**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN**

**Môn: Lập trình mạng**

**Đề tài:Game oẳn tù tì thi đấu đối kháng online.**

|  |  |
| --- | --- |
| Sinh viên: | Mã sinh viên |
| Nguyễn Đức Thắng  Lê Ngọc Minh Sơn | B14DCAT008  B14DCAT056 |

Hà Nội: 2018

**MỤC LỤC**

[**I.** **Phát biểu bài toán** 3](#_Toc495077371)

[**II.** **Phân tích thiết kế hệ thống.** 3](#_Toc495077378)

[**1.** **Phân tích thiết kế hệ thống hướng đối tượng** 3](#_Toc495077379)

[**2.** **Giao thức nói chuyện giữa client và server** 5](#_Toc495077380)

**III. Cài đặt** **6**

**IV. Kết luận** **8**

[Tài liệu tham khảo 8](#_Toc495077383)

**I.Phát biểu bài toán:**

**Game oẳn tù tì thi đấu đối kháng online**:

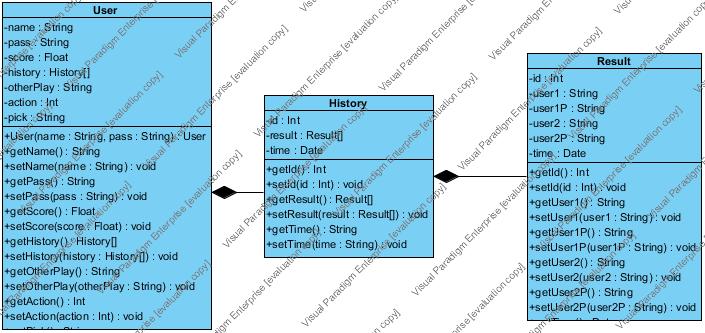
* Hệ thống có một server và nhiều client. Server lưu toàn bộ thông tin và dữ liệu.
* Để chơi, người chơi phải login vào tài khoản của mình từ một máy client. Sau khi login thành công, giao diện hiện lên một danh sách người chơi đang online, mỗi người chơi có các thông tin: tên, tổng số điểm hiện có của người chơi, trạng thái (hoặc đang bận nếu đang chơi với người khác, hoặc đnag rỗi nếu không chơi với ai).
* Muốn mời (thách đấu) ai thì người chơi click vào tên của đối thủ trong danh sách online.
* Khi bị thách đấu, người chơi có thể chấp nhận (OK), hoặc từ chối (Reject).
* Khi chấp nhận, 2 người chơi sẽ vào chơi với nhau, và server sẽ làm trọng tài. Giao diện chơi gồm 3 lựa chọn: giấy, kéo, búa và 2 nút send, thoát.
* Mỗi người chơi chọn 1 trong 3 vật và click vào nút Send. Mỗi nước đi mỗi người chơi có 15s để chọn và gửi lên server.
* Khi cả 2 đối thủ đã click send, server sẽ chấm điểm xem ai thắng và gửi kết quả về cho cả 2 đối thủ: thắng 1 điểm, hòa 0.5 điểm, thua 0 điểm.
* Nếu muốn kết thúc trò chơi với đối thủ hiện tại, người chơi click vào thoát. Hệ thống sẽ thông báo với người còn lại.
* Kết quả các trận đấu được lưu vào server. Mỗi người chơi đều có thể xem bảng xếp hạng các người chơi trong toàn bộ hệ thông, theo lần lượt các tiêu chí: tổng điểm số (giảm dần), tổng số trận đấu (giảm dần).

**II.Phân tích thiết kế hệ thống**

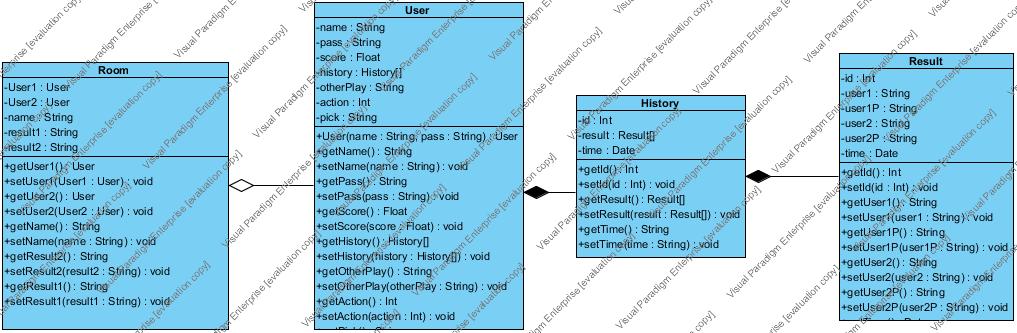
**1.Phân tích thiết kế hệ thống hướng đối tượng**

**a.Sơ đồ lớp thực thể hệ thống**

\_Phía Client:

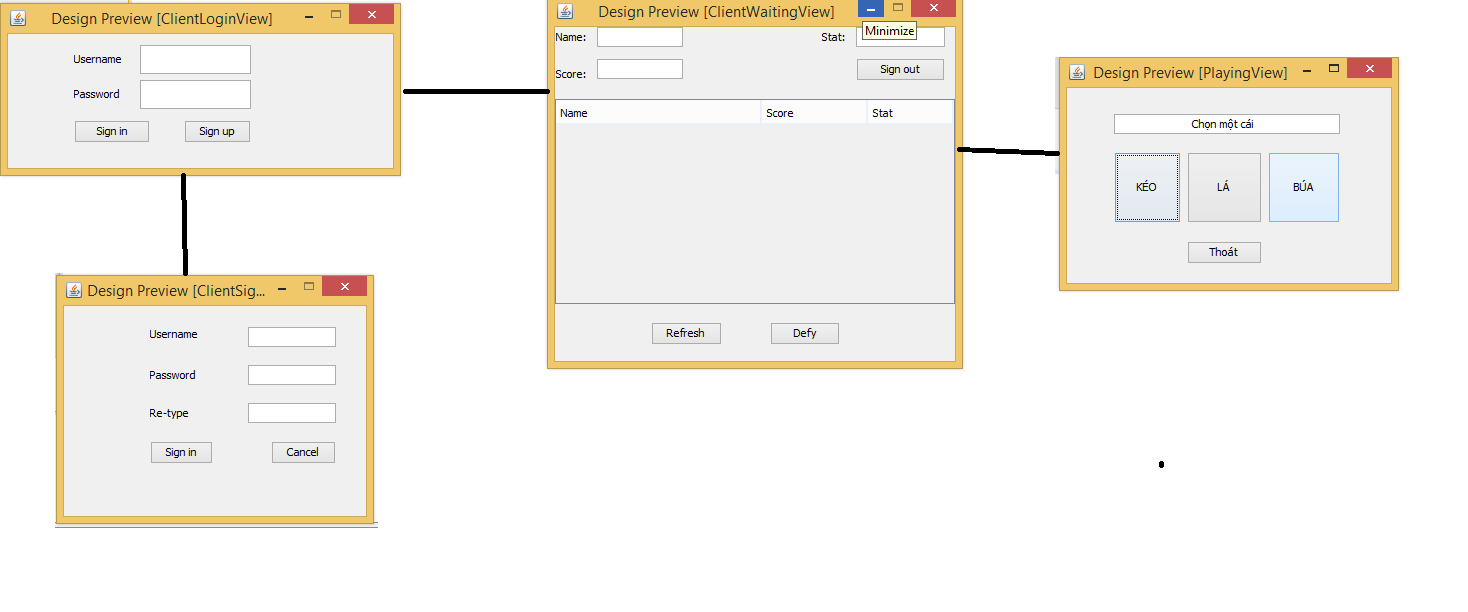


**\_**Phía Server:

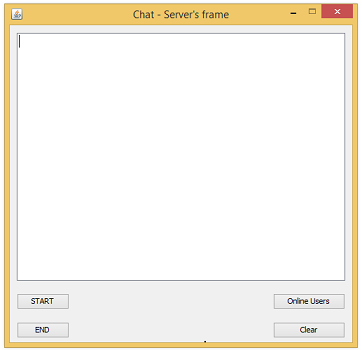


b.Thiết kế tĩnh của hệ thống:

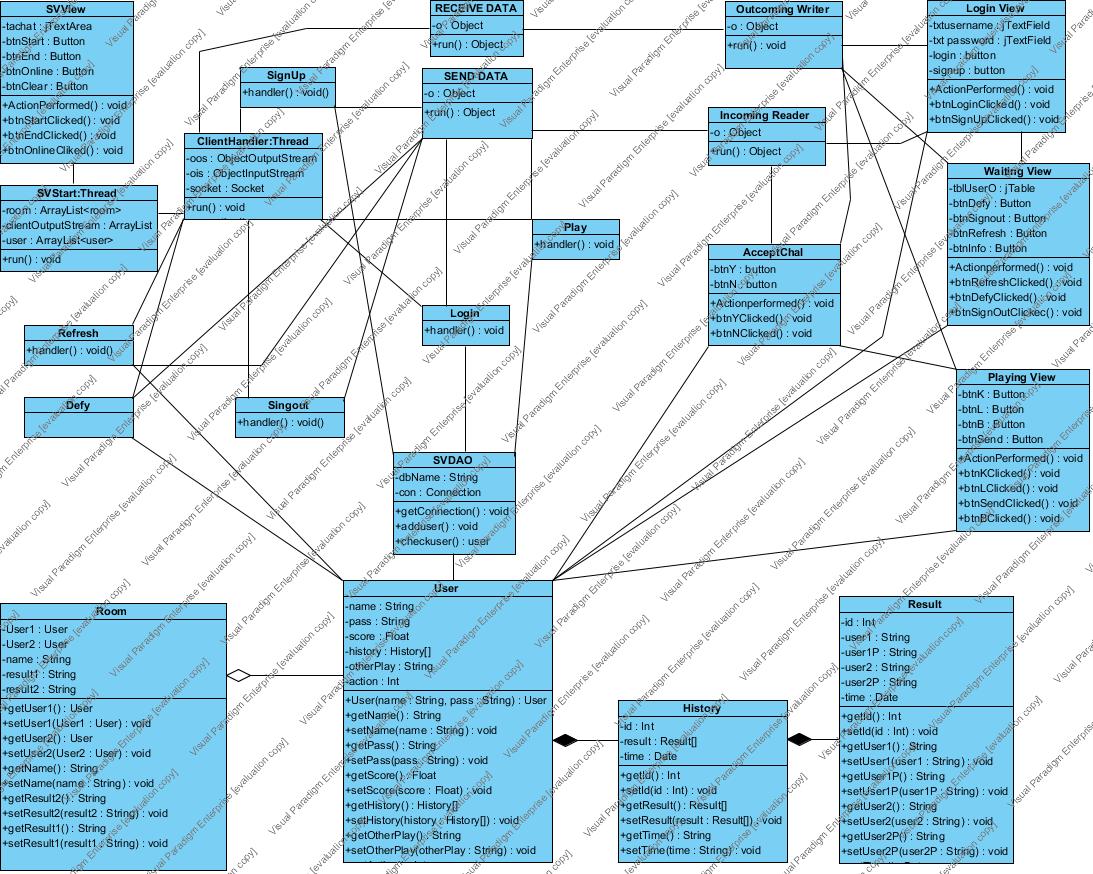
\_Phía Client:



\_Phía server:



**\_Sơ đồ các lớp**



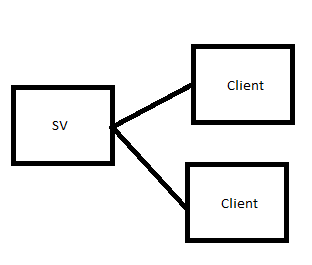
**2.Giao thức nói chuyện giữa Client và Server:TCP/IP**

**\_**Chương trình chạy với multi client nên:

+ Server cần chạy 1 luồn đọc dữ liệu riêng cho mỗi client.

+ Giá trị của mỗi socket từ client sẽ được lưu vào 1 mảng

* Ta có mô hình sau:

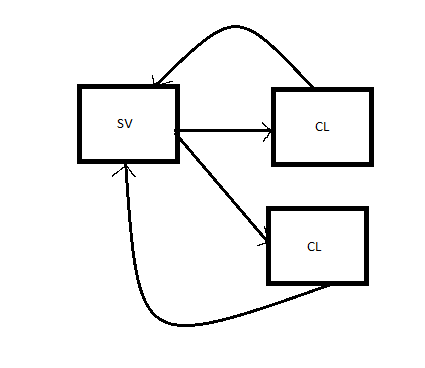


\_Các client có thể tương tác với nhau, ví dụ trường hợp client A gửi client B 1 lời thách đấu thông qua server. Nên:

+Phía Client cũng cần chạy 1 luồng đọc dữ liệu riêng, luôn lắng nghe từ phía server

+Client chỉ được sử dụng duy nhất 1 socket sau khi kết nối tới server.

=>Ta có mô hình như sau:



**III.Cài đặt:**

Các hàm, lớp xử lý quan trọng:

a.Phía Server

\_Class ServerStart: lớp này dùng để mở server, cần lưu ý:

+Sau khi có kết nối từ phía client, ta phải có 1 mảng để lưu các socket từ phía client lại, để xử lý các yêu cầu kết nối giữa các client đnag online với nhau.

clientOutputStreams = new ArrayList();

…..

ObjectOutputStream oos = new ObjectOutputStream(clientSock.getOutputStream();

clientOutputStreams.add(oos);

+Sau đó với mỗi client ta phải tạo 1 luồng truyền dữ liệu riêng, đó là lớp ClientHandler

\_Class ClientHandler (extends Thread): lớp này dùng để lắng nghe dữ liệu từ Client. Các đối tượng gửi lên là các User, sau đó được lớp này bóc tách để lấy giá trị action trong mỗi đối tượng User, sau đó gửi đến các hàm xử lý phù hợp.

\_Các Class Login, SignUp,SignOut,Refresh, Defy: dùng để xử lý dữ liệu từ Client

\_Khi Client A muốn tương tác với Client B mà được B đồng ý, Server sẽ tạo 1 đối tượng Room cho 2 client này. Đối tượng này sẽ có 1 id để SV phân biệt được Client nào đang tương tác với Client nào

\_Sau khi nhận được dữ liệu từ cả 2 phía client, SV thực hiện tính toán và trả về kết quả cho Client. Toàn bộ quá trình tạo room rồi thực hiện kết quả được cài đặt như sau:

if (data[2].equals("playing") == false) {

int vt1 = -1;

int vt2 = -1;

ObjectOutputStream writer1 = null;

ObjectOutputStream writer2 = null;

for (int i = 0; i < users.size(); i++) {

if (data[2].equals(users.get(i))) {

vt1 = i;

}

}

for (int i = 0; i < users.size(); i++) {

if (data[3].equals(users.get(i))) {

vt2 = i;

}

}

if (vt1 != -1 && vt2 != -1) {

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

writer1 = (ObjectOutputStream) clientOutputStreams.get(vt1);

writer2 = (ObjectOutputStream) clientOutputStreams.get(vt2);

System.out.println(writer1);

System.out.println(writer2);

Random rd = new Random();

int checkRoom = 0;

String rdRoom = "1";

while(checkRoom == 0){

checkRoom = 1;

int rdIntRoom = rd.nextInt(1000);

rdRoom = new Integer(rdIntRoom).toString();

for (int i = 0; i < room.size(); i++) {

if (room.get(i).equals(rdRoom)) {

checkRoom = 0;

}

}

}

room.add(new Result(rdRoom));

writer1.writeObject("Play:start:"+data[2]+":"+data[3]+":"+rdRoom);

writer2.writeObject("Play:start:"+data[3]+":"+data[2]+":"+rdRoom);

}

}

if (data[2].equals("playing") == true) {

int vt1 = -1;

int vt2 = -1;

ObjectOutputStream writer1 = null;

ObjectOutputStream writer2 = null;

int j = -1;

for(int i =0 ;i <room.size();i++){

if(data[6].equals(room.get(i).name)){

j = i;

}

}

if(j != -1){ // NẾu vẫn tồn tại cái phòng ý

if(room.get(j).user1.equals("")==true && room.get(j).user2.equals("")==true){

room.get(j).user1 = data[3];

room.get(j).user1P = data[4];

}

if(room.get(j).user2.equals("")==true && data[3].equals(room.get(j).user1) == false){

room.get(j).user2 = data[3];

room.get(j).user2P = data[4];

}

if(data[3].equals(room.get(j).user1) == true && room.get(j).user1P.equals("") == true){

room.get(j).user1P = data[4];

}

if(data[3].equals(room.get(j).user2) == true && room.get(j).user2P.equals("") == true){

room.get(j).user2P = data[4];

}

if(room.get(j).user1.equals("") == false && room.get(j).user2.equals("") == false && room.get(j).user1P.equals("") == false && room.get(j).user2P.equals("") == false ){

room.get(j).XetKetQua();

ta\_chat.append(room.get(j).result1);

ta\_chat.append(room.get(j).result2);

for (int i = 0; i < users.size(); i++) {

if (room.get(j).user1.equals(users.get(i))) {

ta\_chat.append(room.get(j).user1);

ta\_chat.append(users.get(i));

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

vt1 = i;

System.out.println("vt1 = " + vt1);

}

}

for (int i = 0; i < users.size(); i++) {

if (room.get(j).user2.equals(users.get(i))) {

vt2 = i;

ta\_chat.append(room.get(j).user2);

ta\_chat.append(users.get(i));

System.out.println("vt2 = " + vt2);

}

}

writer1 = (ObjectOutputStream) clientOutputStreams.get(vt1);

writer2 = (ObjectOutputStream) clientOutputStreams.get(vt2);

room.get(j).user1P = "";

room.get(j).user2P = "";

writer1.writeObject("HDT:" + room.get(j).result1);

writer2.writeObject("HDT:" + room.get(j).result2);

}

}

}

b.Phía Client

\_Class IncomingReader: đây là lớp dùng để lắng nghe lien tục các dữ liệu từ phía server sau khi yêu cầu thực hiện kết nối với server này, lớp này được cài đặt như sau:

public class IncomingReader implements Runnable

{

@Override

public void run()

{

String[] data;

Object stream;

try

{

while ((stream = reader.readObject()) != null)

{

…….// Các hàm xử lý dữ liệu

}

}catch(Exception ex) { }

}

}

**IV.Kết luận**

Chúng em chọn giao thức TCP làm giao thức truyền thông của ứng dụng vì bộ giao thức internet này tương thích với tất cả các hệ điều hành, các loại phần cứng và mạng máy tính vì vậy ứng dụng này có thể chạy trên nhiều nền tảng khác nhau. Nó cũng có khả năng truyền dữ liệu không bị lỗi, khôi phục dữ liệu bị mất và hạn chế tắc nghẽn trên đường truyền, vì vậy rất phù hợp với các hoạt động truyền thông liên tục giữa server và client của ứng dụng.

Chúng em đã xây dựng ứng dụng trên hai phía client và server theo mô hình MVC. Việc lập trình ứng dụng yêu cầu nhiều kiến thức về lập trình như lập trình với CSDL, lập trình với giao thức TCP/IP, lập trình đa luồng,… Giao diện client rất dễ sử dụng với mọi người dùng, còn giao diện bên phía server được tùy chỉnh để dễ dàng quản lý thông tin người chơi. Game oẳn tù tì thi đấu online này đã được chúng em thử nghiệm và có thể được triển khai trên hệ thống mạng.

**V.Tài liệu tham khảo:**

\_Giáo trình Lập trình mạng – Nguyễn Mạnh Hùng

# \_ Multiple Client - Server Chat Programming Tutorial in  Java:

# http://makemobiapps.blogspot.com/p/multiple-client-server-chat-programming.html