

**제목: 멀티 클라이언트 채팅 프로그램**

|  |  |
| --- | --- |
| **과 목 :** | **JAVA응용프로그래밍** |
| **제출 일자 :** | **05/29** |
| **담당 교수 :** | **천인국교수님** |
| **학 과 :** | **컴퓨터공학과** |
| **학 번 :** | **20174090** |
| **이 름 :** | **이장우** |
|  |  |

1. 멀티 클라이언트 채팅 프로그램이란?

: 여러 개의 클라이언트 프로그램이 서버와 통신하면서 각 각의 역할을 수행하는 것.

2. Sever.java의 전체 구성

import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.HashMap;  
import java.util.StringTokenizer;  
import java.util.stream.IntStream;  
  
public class Server *{* private static HashMap*<*String, ClientHandler*> writers* = new HashMap*<*String, ClientHandler*>()*;  
 private static HashMap*<*String, ArrayList*<*String*>> rooms* = new HashMap*<*String, ArrayList*<*String*>>()*;  
 private Statement stmt;  
  
 public Server*() {* try *{* String url = "jdbc:mysql://localhost:3306/chatprogram\_db?autoReconnect=true&useSSL=false";  
 Class.*forName(*"com.mysql.cj.jdbc.Driver"*)*;  
 Connection con = DriverManager.*getConnection(*url,"root","root"*)*;  
 this.stmt = con.createStatement*()*;  
  
 ServerSocket ss = new ServerSocket*(*5056*)*;  
 *rooms*.put*(*"Main room", new ArrayList*<*String*>())*;  
 while *(*true*) {* Socket socket = ss.accept*()*;  
 System.*out*.println*(*"connection!" + socket*)*;  
 ClientHandler t = new ClientHandler*(*socket, *writers*, *rooms*,stmt*)*;  
 t.start*()*;  
 *}  
 }* catch *(*IOException | ClassNotFoundException | SQLException e*) {* e.printStackTrace*()*;  
 *}  
  
 }* public static void main*(*String*[]* args*) {* new Server*()*;  
 *}* class ClientHandler extends Thread *{* private Socket s;  
 private BufferedReader in;  
 private PrintWriter out;  
 private Statement stmt;  
 final HashMap*<*String, ClientHandler*>* writers;  
 final HashMap*<*String, ArrayList*<*String*>>* rooms;  
 String T\_str;  
 private StringTokenizer st;  
  
 public ClientHandler*(*Socket s, HashMap*<*String, ClientHandler*>* writers, HashMap*<*String, ArrayList*<*String*>>* rooms,Statement stmt*) {* this.s = s;  
 this.writers = writers;  
 this.rooms = rooms;  
 this.stmt = stmt;  
 *}* @Override  
 public void run*() {* try *{* this.in = new BufferedReader*(*new InputStreamReader*(*s.getInputStream*()))*;  
 this.out = new PrintWriter*(*s.getOutputStream*()*, true*)*;  
  
 while *(*true*) {* T\_str = in.readLine*()*;  
  
 System.*out*.println*(*T\_str*)*;  
 st = new StringTokenizer*(*T\_str, "/"*)*;  
 send\_message*(*recieved\_message*(*st*))*;  
 *}  
 }* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
 }* public void login\_success*(*String value*) {* synchronized *(*writers*) {* writers.put*(*value, this*)*;  
 *}* synchronized *(*rooms*) {* rooms.get*(*"Main room"*)*.add*(*value*)*;  
 *}  
 }* public void quitUser*(*String value*) {* synchronized *(*writers*) {* writers.remove*(*value, this*)*;  
 *}  
 }* public void room\_clear*(*String value*) {* synchronized *(*rooms*) {* IntStream.*range(*0, rooms.get*(*value*)*.size*())*.forEach*(*i -> rooms.get*(*"Main room"*)*.add*(*rooms.get*(*value*)*.get*(*i*)))*;  
 rooms.remove*(*value*)*;  
 *}  
 }* private void send\_message*(*int flag*) {* String str = "";  
 String user = "";  
 String userID = "";  
 String sql = "";  
  
 switch *(*flag*) {* case 0:  
 StringBuilder str1 = new StringBuilder*()*;  
 StringBuilder roomID = new StringBuilder*()*;  
 user = st.nextToken*()*;  
 login\_success*(*user*)*;  
 System.*out*.println*(*rooms*)*;  
 for *(*String i : rooms.keySet*()) {* roomID.append*(*i*)*.append*(*","*)*;  
 *}* for *(*String i : writers.keySet*()) {* str1.append*(*i*)*.append*(*","*)*;  
 *}* str = flag + "/" + roomID + "/" + str1;  
 broadcast*(*writers,str*)*;  
 System.*out*.println*(*str*)*;  
 str = "notice/" + user + "'s Online!";  
 broadcast*(*writers, str*)*;  
 break;  
 case 100:  
 str = T\_str;  
 user = st.nextToken*()*;  
 String outmsg = "";  
 quitUser*(*user*)*;  
 broadcast*(*writers, str*)*;  
 if*(*rooms.containsKey*(*user+"'s Room"*)){* room\_clear*(*user + "'s Room"*)*;  
 System.*out*.println*(*rooms*)*;  
 outmsg = "notice/" + user + "'s Room is gone!";  
 broadcast*(*writers, outmsg*)*;  
 *}* outmsg = "notice/" + user + "' Quit!";  
 broadcast*(*writers, outmsg*)*;  
 break;  
 case 1:  
 str = T\_str;  
 userID = st.nextToken*()*;  
 cast*(*writers,rooms,userID, str*)*;  
 break;  
 case 2:  
 String strbuf = st.nextToken*()*;  
 str += "1/Dm From " + st.nextToken*()* + "/" + st.nextToken*()*;  
 writers.get*(*strbuf*)*.out.println*(*str*)*;  
 break;  
 case 3:  
 String room\_name = st.nextToken*()*;  
 rooms.put*(*room\_name, new ArrayList*<*String*>())*;  
 str = "3/" + room\_name;  
 str1 = new StringBuilder*(*"notice/" + room\_name + "'s Open!"*)*;  
 broadcast*(*writers, str*)*;  
 broadcast*(*writers, str1.toString*())*;  
 break;  
 case 4:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 userID = st.nextToken*()*;  
 for *(*String i : rooms.keySet*()) {* for *(*int j = 0; j < rooms.get*(*i*)*.size*()*; j++*) {* if *(*rooms.get*(*i*)*.get*(*j*)*.equals*(*userID*)) {* System.*out*.println*(*"ok" + rooms*)*;  
 rooms.get*(*i*)*.remove*(*j*)*;  
 break;  
 *}* System.*out*.println*(*rooms*)*;  
 *}  
 }* rooms.get*(*str1.toString*())*.add*(*userID*)*;  
 System.*out*.println*(*rooms*)*;  
 break;  
 case 5:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 str = "select username from userinfo where username = \""+str1+"\";";  
 try *{* ResultSet rs = stmt.executeQuery*(*str*)*;  
 if*(*rs.next*())* out.println*(*"5/1"*)*;  
 else  
 out.println*(*"5/0"*)*;  
  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 6:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 str = "select username from userinfo where useraddress = \""+str1+"\";";  
 try *{* ResultSet rs = stmt.executeQuery*(*str*)*;  
 if*(*rs.next*())* out.println*(*"6/0"*)*;  
 else  
 out.println*(*"6/1"*)*;  
  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 7:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 String str2 = st.nextToken*()*;  
 String str3 = st.nextToken*()*;  
 sql = String.*format(*"insert into userinfo(username,password,useraddress) values(\"%s\",\"%s\",\"%s\");",  
 str1.toString*()*,str2,str3*)*;  
 try *{* stmt.executeUpdate*(*sql*)*;  
 out.println*(*"7"*)*;  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 8:  
 user = st.nextToken*()*;  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 sql = String.*format(*"select \* from userinfo where username = \"%s\" and password = \"%s\";",  
 user, str1.toString*())*;  
 try *{* ResultSet rs = stmt.executeQuery*(*sql*)*;  
 if*(*rs.next*())* out.println*(*"8/1/"+user*)*;  
 else  
 out.println*(*"8/2/"+user*)*;  
  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 9:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 sql = String.*format(*"select username,password from userinfo where useraddress =\"%s\";", str1.toString*())*;  
 try *{* ResultSet rs = stmt.executeQuery*(*sql*)*;  
 if*(*rs.next*()){* str1 = new StringBuilder*(*String.*format(*"9/1/%s/%s", rs.getString*(*1*)*, rs.getString*(*2*)))*;  
 out.println*(*str1*)*;  
 *}* else*{* out.println*(*"9/0"*)*;  
 *}  
 }* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}  
 }  
 }  
 }* private int recieved\_message*(*StringTokenizer st*) {* return Integer.*parseInt(*st.nextToken*())*;  
 *}* private void broadcast*(*HashMap*<*String, ClientHandler*>* writers, String str*) {* for *(*String i : writers.keySet*()) {* writers.get*(*i*)*.out.println*(*str*)*;  
 try *{* stmt.executeUpdate*(*String.*format(*"insert into chat\_log(log) values(\"%s\");",str*))*;  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}  
 }  
 }* private void cast*(*HashMap*<*String, ClientHandler*>* writers,HashMap*<*String, ArrayList*<*String*>>* rooms, String str,String str1*) {* Collection*<*ArrayList*<*String*>>* values = rooms.values*()*;  
 for*(*ArrayList*<*String*>* i : values*){* for*(*int j = 0; j< i.size*()*; j++*){* if*(*str.equals*(*i.get*(*j*))){* for*(*j = 0; j < i.size*()*;j++*){* for *(*String k : writers.keySet*()) {* if *(*i.get*(*j*)*.equals*(*k*)){* writers.get*(*k*)*.out.println*(*str1*)*;  
 try *{* stmt.executeUpdate*(*String.*format(*"insert into chat\_log(log) values(\"%s\");",str*))*;  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
}*

2. Sever.java의 세부 구성.

#1. public Sever()

public Server*() {* try *{* String url = "jdbc:mysql://localhost:3306/chatprogram\_db?autoReconnect=true&useSSL=false";  
 Class.*forName(*"com.mysql.cj.jdbc.Driver"*)*;  
 Connection con = DriverManager.*getConnection(*url,"root","root"*)*;  
 this.stmt = con.createStatement*()*;  
  
 ServerSocket ss = new ServerSocket*(*5056*)*;  
 *rooms*.put*(*"Main room", new ArrayList*<*String*>())*;  
 while *(*true*) {* Socket socket = ss.accept*()*;  
 System.*out*.println*(*"connection!" + socket*)*;  
 ClientHandler t = new ClientHandler*(*socket, *writers*, *rooms*,stmt*)*;  
 t.start*()*;  
 *}  
 }* catch *(*IOException | ClassNotFoundException | SQLException e*) {* e.printStackTrace*()*;  
 *}  
}*

1. 변수 stmt에 db를 연결하기 위한 사전작업을 준비합니다.

2. SeverSocket을 생성하고 while문이 무한 루프를 돌면서 서버가 항상 실행되고 연결이 되었을 경우 accept()를 지정하여 미리 생성해 둔 hashmap에 해당 객체를 삽입하고 해당 객체의 쓰레드를 실행 시킵니다.

#2. public static void main(String[] args)

public static void main*(*String*[]* args*) {* new Server*()*;  
*}*

1. 메인 클래스를 통해 서버를 실행 시킵니다.

#3. public void run()

public void run*() {* try *{* this.in = new BufferedReader*(*new InputStreamReader*(*s.getInputStream*()))*;  
 this.out = new PrintWriter*(*s.getOutputStream*()*, true*)*;  
  
 while *(*true*) {* T\_str = in.readLine*()*;  
  
 System.*out*.println*(*T\_str*)*;  
 st = new StringTokenizer*(*T\_str, "/"*)*;  
 send\_message*(*recieved\_message*(*st*))*;  
 *}  
 }* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
}*

1. run()은 해당 클래스에 Thread를 상속받아 재정의 되는 함수로 Thread.run()을 통해 실행 시킬 수 있습니다. 이 클래스의 run메소드의 역할은 서버는 항상 클라이언트로부터 메시지를 받아야 하기 때문에 무한루프를 통해 해당 클라이언트의 메시지를 받을 준비를 하고 있습니다.

#4. public void login\_success(String value)

public void login\_success*(*String value*) {* synchronized *(*writers*) {* writers.put*(*value, this*)*;  
 *}* synchronized *(*rooms*) {* rooms.get*(*"Main room"*)*.add*(*value*)*;  
 *}  
}*

1. 해당 메소드는 로그인이 성공 될 시, writers 라는 Hashmap에 데이터를 삽입합니다. 그리고 room이라는 다른 Hashmap에 대해 키값이 “Main room”인 객체를 가져와 value를 삽입합니다.

#5. public quitUser(String value)

public void quitUser*(*String value*) {* synchronized *(*writers*) {* writers.remove*(*value, this*)*;  
 *}  
}*

1. 해당 메소드는 유저가 프로그램을 종료할 시, 해당 유저의 객체를 writers에서 제거하는 함수 입니다.

#6. public room\_clear(String value)

public void room\_clear*(*String value*) {* synchronized *(*rooms*) {* IntStream.*range(*0, rooms.get*(*value*)*.size*())*.forEach*(*i -> rooms.get*(*"Main room"*)*.add*(*rooms.get*(*value*)*.get*(*i*)))*;  
 rooms.remove*(*value*)*;  
 *}  
}*

1. 해당 메소드는 유저가 프로그램을 종료할 시, 해당 유저가 만든 채팅방을 해제하고 그 채팅방에 있던 유저를 “Main room”에 옮기는 작업을 하는 메소드 입니다.

#7. private void send\_message(int flag)

private void send\_message*(*int flag*) {* String str = "";  
 String user = "";  
 String userID = "";  
 String sql = "";  
  
 switch *(*flag*) {* case 0:  
 StringBuilder str1 = new StringBuilder*()*;  
 StringBuilder roomID = new StringBuilder*()*;  
 user = st.nextToken*()*;  
 login\_success*(*user*)*;  
 System.*out*.println*(*rooms*)*;  
 for *(*String i : rooms.keySet*()) {* roomID.append*(*i*)*.append*(*","*)*;  
 *}* for *(*String i : writers.keySet*()) {* str1.append*(*i*)*.append*(*","*)*;  
 *}* str = flag + "/" + roomID + "/" + str1;  
 broadcast*(*writers,str*)*;  
 System.*out*.println*(*str*)*;  
 str = "notice/" + user + "'s Online!";  
 broadcast*(*writers, str*)*;  
 break;  
 case 100:  
 str = T\_str;  
 user = st.nextToken*()*;  
 String outmsg = "";  
 quitUser*(*user*)*;  
 broadcast*(*writers, str*)*;  
 if*(*rooms.containsKey*(*user+"'s Room"*)){* room\_clear*(*user + "'s Room"*)*;  
 System.*out*.println*(*rooms*)*;  
 outmsg = "notice/" + user + "'s Room is gone!";  
 broadcast*(*writers, outmsg*)*;  
 *}* outmsg = "notice/" + user + "' Quit!";  
 broadcast*(*writers, outmsg*)*;  
 break;  
 case 1:  
 str = T\_str;  
 userID = st.nextToken*()*;  
 cast*(*writers,rooms,userID, str*)*;  
 break;  
 case 2:  
 String strbuf = st.nextToken*()*;  
 str += "1/Dm From " + st.nextToken*()* + "/" + st.nextToken*()*;  
 writers.get*(*strbuf*)*.out.println*(*str*)*;  
 break;  
 case 3:  
 String room\_name = st.nextToken*()*;  
 rooms.put*(*room\_name, new ArrayList*<*String*>())*;  
 str = "3/" + room\_name;  
 str1 = new StringBuilder*(*"notice/" + room\_name + "'s Open!"*)*;  
 broadcast*(*writers, str*)*;  
 broadcast*(*writers, str1.toString*())*;  
 break;  
 case 4:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 userID = st.nextToken*()*;  
 for *(*String i : rooms.keySet*()) {* for *(*int j = 0; j < rooms.get*(*i*)*.size*()*; j++*) {* if *(*rooms.get*(*i*)*.get*(*j*)*.equals*(*userID*)) {* System.*out*.println*(*"ok" + rooms*)*;  
 rooms.get*(*i*)*.remove*(*j*)*;  
 break;  
 *}* System.*out*.println*(*rooms*)*;  
 *}  
 }* rooms.get*(*str1.toString*())*.add*(*userID*)*;  
 System.*out*.println*(*rooms*)*;  
 break;  
 case 5:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 str = "select username from userinfo where username = \""+str1+"\";";  
 try *{* ResultSet rs = stmt.executeQuery*(*str*)*;  
 if*(*rs.next*())* out.println*(*"5/1"*)*;  
 else  
 out.println*(*"5/0"*)*;  
  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 6:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 str = "select username from userinfo where useraddress = \""+str1+"\";";  
 try *{* ResultSet rs = stmt.executeQuery*(*str*)*;  
 if*(*rs.next*())* out.println*(*"6/0"*)*;  
 else  
 out.println*(*"6/1"*)*;  
  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 7:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 String str2 = st.nextToken*()*;  
 String str3 = st.nextToken*()*;  
 sql = String.*format(*"insert into userinfo(username,password,useraddress) values(\"%s\",\"%s\",\"%s\");",  
 str1.toString*()*,str2,str3*)*;  
 try *{* stmt.executeUpdate*(*sql*)*;  
 out.println*(*"7"*)*;  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 8:  
 user = st.nextToken*()*;  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 sql = String.*format(*"select \* from userinfo where username = \"%s\" and password = \"%s\";",  
 user, str1.toString*())*;  
 try *{* ResultSet rs = stmt.executeQuery*(*sql*)*;  
 if*(*rs.next*())* out.println*(*"8/1/"+user*)*;  
 else  
 out.println*(*"8/2/"+user*)*;  
  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}* break;  
 case 9:  
 str1 = new StringBuilder*(*st.nextToken*())*;  
 sql = String.*format(*"select username,password from userinfo where useraddress =\"%s\";", str1.toString*())*;  
 try *{* ResultSet rs = stmt.executeQuery*(*sql*)*;  
 if*(*rs.next*()){* str1 = new StringBuilder*(*String.*format(*"9/1/%s/%s", rs.getString*(*1*)*, rs.getString*(*2*)))*;  
 out.println*(*str1*)*;  
 *}* else*{* out.println*(*"9/0"*)*;  
 *}  
 }* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}  
 }  
 }  
}*

1. 일단 이 메소드를 설명 하자면 강의 내용에서 나왔던 것과 같이 서버와 클라이언트는 서로 메시지를 주고 받을 때 특정한 형식을 지닌다고 배웠고 그것을 Protocol이라고 들었습니다. 그래서 저는 이 채팅 프로그램만의 Protocol을 구성 하였습니다.

해당 Protocol의 구분은 StringTokenizer 컬렉션을 이용하여 작업했습니다.

|  |  |  |
| --- | --- | --- |
| Protocol | Content | Use |
| 0 | “0/userID” | 누군가 처음 로그인에 성공했을 때 |
| 100 | “100/userID” | 누군가 접속을 종료 했을 때 |
| 1 | “1/userID/Send\_Text” | 누군가 메시지(Send\_Text)를 입력하여 보낼때 |
| 2 | “2/To\_userID/From\_userID/Send\_Text” | 어떠한 사용자(From\_userID)가 어떠한 사용자(To\_userID) 가 메시지(Send\_Text)를 보낼 때 |
| 3 | “3/userID” | 어떠한 사용자(userID)가 채팅방을 개설할 때 |
| 4 | “4/Room/userID” | 어떠한 사용자(userID)가 어떤 채팅방(Room)에 접속 할때 |
| 5 | “5/userID” | ID(userID) 중복체크 |
| 6 | “6/userEMAIL” | Email(userEAMIL) 중복체크 |
| 7 | “7/userID/userPW/userEMAIL” | 회원 가입 시 (ID(userID),Password(userPW),E-mail(userEMAIL)) |
| 8 | “8/userID/userPW/userEMAIL” | 로그인 시 (ID(userID),Password(userPW),E-mail(userEMAIL) |
| 9 | “9/userEMAIL” | E-mail(userEMAIL)로 ID를 찾을 때 |

#8. private int received\_message(StringTokenizer st)

private int recieved\_message*(*StringTokenizer st*) {* return Integer.*parseInt(*st.nextToken*())*;  
*}*

1. 토큰을 받아 정수형으로 변환 시키는 메소드입니다.

#9 private void broadcast(HashMap<String, ClientHandler>, String str)

private void broadcast*(*HashMap*<*String, ClientHandler*>* writers, String str*) {* for *(*String i : writers.keySet*()) {* writers.get*(*i*)*.out.println*(*str*)*;  
 try *{* stmt.executeUpdate*(*String.*format(*"insert into chat\_log(log) values(\"%s\");",str1*))*;  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}  
 }  
}*

1. 해당 메소드는 서버측에서 클라이언트의 메시지를 받은 후 가공한 str(Protocol)를 writers에 저장된 모든 클라이언트들에게 전송하는 역할을 합니다. 그 후에 DB에 로그를 저장합니다.

#10 private void cast(HashMap<String, ClientHandler>,HashMap<string, ArrayList<String> room, String str, String str1)

private void cast*(*HashMap*<*String, ClientHandler*>* writers,HashMap*<*String, ArrayList*<*String*>>* rooms, String str,String str1*) {* Collection*<*ArrayList*<*String*>>* values = rooms.values*()*;  
 for*(*ArrayList*<*String*>* i : values*){* for*(*int j = 0; j< i.size*()*; j++*){* if*(*str.equals*(*i.get*(*j*))){* for*(*j = 0; j < i.size*()*;j++*){* for *(*String k : writers.keySet*()) {* if *(*i.get*(*j*)*.equals*(*k*)){* writers.get*(*k*)*.out.println*(*str1*)*;  
 try *{* stmt.executeUpdate*(*String.*format(*"insert into chat\_log(log) values(\"%s\");",str*))*;  
 *}* catch *(*SQLException throwables*) {* throwables.printStackTrace*()*;  
 *}  
 }  
 }  
 }  
 }  
 }  
 }  
}*

1. 해당 메소드는 broadcast메소드와 비슷한 역할을 하지만 차이점이 이 메소드는 특정한 채팅방에 있는 유저에게만 메시지를 보내는 메소드입니다. 그 후에 broadcast와 동일하게 DB에 로그를 저장합니다.

3. Client.java의 전체 구성

import javax.swing.\*;  
import javax.swing.border.EmptyBorder;  
import java.awt.event.\*;  
import java.io.\*;  
import java.net.Socket;  
import java.util.\*;  
  
public class Client extends JFrame implements ActionListener,Runnable *{* //채팅창에 필요한 변수들  
 private final JFrame MainFrame = new JFrame*()*;  
 JTextArea Chat = new JTextArea*()*;  
 private final JTextField Send\_mg = new JTextField*()*;  
 JLabel Login\_people = new JLabel*(*"Online"*)*;  
 JList*<*String*>* People\_list = new JList*<>()*;  
 JButton Send\_personal = new JButton*(*"Send DM"*)*;  
 JLabel Room = new JLabel*(*"Chat Room List"*)*;  
 JList*<*String*>* Room\_list = new JList*<>()*;  
 JButton Join\_room = new JButton*(*"Join"*)*;  
 JButton Make\_room = new JButton*(*"Make"*)*;  
 JButton Send\_text = new JButton*(*"Send"*)*;  
  
 //로그인시 필요한 변수들  
 private final JFrame LoginFrame = new JFrame*()*;  
 private final JTextField Login\_textField = new JTextField*()*;  
 private final JPasswordField Login\_passwordField = new JPasswordField*()*;  
  
 JLabel ID\_label = new JLabel*(*"ID"*)*;  
 JLabel PW\_label = new JLabel*(*"PW"*)*;  
  
 JButton Join\_button = new JButton*(*"Join"*)*;  
 JButton Search\_button = new JButton*(*"Search"*)*;  
 JButton Login\_button = new JButton*(*"Login"*)*;  
  
  
 //회원가입시 필요한 변수들  
 private final JFrame JoinFrame = new JFrame*()*;  
 private final JTextField ID\_textField = new JTextField*()*;  
 private final JPasswordField Join\_passwordField = new JPasswordField*()*;  
 private final JTextField Email\_textField = new JTextField*()*;  
 JButton ID\_check\_button = new JButton*(*"Check"*)*;  
 JButton Email\_check\_button = new JButton*(*"Check"*)*;  
 JButton Join\_complete\_button = new JButton*(*"Join"*)*;  
 JButton Join\_canncel\_button = new JButton*(*"Canncel"*)*;  
 int id\_flag = 0;  
 int email\_flag = 0;  
  
 //네트워크 통신을 위한 변수들  
 private BufferedReader in;  
 private PrintWriter out;  
 private StringTokenizer st;  
 Vector*<*String*>* user\_list = new Vector*<*String*>()*;  
 Vector*<*String*>* room\_list = new Vector*<*String*>()*;  
  
 public void Login\_frame*() {* LoginFrame.setTitle*(*"Login"*)*;  
 LoginFrame.setDefaultCloseOperation*(*JFrame.*DISPOSE\_ON\_CLOSE)*;  
 LoginFrame.setBounds*(*100, 100, 377, 220*)*;  
 JPanel login\_contentPane = new JPanel*()*;  
 login\_contentPane.setBorder*(*new EmptyBorder*(*5, 5, 5, 5*))*;  
 LoginFrame.setContentPane*(*login\_contentPane*)*;  
 login\_contentPane.setLayout*(*null*)*;  
  
 Login\_textField.setBounds*(*98, 28, 251, 33*)*;  
 login\_contentPane.add*(*Login\_textField*)*;  
 Login\_textField.setColumns*(*10*)*;  
  
 ID\_label.setBounds*(*26, 37, 36, 15*)*;  
 login\_contentPane.add*(*ID\_label*)*;  
  
 PW\_label.setBounds*(*26, 95, 57, 15*)*;  
 login\_contentPane.add*(*PW\_label*)*;  
  
 Join\_button.setBounds*(*26, 148, 97, 23*)*;  
 login\_contentPane.add*(*Join\_button*)*;  
 Join\_button.addActionListener*(*this*)*;  
  
 Search\_button.setBounds*(*135, 148, 97, 23*)*;  
 login\_contentPane.add*(*Search\_button*)*;  
 Search\_button.addActionListener*(*this*)*;  
  
 Login\_button.setBounds*(*244, 148, 97, 23*)*;  
 login\_contentPane.add*(*Login\_button*)*;  
 Login\_button.addActionListener*(*this*)*;  
  
 Login\_passwordField.setBounds*(*98, 86, 251, 33*)*;  
 login\_contentPane.add*(*Login\_passwordField*)*;  
  
 LoginFrame.setVisible*(*true*)*;  
 Thread t = new Thread*(*this*)*;  
 t.start*()*;  
 *}* public void Main\_frame*() {* MainFrame.addWindowListener*(*new WindowAdapter*() {* @Override  
 public void windowClosing*(*WindowEvent e*) {* out.println*(*"100/"+Login\_textField.getText*())*;  
 MainFrame.setVisible*(*false*)*;  
 MainFrame.dispose*()*;  
 *}  
 })*;  
 MainFrame.setBounds*(*100, 100, 721, 606*)*;  
 JPanel main\_contentPane = new JPanel*()*;  
 main\_contentPane.setBorder*(*new EmptyBorder*(*5, 5, 5, 5*))*;  
 MainFrame.setContentPane*(*main\_contentPane*)*;  
 main\_contentPane.setLayout*(*null*)*;  
  
 JScrollPane Chat\_scrollPane = new JScrollPane*()*;  
 Chat\_scrollPane.setBounds*(*167, 10, 480, 460*)*;  
 main\_contentPane.add*(*Chat\_scrollPane*)*;  
  
 Chat\_scrollPane.setViewportView*(*Chat*)*;  
 Chat.setEditable*(*false*)*;  
 Send\_mg.setBounds*(*167, 493, 390, 31*)*;  
 main\_contentPane.add*(*Send\_mg*)*;  
 Send\_mg.setColumns*(*10*)*;  
  
 Login\_people.setBounds*(*36, 15, 81, 15*)*;  
 main\_contentPane.add*(*Login\_people*)*;  
  
 JScrollPane People\_scrollPane = new JScrollPane*()*;  
 People\_scrollPane.setBounds*(*22, 40, 112, 157*)*;  
 main\_contentPane.add*(*People\_scrollPane*)*;  
  
 People\_scrollPane.setViewportView*(*People\_list*)*;  
 People\_list.setSelectionMode*(*ListSelectionModel.*SINGLE\_SELECTION)*;  
  
 Send\_personal.setBounds*(*22, 207, 112, 23*)*;  
 main\_contentPane.add*(*Send\_personal*)*;  
 Send\_personal.addActionListener*(*this*)*;  
  
 Room.setBounds*(*36, 240, 81, 15*)*;  
 main\_contentPane.add*(*Room*)*;  
  
 JScrollPane Room\_scrollPane = new JScrollPane*()*;  
 Room\_scrollPane.setBounds*(*22, 270, 112, 184*)*;  
 main\_contentPane.add*(*Room\_scrollPane*)*;  
  
 Room\_scrollPane.setViewportView*(*Room\_list*)*;  
  
 Join\_room.setBounds*(*20, 464, 114, 23*)*;  
 main\_contentPane.add*(*Join\_room*)*;  
 Join\_room.addActionListener*(*this*)*;  
  
 Make\_room.setBounds*(*22, 497, 112, 23*)*;  
 main\_contentPane.add*(*Make\_room*)*;  
 Make\_room.addActionListener*(*this*)*;  
  
 Send\_text.setBounds*(*563, 493, 81, 31*)*;  
 main\_contentPane.add*(*Send\_text*)*;  
 Send\_text.addActionListener*(*this*)*;  
  
 MainFrame.setVisible*(*true*)*;  
 Thread t = new Thread*(*this*)*;  
 t.start*()*;  
 *}* public void Join\_frame*(){* JoinFrame.setTitle*(*"Join!"*)*;  
 JoinFrame.setDefaultCloseOperation*(*JFrame.*DISPOSE\_ON\_CLOSE)*;  
 JoinFrame.setBounds*(*100, 100, 439, 271*)*;  
 JPanel join\_contentPane = new JPanel*()*;  
 join\_contentPane.setBorder*(*new EmptyBorder*(*5, 5, 5, 5*))*;  
 JoinFrame.setContentPane*(*join\_contentPane*)*;  
 join\_contentPane.setLayout*(*null*)*;  
  
 JLabel lblNewLabel = new JLabel*(*"Username"*)*;  
 lblNewLabel.setBounds*(*12, 40, 58, 15*)*;  
 join\_contentPane.add*(*lblNewLabel*)*;  
  
 JLabel lblNewLabel\_1 = new JLabel*(*"Password"*)*;  
 lblNewLabel\_1.setBounds*(*12, 88, 57, 15*)*;  
 join\_contentPane.add*(*lblNewLabel\_1*)*;  
  
 JLabel lblNewLabel\_2 = new JLabel*(*"e-mail"*)*;  
 lblNewLabel\_2.setBounds*(*12, 142, 44, 15*)*;  
 join\_contentPane.add*(*lblNewLabel\_2*)*;  
  
 ID\_textField.setBounds*(*86, 37, 220, 21*)*;  
 join\_contentPane.add*(*ID\_textField*)*;  
 ID\_textField.setColumns*(*10*)*;  
  
 Join\_passwordField.setBounds*(*86, 85, 220, 21*)*;  
 join\_contentPane.add*(*Join\_passwordField*)*;  
  
 Email\_textField.setBounds*(*86, 139, 220, 21*)*;  
 join\_contentPane.add*(*Email\_textField*)*;  
 Email\_textField.setColumns*(*10*)*;  
  
 ID\_check\_button.setBounds*(*318, 36, 97, 23*)*;  
 join\_contentPane.add*(*ID\_check\_button*)*;  
 ID\_check\_button.addActionListener*(*this*)*;  
  
 Email\_check\_button.setBounds*(*318, 138, 97, 23*)*;  
 join\_contentPane.add*(*Email\_check\_button*)*;  
 Email\_check\_button.addActionListener*(*this*)*;  
  
 Join\_complete\_button.setBounds*(*86, 196, 97, 23*)*;  
 join\_contentPane.add*(*Join\_complete\_button*)*;  
 Join\_complete\_button.addActionListener*(*this*)*;  
  
 Join\_canncel\_button.setBounds*(*209, 196, 97, 23*)*;  
 join\_contentPane.add*(*Join\_canncel\_button*)*;  
 Join\_canncel\_button.addActionListener*(*this*)*;  
  
 JoinFrame.setVisible*(*true*)*;  
 *}* @Override  
 public void actionPerformed*(*ActionEvent e*) {* if*(*e.getSource*()* == Login\_button*){* out.println*(*"8/"+Login\_textField.getText*()*+"/"+String.*valueOf(*Login\_passwordField.getPassword*()))*;  
 *}* if*(*e.getSource*()* == Send\_text*){* out.println*(*"1/"+Login\_textField.getText*()*+"/"+Send\_mg.getText*())*;  
 *}* if*(*e.getSource*()* == Send\_personal*){* Chat.append*(*"Dm To "+ People\_list.getSelectedValue*()* + ":" + Send\_mg.getText*()*.trim*()*+"\n"*)*;  
 out.println*(*"2/"+People\_list.getSelectedValue*()*+"/"+Login\_textField.getText*()*+"/"+  
 Send\_mg.getText*())*;  
 *}* if*(*e.getSource*()* == Make\_room*){* out.println*(*"3/"+Login\_textField.getText*()*+"'s Room"*)*;  
 *}* if*(*e.getSource*()* == Join\_room*){* out.println*(*"4/"+Room\_list.getSelectedValue*()*+"/"+Login\_textField.getText*())*;  
 *}* if*(*e.getSource*()* == Join\_button *){* Join\_frame*()*;  
 *}* if*(*e.getSource*()* == ID\_check\_button*){* out.println*(*"5/"+ID\_textField.getText*())*;  
 *}* if*(*e.getSource*()* == Email\_check\_button*){* out.println*(*"6/"+Email\_textField.getText*())*;  
 *}* if*(*e.getSource*()* == Join\_complete\_button*){* out.println*(*"5/"+ID\_textField.getText*())*;  
 out.println*(*"6/"+Email\_textField.getText*())*;  
 if*(*id\_flag == 1 && email\_flag == 1*){* out.println*(*"7/"+ID\_textField.getText*()*+"/"+String.*valueOf(*Join\_passwordField.getPassword*())*+"/"+Email\_textField.getText*())*;  
 *}* else*{* if*(*id\_flag == 0*){* JOptionPane.*showMessageDialog(*null,"Check the ID!","Can't Confirm!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}* if*(*email\_flag == 0*){* JOptionPane.*showMessageDialog(*null,"Check the E-Mail!","Can't Confirm!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }  
 }* if*(*e.getSource*()* == Search\_button*){* JOptionPane Search = new JOptionPane*()*;  
 String Email = JOptionPane.*showInputDialog(*"Input your E-mail!"*)*;  
  
 out.println*(*"9/"+Email*)*;  
 *}* if*(*e.getSource*()* == Join\_canncel\_button*){* JoinFrame.dispose*()*;  
 *}  
 }* public Client*() {* try *{* Socket s = new Socket*(*"localhost", 5056*)*;  
 in = new BufferedReader*(*new InputStreamReader*(*s.getInputStream*()))*;  
 out = new PrintWriter*(*s.getOutputStream*()*, true*)*;  
 *}* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
 }* public static void main*(*String*[]* args*) {* Client c = new Client*()*;  
 c.Login\_frame*()*;  
 *}* @Override  
 public void run*() {* while *(*true*){* try *{* String msg = in.readLine*()*;  
  
 System.*out*.println*(*msg*)*;  
 inmessage*(*msg*)*;  
 *}* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
 }  
 }* public void inmessage*(*String msg*){* st = new StringTokenizer*(*msg,"/"*)*;  
 String flag = st.nextToken*()*;  
  
 switch *(*flag*) {* case "0" -> *{* String stb\_r = st.nextToken*()*;  
 String stb = st.nextToken*()*;  
 StringTokenizer sttb\_r = new StringTokenizer*(*stb\_r, ","*)*;  
 while *(*sttb\_r.hasMoreTokens*()) {* room\_list.add*(*sttb\_r.nextToken*())*;  
 TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*room\_list*)*;  
 Vector*<*String*>* room\_list = new Vector*<*String*>(*buf\_list*)*;  
 Room\_list.setListData*(*room\_list*)*;  
 *}* StringTokenizer sttb = new StringTokenizer*(*stb, ","*)*;  
 while *(*sttb.hasMoreTokens*()) {* user\_list.add*(*sttb.nextToken*())*;  
 TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*user\_list*)*;  
 Vector*<*String*>* user\_list = new Vector*<*String*>(*buf\_list*)*;  
 People\_list.setListData*(*user\_list*)*;  
 *}  
  
 }* case "1" -> Chat.append*(*st.nextToken*()* + ":" + st.nextToken*()*.trim*()* + "\n"*)*;  
 case "100" -> *{* String buffer = st.nextToken*()*;  
 String buffer\_r = buffer + "'s Room";  
 for *(*int i = 0; i < room\_list.size*()*; i++*) {* if *(*room\_list.get*(*i*)*.equals*(*buffer\_r*)) {* room\_list.remove*(*i*)*;  
 *}  
 }* for *(*int i = 0; i < user\_list.size*()*; i++*) {* if *(*user\_list.get*(*i*)*.equals*(*buffer*)) {* user\_list.remove*(*i*)*;  
 *}  
 }* TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*user\_list*)*;  
 Vector*<*String*>* user\_list = new Vector*<*String*>(*buf\_list*)*;  
 TreeSet*<*String*>* buf\_list\_r = new TreeSet*<*String*>(*room\_list*)*;  
 Vector*<*String*>* room\_list = new Vector*<*String*>(*buf\_list\_r*)*;  
 People\_list.setListData*(*user\_list*)*;  
 Room\_list.setListData*(*room\_list*)*;  
 *}* case "3" -> *{* String stb = st.nextToken*()*;  
 room\_list.add*(*stb*)*;  
 TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*room\_list*)*;  
 Vector*<*String*>* room\_list = new Vector*<*String*>(*buf\_list*)*;  
 Room\_list.setListData*(*room\_list*)*;  
 *}* case "5" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* JOptionPane.*showMessageDialog(*null, "You can't use this ID!", "Can't Use!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}* else *{* id\_flag = 1;  
 JOptionPane.*showMessageDialog(*null, "You can use this ID!", "Can Use!",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 *}  
 }* case "6" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* email\_flag = 1;  
 JOptionPane.*showMessageDialog(*null, "You can use this E-Mail!", "Can Use!",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 *}* else *{* JOptionPane.*showMessageDialog(*null, "You can't use this E-Mail!", "Can't Use!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }* case "7" -> *{* JOptionPane.*showMessageDialog(*null, "Welcome!", "Welcome!",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 JoinFrame.dispose*()*;  
 *}* case "8" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* stb = st.nextToken*()*;  
 LoginFrame.dispose*()*;  
 MainFrame.setTitle*(*stb + "'s Chat!"*)*;  
 out.println*(*"0/" + stb*)*;  
 Main\_frame*()*;  
 *}* else *{* JOptionPane.*showMessageDialog(*null, "You can't Login!!", "Can't Login!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }* case "9" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* JOptionPane.*showMessageDialog(*null, "ID : " + st.nextToken*()* + "\nPW : " + st.nextToken*()*, "Result",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 *}* else *{* JOptionPane.*showMessageDialog(*null, "We can't Find!!", "We can't find!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }* case "notice" -> Chat.append*(*st.nextToken*()* + "\n"*)*;  
 *}  
 }  
}*

*}*

#1. public void Login\_frame() : 로그인 화면 구성

public void Login\_frame*() {* LoginFrame.setTitle*(*"Login"*)*;  
 LoginFrame.setDefaultCloseOperation*(*JFrame.*DISPOSE\_ON\_CLOSE)*;  
 LoginFrame.setBounds*(*100, 100, 377, 220*)*;  
 JPanel login\_contentPane = new JPanel*()*;  
 login\_contentPane.setBorder*(*new EmptyBorder*(*5, 5, 5, 5*))*;  
 LoginFrame.setContentPane*(*login\_contentPane*)*;  
 login\_contentPane.setLayout*(*null*)*;  
  
 Login\_textField.setBounds*(*98, 28, 251, 33*)*;  
 login\_contentPane.add*(*Login\_textField*)*;  
 Login\_textField.setColumns*(*10*)*;  
  
 ID\_label.setBounds*(*26, 37, 36, 15*)*;  
 login\_contentPane.add*(*ID\_label*)*;  
  
 PW\_label.setBounds*(*26, 95, 57, 15*)*;  
 login\_contentPane.add*(*PW\_label*)*;  
  
 Join\_button.setBounds*(*26, 148, 97, 23*)*;  
 login\_contentPane.add*(*Join\_button*)*;  
 Join\_button.addActionListener*(*this*)*;  
  
 Search\_button.setBounds*(*135, 148, 97, 23*)*;  
 login\_contentPane.add*(*Search\_button*)*;  
 Search\_button.addActionListener*(*this*)*;  
  
 Login\_button.setBounds*(*244, 148, 97, 23*)*;  
 login\_contentPane.add*(*Login\_button*)*;  
 Login\_button.addActionListener*(*this*)*;  
  
 Login\_passwordField.setBounds*(*98, 86, 251, 33*)*;  
 login\_contentPane.add*(*Login\_passwordField*)*;  
  
 LoginFrame.setVisible*(*true*)*;  
 Thread t = new Thread*(*this*)*;  
 t.start*()*;  
*}*

#2. public void Main\_frame() : 메인 채팅 화면 구성

public void Main\_frame*() {* MainFrame.addWindowListener*(*new WindowAdapter*() {* @Override  
 public void windowClosing*(*WindowEvent e*) {* out.println*(*"100/"+Login\_textField.getText*())*;  
 MainFrame.setVisible*(*false*)*;  
 MainFrame.dispose*()*;  
 *}  
 })*;  
 MainFrame.setBounds*(*100, 100, 721, 606*)*;  
 JPanel main\_contentPane = new JPanel*()*;  
 main\_contentPane.setBorder*(*new EmptyBorder*(*5, 5, 5, 5*))*;  
 MainFrame.setContentPane*(*main\_contentPane*)*;  
 main\_contentPane.setLayout*(*null*)*;  
  
 JScrollPane Chat\_scrollPane = new JScrollPane*()*;  
 Chat\_scrollPane.setBounds*(*167, 10, 480, 460*)*;  
 main\_contentPane.add*(*Chat\_scrollPane*)*;  
  
 Chat\_scrollPane.setViewportView*(*Chat*)*;  
 Chat.setEditable*(*false*)*;  
 Send\_mg.setBounds*(*167, 493, 390, 31*)*;  
 main\_contentPane.add*(*Send\_mg*)*;  
 Send\_mg.setColumns*(*10*)*;  
  
 Login\_people.setBounds*(*36, 15, 81, 15*)*;  
 main\_contentPane.add*(*Login\_people*)*;  
  
 JScrollPane People\_scrollPane = new JScrollPane*()*;  
 People\_scrollPane.setBounds*(*22, 40, 112, 157*)*;  
 main\_contentPane.add*(*People\_scrollPane*)*;  
  
 People\_scrollPane.setViewportView*(*People\_list*)*;  
 People\_list.setSelectionMode*(*ListSelectionModel.*SINGLE\_SELECTION)*;  
  
 Send\_personal.setBounds*(*22, 207, 112, 23*)*;  
 main\_contentPane.add*(*Send\_personal*)*;  
 Send\_personal.addActionListener*(*this*)*;  
  
 Room.setBounds*(*36, 240, 81, 15*)*;  
 main\_contentPane.add*(*Room*)*;  
  
 JScrollPane Room\_scrollPane = new JScrollPane*()*;  
 Room\_scrollPane.setBounds*(*22, 270, 112, 184*)*;  
 main\_contentPane.add*(*Room\_scrollPane*)*;  
  
 Room\_scrollPane.setViewportView*(*Room\_list*)*;  
  
 Join\_room.setBounds*(*20, 464, 114, 23*)*;  
 main\_contentPane.add*(*Join\_room*)*;  
 Join\_room.addActionListener*(*this*)*;  
  
 Make\_room.setBounds*(*22, 497, 112, 23*)*;  
 main\_contentPane.add*(*Make\_room*)*;  
 Make\_room.addActionListener*(*this*)*;  
  
 Send\_text.setBounds*(*563, 493, 81, 31*)*;  
 main\_contentPane.add*(*Send\_text*)*;  
 Send\_text.addActionListener*(*this*)*;  
  
 MainFrame.setVisible*(*true*)*;  
 Thread t = new Thread*(*this*)*;  
 t.start*()*;  
*}*

#3. public void Join\_frame() : 회원가입 창 구성

public void Join\_frame*(){* JoinFrame.setTitle*(*"Join!"*)*;  
 JoinFrame.setDefaultCloseOperation*(*JFrame.*DISPOSE\_ON\_CLOSE)*;  
 JoinFrame.setBounds*(*100, 100, 439, 271*)*;  
 JPanel join\_contentPane = new JPanel*()*;  
 join\_contentPane.setBorder*(*new EmptyBorder*(*5, 5, 5, 5*))*;  
 JoinFrame.setContentPane*(*join\_contentPane*)*;  
 join\_contentPane.setLayout*(*null*)*;  
  
 JLabel lblNewLabel = new JLabel*(*"Username"*)*;  
 lblNewLabel.setBounds*(*12, 40, 58, 15*)*;  
 join\_contentPane.add*(*lblNewLabel*)*;  
  
 JLabel lblNewLabel\_1 = new JLabel*(*"Password"*)*;  
 lblNewLabel\_1.setBounds*(*12, 88, 57, 15*)*;  
 join\_contentPane.add*(*lblNewLabel\_1*)*;  
  
 JLabel lblNewLabel\_2 = new JLabel*(*"e-mail"*)*;  
 lblNewLabel\_2.setBounds*(*12, 142, 44, 15*)*;  
 join\_contentPane.add*(*lblNewLabel\_2*)*;  
  
 ID\_textField.setBounds*(*86, 37, 220, 21*)*;  
 join\_contentPane.add*(*ID\_textField*)*;  
 ID\_textField.setColumns*(*10*)*;  
  
 Join\_passwordField.setBounds*(*86, 85, 220, 21*)*;  
 join\_contentPane.add*(*Join\_passwordField*)*;  
  
 Email\_textField.setBounds*(*86, 139, 220, 21*)*;  
 join\_contentPane.add*(*Email\_textField*)*;  
 Email\_textField.setColumns*(*10*)*;  
  
 ID\_check\_button.setBounds*(*318, 36, 97, 23*)*;  
 join\_contentPane.add*(*ID\_check\_button*)*;  
 ID\_check\_button.addActionListener*(*this*)*;  
  
 Email\_check\_button.setBounds*(*318, 138, 97, 23*)*;  
 join\_contentPane.add*(*Email\_check\_button*)*;  
 Email\_check\_button.addActionListener*(*this*)*;  
  
 Join\_complete\_button.setBounds*(*86, 196, 97, 23*)*;  
 join\_contentPane.add*(*Join\_complete\_button*)*;  
 Join\_complete\_button.addActionListener*(*this*)*;  
  
 Join\_canncel\_button.setBounds*(*209, 196, 97, 23*)*;  
 join\_contentPane.add*(*Join\_canncel\_button*)*;  
 Join\_canncel\_button.addActionListener*(*this*)*;  
  
 JoinFrame.setVisible*(*true*)*;  
*}*

#4. public void actionPerformed(ActionEvent e) : 버튼을 눌렀을 때 각각의 행동을 정의해둔 메소드입니다.

public void actionPerformed*(*ActionEvent e*) {* if*(*e.getSource*()* == Login\_button*){* out.println*(*"8/"+Login\_textField.getText*()*+"/"+String.*valueOf(*Login\_passwordField.getPassword*()))*;  
 *}* if*(*e.getSource*()* == Send\_text*){* out.println*(*"1/"+Login\_textField.getText*()*+"/"+Send\_mg.getText*())*;  
 *}* if*(*e.getSource*()* == Send\_personal*){* Chat.append*(*"Dm To "+ People\_list.getSelectedValue*()* + ":" + Send\_mg.getText*()*.trim*()*+"\n"*)*;  
 out.println*(*"2/"+People\_list.getSelectedValue*()*+"/"+Login\_textField.getText*()*+"/"+  
 Send\_mg.getText*())*;  
 *}* if*(*e.getSource*()* == Make\_room*){* out.println*(*"3/"+Login\_textField.getText*()*+"'s Room"*)*;  
 *}* if*(*e.getSource*()* == Join\_room*){* out.println*(*"4/"+Room\_list.getSelectedValue*()*+"/"+Login\_textField.getText*())*;  
 *}* if*(*e.getSource*()* == Join\_button *){* Join\_frame*()*;  
 *}* if*(*e.getSource*()* == ID\_check\_button*){* out.println*(*"5/"+ID\_textField.getText*())*;  
 *}* if*(*e.getSource*()* == Email\_check\_button*){* out.println*(*"6/"+Email\_textField.getText*())*;  
 *}* if*(*e.getSource*()* == Join\_complete\_button*){* out.println*(*"5/"+ID\_textField.getText*())*;  
 out.println*(*"6/"+Email\_textField.getText*())*;  
 if*(*id\_flag == 1 && email\_flag == 1*){* out.println*(*"7/"+ID\_textField.getText*()*+"/"+String.*valueOf(*Join\_passwordField.getPassword*())*+"/"+Email\_textField.getText*())*;  
 *}* else*{* if*(*id\_flag == 0*){* JOptionPane.*showMessageDialog(*null,"Check the ID!","Can't Confirm!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}* if*(*email\_flag == 0*){* JOptionPane.*showMessageDialog(*null,"Check the E-Mail!","Can't Confirm!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }  
 }* if*(*e.getSource*()* == Search\_button*){* JOptionPane Search = new JOptionPane*()*;  
 String Email = JOptionPane.*showInputDialog(*"Input your E-mail!"*)*;  
  
 out.println*(*"9/"+Email*)*;  
 *}* if*(*e.getSource*()* == Join\_canncel\_button*){* JoinFrame.dispose*()*;  
 *}  
}*

#5 public Client()

public Client*() {* try *{* Socket s = new Socket*(*"localhost", 5056*)*;  
 in = new BufferedReader*(*new InputStreamReader*(*s.getInputStream*()))*;  
 out = new PrintWriter*(*s.getOutputStream*()*, true*)*;  
 *}* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
}*

1. Client클래스를 초기화하는 역할이고 Client클래스가 생성되면 socket을 생성하여 Sever와 통신할 수 있게 한다.

#6. public void run()

public void run*() {* while *(*true*){* try *{* String msg = in.readLine*()*;  
  
 System.*out*.println*(*msg*)*;  
 inmessage*(*msg*)*;  
 *}* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}  
 }  
}*

1. 이 메소드는 클라이언트도 서버와 같이 통신을 위해 항상 준비되어있어야 하기 때문에 무한루프를 통해 상시 대기 시킨다.

#7. public void inmessage(String msg)

public void inmessage*(*String msg*){* st = new StringTokenizer*(*msg,"/"*)*;  
 String flag = st.nextToken*()*;  
  
 switch *(*flag*) {* case "0" -> *{* String stb\_r = st.nextToken*()*;  
 String stb = st.nextToken*()*;  
 StringTokenizer sttb\_r = new StringTokenizer*(*stb\_r, ","*)*;  
 while *(*sttb\_r.hasMoreTokens*()) {* room\_list.add*(*sttb\_r.nextToken*())*;  
 TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*room\_list*)*;  
 Vector*<*String*>* room\_list = new Vector*<*String*>(*buf\_list*)*;  
 Room\_list.setListData*(*room\_list*)*;  
 *}* StringTokenizer sttb = new StringTokenizer*(*stb, ","*)*;  
 while *(*sttb.hasMoreTokens*()) {* user\_list.add*(*sttb.nextToken*())*;  
 TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*user\_list*)*;  
 Vector*<*String*>* user\_list = new Vector*<*String*>(*buf\_list*)*;  
 People\_list.setListData*(*user\_list*)*;  
 *}  
  
 }* case "1" -> Chat.append*(*st.nextToken*()* + ":" + st.nextToken*()*.trim*()* + "\n"*)*;  
 case "100" -> *{* String buffer = st.nextToken*()*;  
 String buffer\_r = buffer + "'s Room";  
 for *(*int i = 0; i < room\_list.size*()*; i++*) {* if *(*room\_list.get*(*i*)*.equals*(*buffer\_r*)) {* room\_list.remove*(*i*)*;  
 *}  
 }* for *(*int i = 0; i < user\_list.size*()*; i++*) {* if *(*user\_list.get*(*i*)*.equals*(*buffer*)) {* user\_list.remove*(*i*)*;  
 *}  
 }* TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*user\_list*)*;  
 Vector*<*String*>* user\_list = new Vector*<*String*>(*buf\_list*)*;  
 TreeSet*<*String*>* buf\_list\_r = new TreeSet*<*String*>(*room\_list*)*;  
 Vector*<*String*>* room\_list = new Vector*<*String*>(*buf\_list\_r*)*;  
 People\_list.setListData*(*user\_list*)*;  
 Room\_list.setListData*(*room\_list*)*;  
 *}* case "3" -> *{* String stb = st.nextToken*()*;  
 room\_list.add*(*stb*)*;  
 TreeSet*<*String*>* buf\_list = new TreeSet*<*String*>(*room\_list*)*;  
 Vector*<*String*>* room\_list = new Vector*<*String*>(*buf\_list*)*;  
 Room\_list.setListData*(*room\_list*)*;  
 *}* case "5" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* JOptionPane.*showMessageDialog(*null, "You can't use this ID!", "Can't Use!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}* else *{* id\_flag = 1;  
 JOptionPane.*showMessageDialog(*null, "You can use this ID!", "Can Use!",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 *}  
 }* case "6" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* email\_flag = 1;  
 JOptionPane.*showMessageDialog(*null, "You can use this E-Mail!", "Can Use!",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 *}* else *{* JOptionPane.*showMessageDialog(*null, "You can't use this E-Mail!", "Can't Use!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }* case "7" -> *{* JOptionPane.*showMessageDialog(*null, "Welcome!", "Welcome!",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 JoinFrame.dispose*()*;  
 *}* case "8" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* stb = st.nextToken*()*;  
 LoginFrame.dispose*()*;  
 MainFrame.setTitle*(*stb + "'s Chat!"*)*;  
 out.println*(*"0/" + stb*)*;  
 Main\_frame*()*;  
 *}* else *{* JOptionPane.*showMessageDialog(*null, "You can't Login!!", "Can't Login!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }* case "9" -> *{* String stb = st.nextToken*()*;  
 if *(*stb.equals*(*"1"*)) {* JOptionPane.*showMessageDialog(*null, "ID : " + st.nextToken*()* + "\nPW : " + st.nextToken*()*, "Result",  
 JOptionPane.*PLAIN\_MESSAGE)*;  
 *}* else *{* JOptionPane.*showMessageDialog(*null, "We can't Find!!", "We can't find!",  
 JOptionPane.*ERROR\_MESSAGE)*;  
 *}  
 }* case "notice" -> Chat.append*(*st.nextToken*()* + "\n"*)*;  
 *}  
 }  
}*

1. 서버로부터 통신하여 메시지를 받게 된 경우 해당 메소드를 통해 가공을 한 뒤 클라이언트에게 보여준다.