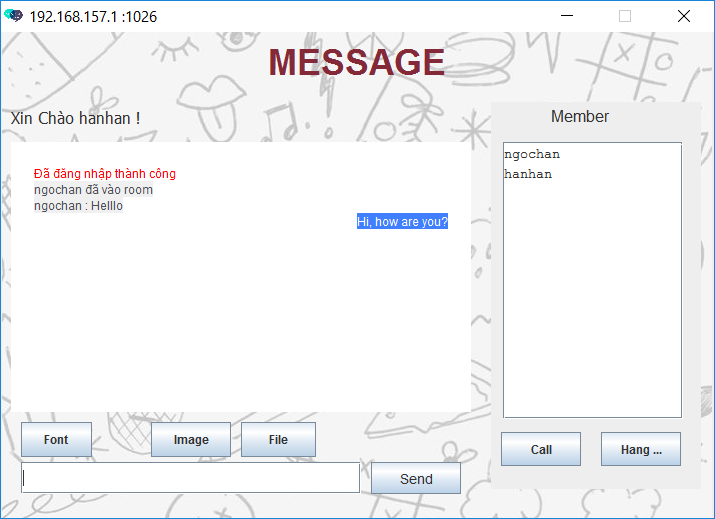


Figure 4.1‑7 Cửa sổ chương trình chat hoạt động

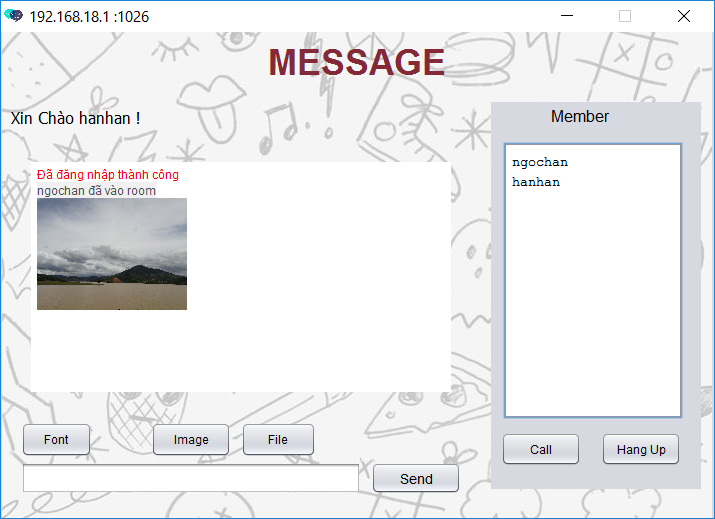


Figure 4.1‑8 Giao diện khung chat khi gửi / nhận ảnh.

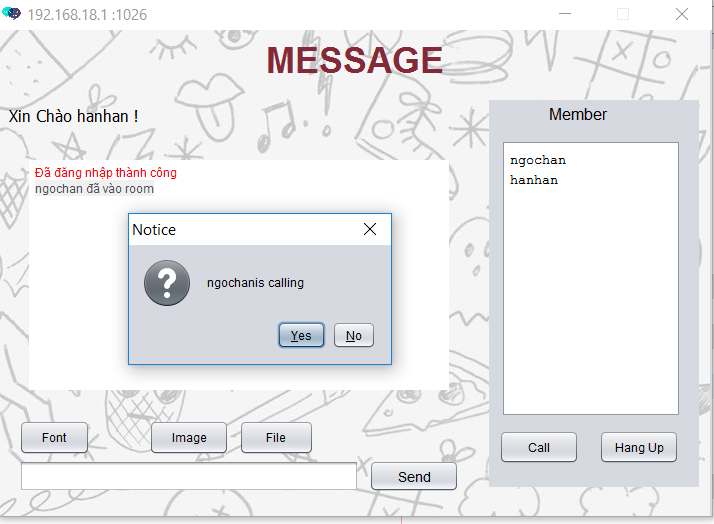


Figure 4.1‑9 Giao diện khi có cuộc gọi tới

## Đánh giá ưu khuyết điểm chương trình

### Ưu điểm

Hoàn thành được mục tiêu ban đầu là xây dựng một ứng dụng trên nền ngôn ngữ Java, trau dồi thêm kiến thức cần có của một lập trình viên.

Chương trình khá đầy đủ chức năng:

* Xử lý đa luồng.
* Xử lý cơ sở dữ liệu nhỏ.
* Xử lý truyền tín hiệu dưới nhiều dạng.

### Khuyết điểm

Ngoài các ưu điểm đạt được, chương trình còn tồn tại một số hạn chế sau:

* Chương trình còn chưa tối ưu về tốc độ.
* Giao diện còn đơn sơ.
* Chưa dùng được các kỹ thuật nâng cao về cơ sở dữ liệu cũng như luồng.

--**HẾT**--