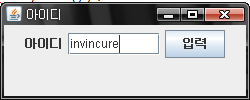
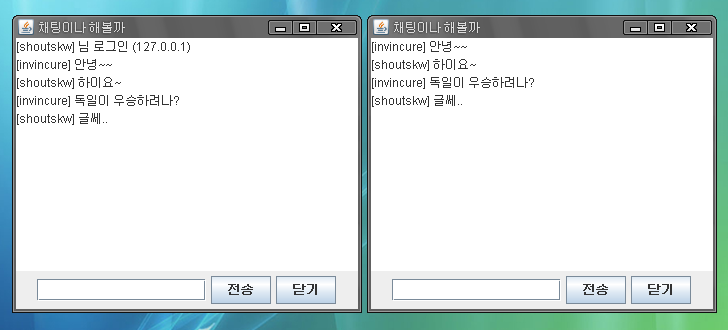
**GUI 채팅 프로그램**

**실행화면(당근 MultiChatServer.java먼저 실행 -> MultiChatClient.java 실행)**

****

****

**<ClientFrame.java>**

**import** java.awt.FlowLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.io.IOException;

**import** java.net.Socket;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**import** javax.swing.JTextArea;

**import** javax.swing.JTextField;

**class** Id **extends** JFrame **implements** ActionListener{

**static** JTextField *tf*=**new** JTextField(8);

JButton btn = **new** JButton("입력");

WriteThread wt;

ClientFrame cf;

**public** Id(){}

**public** Id(WriteThread wt, ClientFrame cf) {

**super**("아이디");

**this**.wt = wt;

**this**.cf = cf;

setLayout(**new** FlowLayout());

add(**new** JLabel("아이디"));

add(*tf*);

add(btn);

btn.addActionListener(**this**);

setBounds(300, 300, 250, 100);

setVisible(**true**);

}

**public** **void** actionPerformed(ActionEvent e) {

wt.sendMsg();

cf.isFirst = **false**;

cf.setVisible(**true**);

**this**.dispose();

}

**static** **public** String getId(){

**return** *tf*.getText();

}

}

**public** **class** ClientFrame **extends** JFrame **implements** ActionListener{

JTextArea txtA = **new** JTextArea();

JTextField txtF = **new** JTextField(15);

JButton btnTransfer = **new** JButton("전송");

JButton btnExit = **new** JButton("닫기");

**boolean** isFirst=**true**;

JPanel p1 = **new** JPanel();

Socket socket;

WriteThread wt;

**public** ClientFrame(Socket socket) {

**super**("채팅이나 해볼까");

**this**.socket = socket;

wt = **new** WriteThread(**this**);

**new** Id(wt, **this**);

add("Center", txtA);

p1.add(txtF);

p1.add(btnTransfer);

p1.add(btnExit);

add("South", p1);

//메세지를 전송하는 클래스 생성.

btnTransfer.addActionListener(**this**);

btnExit.addActionListener(**this**);

setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);

setBounds(300, 300, 350, 300);

setVisible(**false**);

}

**public** **void** actionPerformed(ActionEvent e){

String id = Id.*getId*();

**if**(e.getSource()==btnTransfer){//전송버튼 눌렀을 경우

//메세지 입력없이 전송버튼만 눌렀을 경우

**if**(txtF.getText().equals("")){

**return**;

}

txtA.append("["+id+"] "+ txtF.getText()+"\n");

wt.sendMsg();

txtF.setText("");

}**else**{

**this**.dispose();

}

}

}

**<MultiChatClient.java>**

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.PrintWriter;

**import** java.net.InetAddress;

**import** java.net.Socket;

// 키보드로 전송문자열 입력받아 서버로 전송하는 스레드

**class** WriteThread{

Socket socket;

ClientFrame cf;

String str;

String id;

**public** WriteThread(ClientFrame cf) {

**this**.cf = cf;

**this**.socket= cf.socket;

}

**public** **void** sendMsg() {

//키보드로부터 읽어오기 위한 스트림객체 생성

BufferedReader br=

**new** BufferedReader(**new** InputStreamReader(System.*in*));

PrintWriter pw=**null**;

**try**{

//서버로 문자열 전송하기 위한 스트림객체 생성

pw=**new** PrintWriter(socket.getOutputStream(),**true**);

//첫번째 데이터는 id 이다. 상대방에게 id와 함께 내 IP를 전송한다.

**if**(cf.isFirst==**true**){

InetAddress iaddr=socket.getLocalAddress();

String ip = iaddr.getHostAddress();

getId();

System.*out*.println("ip:"+ip+"id:"+id);

str = "["+id+"] 님 로그인 ("+ip+")";

}**else**{

str= "["+id+"] "+cf.txtF.getText();

}

//입력받은 문자열 서버로 보내기

pw.println(str);

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}**finally**{

**try**{

**if**(br!=**null**) br.close();

//if(pw!=null) pw.close();

//if(socket!=null) socket.close();

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}

}

}

**public** **void** getId(){

id = Id.*getId*();

}

}

//서버가 보내온 문자열을 전송받는 스레드

**class** ReadThread **extends** Thread{

Socket socket;

ClientFrame cf;

**public** ReadThread(Socket socket, ClientFrame cf) {

**this**.cf = cf;

**this**.socket=socket;

}

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

BufferedReader br=**null**;

**try**{

//서버로부터 전송된 문자열 읽어오기 위한 스트림객체 생성

br=**new** BufferedReader(

**new** InputStreamReader(socket.getInputStream()));

**while**(**true**){

//소켓으로부터 문자열 읽어옴

String str=br.readLine();

**if**(str==**null**){

System.*out*.println("접속이 끊겼음");

**break**;

}

//전송받은 문자열 화면에 출력

//System.out.println("[server] " + str);

cf.txtA.append(str+"\n");

}

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}**finally**{

**try**{

**if**(br!=**null**) br.close();

**if**(socket!=**null**) socket.close();

}**catch**(IOException ie){}

}

}

}

**public** **class** MultiChatClient {

**public** **static** **void** main(String[] args) {

Socket socket=**null**;

ClientFrame cf;

**try**{

socket=**new** Socket("127.0.0.1",3000);

System.*out*.println("연결성공!");

cf = **new** ClientFrame(socket);

**new** ReadThread(socket, cf).start();

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}

}

}

**<MultiChatServer.java>**

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.PrintWriter;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.util.Vector;

//클라이언트로 부터 전송된 문자열을 받아서 다른 클라이언트에게 문자열을

//보내주는 스레드

**class** EchoThread **extends** Thread{

Socket socket;

Vector<Socket> vec;

**public** EchoThread(Socket socket, Vector<Socket> vec){

**this**.socket = socket;

**this**.vec = vec;

}

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

BufferedReader br = **null**;

**try**{

br = **new** BufferedReader(

**new** InputStreamReader(socket.getInputStream()));

String str =**null**;

**while**(**true**){

//클라이언트로 부터 문자열 받기

str=br.readLine();

//상대가 접속을 끊으면 break;

**if**(str==**null**){

//벡터에서 없애기

vec.remove(socket);

**break**;

}

//연결된 소켓들을 통해서 다른 클라이언트에게 문자열 보내주기

sendMsg(str);

}

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}**finally**{

**try**{

**if**(br != **null**) br.close();

**if**(socket != **null**) socket.close();

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}

}

}

//전송받은 문자열 다른 클라이언트들에게 보내주는 메서드

**public** **void** sendMsg(String str){

**try**{

**for**(Socket socket:vec){

//for를 돌되 현재의 socket이 데이터를 보낸 클라이언트인 경우를 제외하고

//나머지 socket들에게만 데이터를 보낸다.

**if**(socket != **this**.socket){

PrintWriter pw =

**new** PrintWriter(socket.getOutputStream(), **true**);

pw.println(str);

pw.flush();

//단,여기서 얻어온 소켓들은 남의것들이기 때문에 여기서 닫으면 안된다.

}

}

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}

}

}

**public** **class** MultiChatServer {

**public** **static** **void** main(String[] args) {

ServerSocket server = **null**;

Socket socket =**null**;

//클라이언트와 연결된 소켓들을 배열처럼 저장할 벡터객체 생성

Vector<Socket> vec = **new** Vector<Socket>();

**try**{

server= **new** ServerSocket(3000);

**while**(**true**){

System.*out*.println("접속대기중..");

socket = server.accept();

//클라이언트와 연결된 소켓을 벡터에 담기

vec.add(socket);

//스레드 구동

**new** EchoThread(socket, vec).start();

}

}**catch**(IOException ie){

System.*out*.println(ie.getMessage());

}

}

}