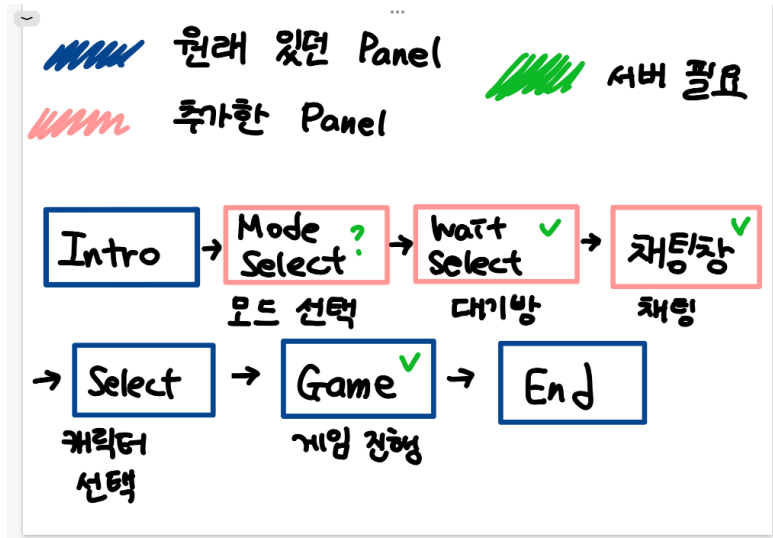


202312746 이가은

팀원 전체가 아이디어 공유를 진행하였다. 채팅방 + 대기방을 구현하였다. 각 모드와 기능에 대해 합치는 작업을 팀원 전체가 진행했다.



↑ 그림 내 인트로 패널, 모드 선택 패널을 제작한 마음이 작성한 초기 아이디어 보드이다.

사용자는 <로그인 -> 채팅방 + 대기방에 입장 -> 게임 모드로 넘어가기 -> 게임 실행>의 단계를 거치는 것으로 기획하였다. 마음과 이가은은 <로그인 -> 채팅방 + 대기방 입장>을 구현하였고 이진선과 김준형은 <게임 모드로 넘어가기 -> 게임 실행>을 구현하였다.

나(이가은)는 로그인창, 채팅 + 대기방을 만들었다. 클라이언트가 로그인을 하면 채팅 + 대기방으로 넘어가게 된다. 채팅 + 대기방은 클라이언트들끼리 채팅을 치거나 쪽지를 주고 받을 수 있고 준비 버튼을 누르면 게임 모드로 전환된다.

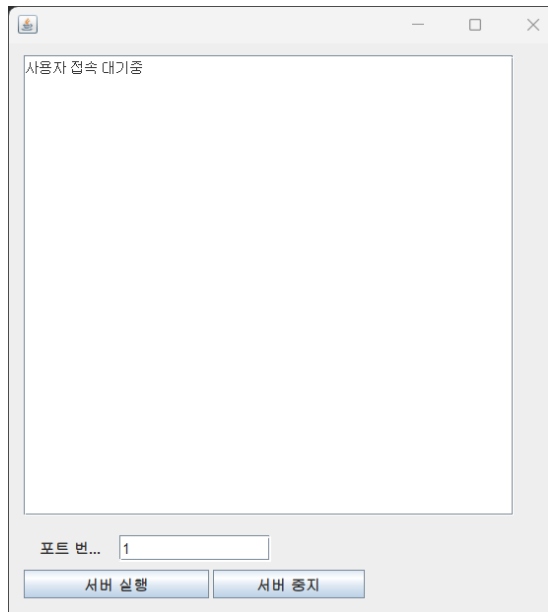
chatClient, chatServer 을 작성하였다.

자바 스윙, 소켓 통신과 멀티스레드를 이용하여 채팅방을 구현하였다.

JFrame을 이용하여 로그인창, 채팅+대기방 UI를 만들고 통신 부분의 코드를 작성하였다.

서버 코드는 클라이언트의 접속을 수락하고 여러 클라이언트 간의 통신을 처리한다.

먼저 서버창이다.



제일 먼저 실행하는 코드창이다.

JPrave으로 gui를 만들었다.

포트 번호 텍스트필드에 포트번호를 입력하여 서버를 실행한다.

현재 진행 상황을 알 수 있게끔 텍스트에리아를 만들었다.

먼저 서버 코드이다. 필요한 변수들을 전역 변수로 선언해주었다.

```

Client.java  Server.java ×
1 package panels;
2
3 import java.awt.event.ActionEvent;
4
25
26 public class Server extends JFrame implements ActionListener {
27     private JPanel contentPane;
28     private JTextField port_tf;
29     private JTextArea textArea = new JTextArea();
30     private JButton start_button = new JButton("서버 실행");
31     private JButton stop_button = new JButton("서버 중지");
32
33     private ServerSocket server_socket;
34     private Socket socket;
35     private int port;
36     private Vector uer_vc = new Vector();
37     private Vector room_vc = new Vector();
38
39     private Vector user_vc = new Vector();
40     StringTokenizer st;

```

서버 코드의 생성자다.

```

42 Server() {
43     init();
44     start();
45 }

```

Init() 메소드는 위에 있는 서버창 ui 코드이다.

```

Client.java  Server.java ×
52● private void init() {
53     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
54     setBounds(100, 100, 464, 511);
55     contentPane = new JPanel();
56     contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
57
58     setContentPane(contentPane);
59     contentPane.setLayout(null);
60
61     JScrollPane scrollPane = new JScrollPane();
62     scrollPane.setBounds(12, 10, 398, 374);
63     contentPane.add(scrollPane);
64
65     scrollPane.setViewportView(textArea);
66     textArea.setEditable(false);
67
68     JLabel lblNewLabel = new JLabel("포트 번호");
69     lblNewLabel.setBounds(25, 403, 50, 15);
70     contentPane.add(lblNewLabel);
71
72     port_tf = new JTextField();
73     port_tf.setBounds(90, 400, 123, 21);
74     contentPane.add(port_tf);
75     port_tf.setColumns(10);
76
77     start_button.setBounds(12, 428, 151, 23);
78     contentPane.add(start_button);
79
80     stop_button.setBounds(166, 428, 123, 23);
81     contentPane.add(stop_button);
82
83     /**
84     stop_button.setEnabled(false);
85
86     /**
87
88     this.setVisible(true);
89 }
90

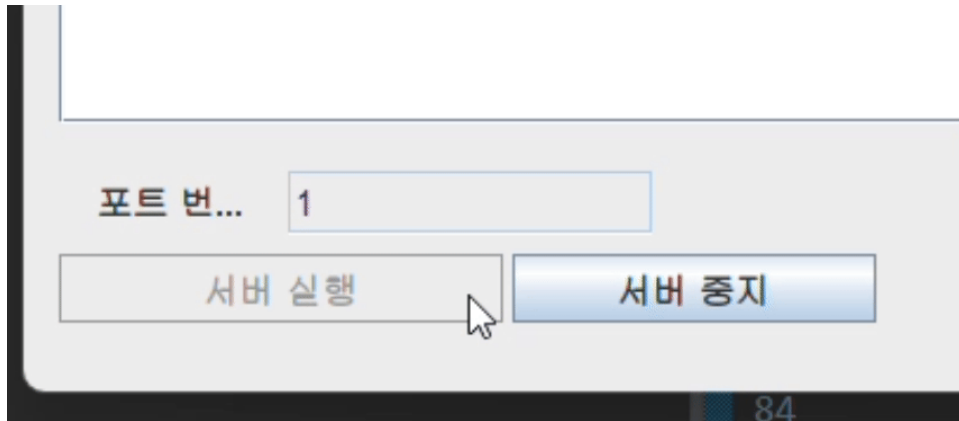
```

JFrame으로 디자인한 코드를 그대로 가져왔다.

약간 추가한 코드는 주석을 달아놓은 부분이다. 주석을 달아놓은 부분이 없으면 위의 서버창에서 서버 시작을 눌렀는지 직관적으로 알 수 없어 불편하였다.

개선하기 위해 먼저 서버 중지 버튼을 클릭할 수 없게 하였고 아래 코드들 중 서버에 연결이 되거나 서버와 연결이 끊겼을 때 `setEnabled()`을 추가하였다.

결론은 버튼을 하나만 누를 수 있게 기능을 추가하였다. (나름 고심하였다..)



Start() 메소드는 버튼 액션 리스너이다.

```
47 private void start() {  
48     start_button.addActionListener(this);  
49     stop_button.addActionListener(this);  
50 }  
51
```

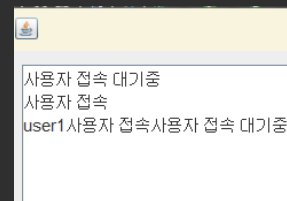
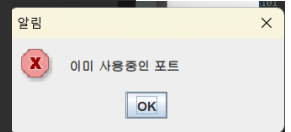
이제 서버 코드 내의 통신과 연결 부분이다.

서버를 시작하는 부분이다.

```

91● private void Server_start() {
92     try {
93         server_socket = new ServerSocket(port);
94     } catch (IOException e) {
95         JOptionPane.showMessageDialog(null, "이미 사용중인 포트", "알림", JOptionPane.ERROR_MESSAGE);
96     }
97
98     if (server_socket != null) {
99         Connection();
100     }
101 }
102
103● private void Connection() {
104●     Thread th = new Thread(new Runnable() {
105
106●         @Override
107         public void run() {
108             while (true) {
109                 try {
110                     textArea.append("사용자 접속 대기중\n");
111                     socket = server_socket.accept();
112                     textArea.append("사용자 접속\n");
113
114                     UserInfo user = new UserInfo(socket);
115
116                     user.start();
117
118                 } catch (IOException e) {
119                     //6월 13일 수정 : 114번줄 코드 지움
120                     break;
121                 }
122             }
123         }
124     });
125
126     th.start();
127 }

```



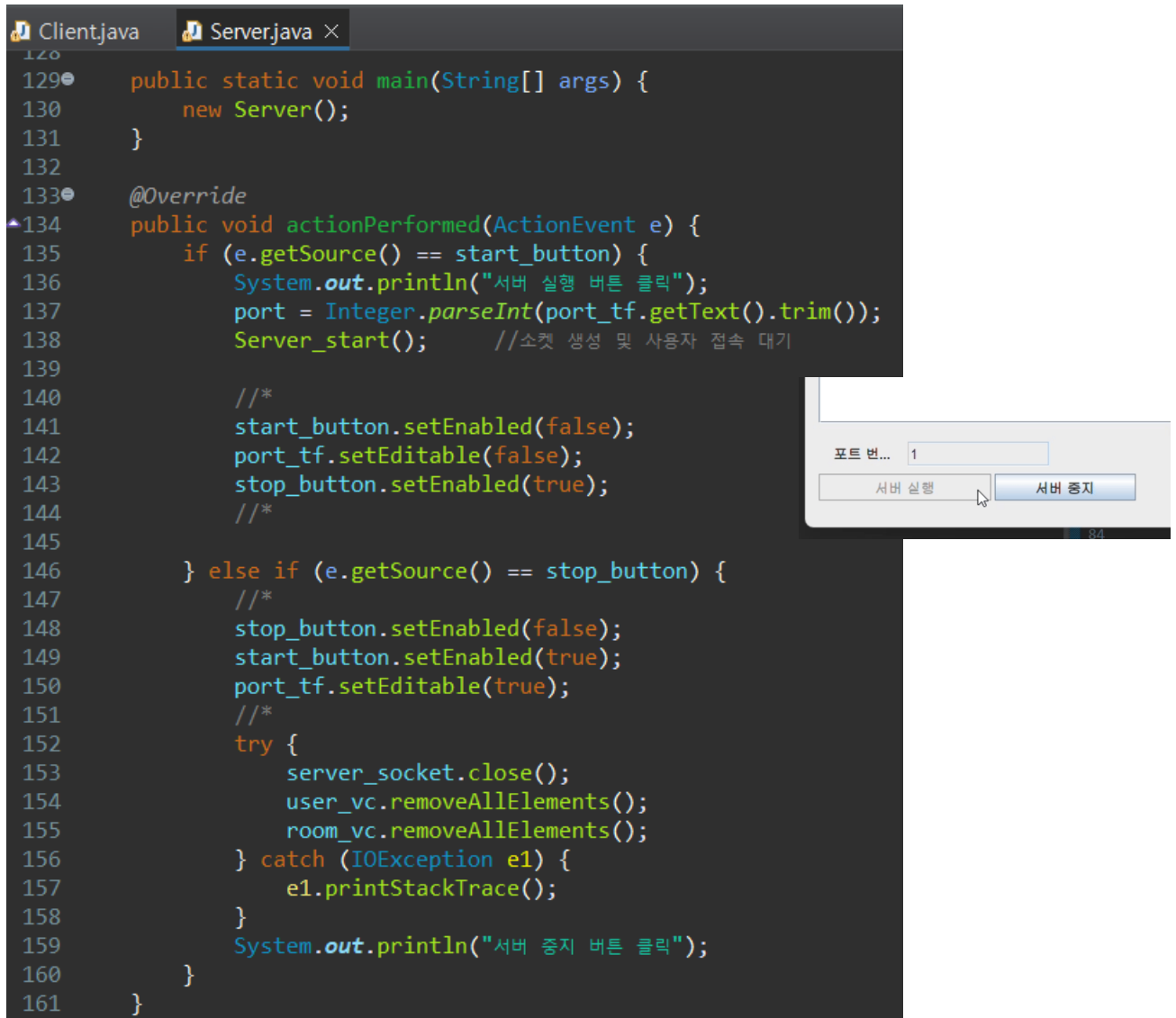
Server_start()는 'ServerSocket' 객체를 생성해 try/cath 구문으로 서버를 시작하는 역할을 한다. 만약 이미 열려있는 서버 포트에 서버를 열게 되면 알림창을 뜨게 오류를 설정하였다. 그리고 클라이언트의 접속을 기다린다.

서버가 열리면 connection() 메소드로 넘어간다.

사용자의 접속을 허락해 클라이언트를 받으면서 gui도 계속 돌아가야하므로 멀티 스레드를 생성하였다. 서버창의 용도에 맞게 이벤트가 발생하면(예. 사용자 접속) textArea.append를 이용해 서버창에 멘트를 출력한다.

또한 클라이언트가 접속하면 UserInfo 클래스를 이용해 새로운 user 정보를 생성한다.

메인 메소드와 actionPerformed 메서드이다.



버튼을 눌렀을 때의 처리를 구현하였다.

서버 시작 버튼을 눌렀을 때, 서버창에서 입력한 포트번호를 매개변수를 int형으로 받아 (getText().trim()) 해당 포트에서 서버 소켓을 연다.

주석 있는 부분이 하나의 버튼만 사용할 수 있게 제한을 걸어둔 부분이다.

서버 중지 버튼을 눌렀을 때, 서버를 닫아야하니 try/cath 구문으로 서버를 닫고 있는 정보(사용자, 채팅방)를 담고 있는 벡터를 지웠다.(vc_removeALLElements)

클라이언트와 통신을 처리하는 부분이다.

```
Client.java  Server.java ×
163● class UserInfo extends Thread {
164    private OutputStream os;
165    private InputStream is;
166    private DataOutputStream dos;
167    private DataInputStream dis;
168
169    private Socket user_socket;
170    private String Nickname = "";
171
172    private boolean RoomCh = true;
173    private boolean ReadyCh = false; // 1. readych bool 자료형 만들어서 false로 초기화
174
175●   UserInfo(Socket soc) {
176       this.user_socket = soc;
177       UserNetwork();
178   }
```

먼저 전역 변수로 필요한 변수들을 선언해주었다. 클라이언트들의 입출력 스트림이 오가니까 os, is, dos, dis의 변수를 선언하였다. 그리고 클라이언트의 소켓인 user_socket도 선언하였다. RoomCh은 채팅방의 중복된 생성을 막기 위해 설정하였다.

게임 모드를 구현하는 팀의 코드를 어떻게 연결할까 팀원 다같이 고민하였다. 결론은 내 채팅방 + 대기방에서 준비 신호를 보내고 어떻게 되었든 준비 신호가 true가 되면 모드 패널로 레이아웃으로 전환하자고 회의를 하여 ReadyCh 변수를 false로 초기화 하였다.

UserInfo 생성자를 이용해 사용자 소켓을 초기화하고 UserNetwork 메소드를 연다.


```

private void UserNetwork() {
    try {
        is = user_socket.getInputStream();
        dis = new DataInputStream(is);
        os = user_socket.getOutputStream();
        dos = new DataOutputStream(os);
        Nickname = dis.readUTF();
        textArea.append(Nickname + "사용자 접속");

        System.out.println("현재 접속된 사용자 수: " + (user_vc.size() + 1));

        Broadcast("NewUser/" + Nickname);

        for (int i = 0; i < user_vc.size(); i++) {
            UserInfo u = (UserInfo) user_vc.elementAt(i);
            send_Message("OldUser/" + u.Nickname);
        }

        for(int i = 0; i < room_vc.size(); i++) {
            RoomInfo r = (RoomInfo) room_vc.elementAt(i);
            send_Message("OldRoom/" + r.Room_name);
        }

        Broadcast("room_list_update/ ");

        user_vc.add(this);

        Broadcast("user_list_update/ ");

    } catch (IOException e) {
        JOptionPane.showMessageDialog(null, "stream 설정 에러", "알림", JOptionPane.ERROR_MESSAGE);
    }
}

```

현재 접속된 사용자 수: 1

UserNetwork() 메소드는 클라이언트와 초기 연결을 처리한다. 클라이언트와 네트워크 스트림을 주고 받는다.

사용자 소켓으로부터 입출력 스트림을 생성한다. 또한 readUTF()를 이용해 사용자의 닉네임을 읽어온다. 클라이언트가 접속하면 서버 콘솔에 문구를 출력하고 (System.out.println) 모든 사용자에게 새로운 클라이언트가 접속했다고 알린다. (Broadcast)

첫번째 for문은 새로운 사용자들에게 'user_vc'에 저장된 기존 사용자의 정보를 전송하는 부분이고

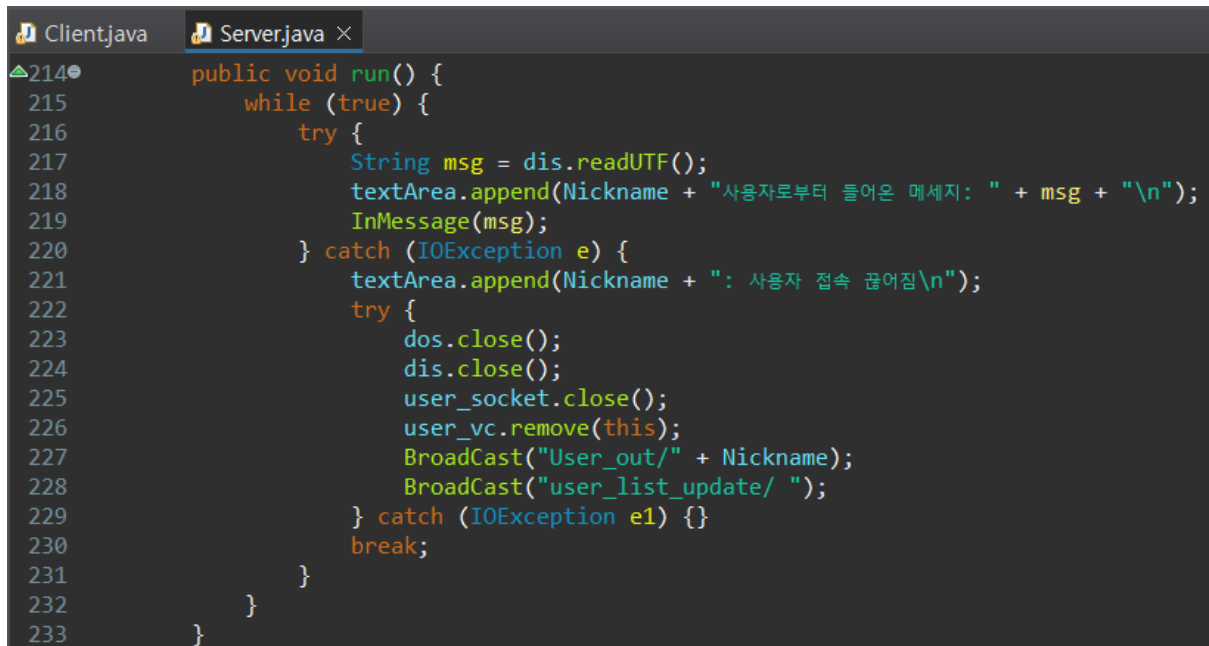
두번째 for문은 새로운 사용자에게 'room_vc'에 저장된 현재 존재하는 모든 방의 정보를 전송하는 부분이다.

방의 정보를 전송하면 모든 사용자에게(Broadcast) 방 목록이 업데이트되었다고 알린다.

현재 사용자를 사용자 벡터에 추가하고 (user_vc.add(this)) 모든 사용자에게 (Broadcast) 유저 목록이 업데이트되었다고 알린다.

Catch는 오류 처리 부분이다.

run() 메소드이다. 클라이언트와 지속적인 통신을 유지하는 역할이다.



```
Client.java Server.java ×
214 public void run() {
215     while (true) {
216         try {
217             String msg = dis.readUTF();
218             textArea.append(Nickname + "사용자로부터 들어온 메세지: " + msg + "\n");
219             InMessage(msg);
220         } catch (IOException e) {
221             textArea.append(Nickname + ": 사용자 접속 끊어짐\n");
222             try {
223                 dos.close();
224                 dis.close();
225                 user_socket.close();
226                 user_vc.remove(this);
227                 BroadCast("User_out/" + Nickname);
228                 BroadCast("user_list_update/ ");
229             } catch (IOException e1) {}
230             break;
231         }
232     }
233 }
```

UserInfo 클래스의 일부분이다. 클라이언트의 메시지를 수신하고 통신하는 스레드 부분이다.

입력스트림을 'UTF'로 인코딩 된 메시지로 읽어 msg로 저장해 클라이언트의 메시지를 수신한다.

InMessage로 수신한 메시지를 처리한다.

예외가 발생하면 사용자의 접속이 끊어졌음을 알리고 데이터 스트림과 소켓을 닫는다. 그리고 사용자 벡터에서 현재 사용자를 제거하고 모든 사용자에게 사용자가 나갔음을 알린다.

또한 클라이언트들로부터 지속적으로 메시지를 수신하기 위해 while(true)를 사용하였다.

로직은 매우 비슷하다. 클라이언트와 통신을 시도해서 통신이 잘 되면 스트림을 처리한다. 만약 예외가 발생한다면 서버창에 문구를 띄우고 서버에 알리고 소켓을 닫고.. 모든 사용자에게 알린다.

InMessage() 메소드이다.

```
Client.java Server.java X
235 private void InMessage(String str) {
236     st = new StringTokenizer(str, "/");
237
238     String protocol = st.nextToken();
239     String message = st.nextToken();
240
241     System.out.println("프로토콜: " + protocol);
242     System.out.println("메세지: " + message);
243
244     if (protocol.equals("Note")) {
245         String note = st.nextToken();
246         System.out.println("받는 사람: " + message);
247         System.out.println("보낼 내용: " + note);
248
249         for(int i = 0; i < user_vc.size(); i++) {
250             UserInfo u = (UserInfo)user_vc.elementAt(i);
251
252             if(u.Nickname.equals(message)) {
253                 u.send_Message("Note/" + Nickname + "/" + note);
254             }
255         }
256     }
257
258     else if (protocol.equals("CreateRoom")) {
259         for (int i = 0; i < room_vc.size(); i++) {
260             RoomInfo r = (RoomInfo)room_vc.elementAt(i);
261
262             if (r.Room_name.equals(message)) {
263                 send_Message("CreateRoomFail/ok");
264                 RoomCh = false;
265                 break;
266             }
267         }
268
269         if (RoomCh) {
270             RoomInfo new_room = new RoomInfo(message, this);
271             room_vc.add(new_room);
272             send_Message("CreateRoom/" + message);
273
274             BroadCast("New_Room/" + message);
275         }
276         RoomCh = true;
277     }
}
```

```

Clientjava  Serverjava X
278
279     else if (protocol.equals("Chatting")) {
280         String msg = st.nextToken();
281         for(int i = 0; i < room_vc.size(); i++) {
282             RoomInfo r = (RoomInfo)room_vc.elementAt(i);
283
284             if(r.Room_name.equals(message)) {
285                 r.BroadCast_Room("Chatting/" + Nickname + "/" + msg);
286             }
287         }
288     }
289
290     else if (protocol.equals("JoinRoom")) {
291         for(int i = 0; i < room_vc.size(); i++) {
292             RoomInfo r = (RoomInfo)room_vc.elementAt(i);
293             if(r.Room_name.equals(message)) {
294                 r.BroadCast_Room("Chatting/알림/*****" + Nickname + "님이 입장*****");
295                 r.Add_User(this);
296                 send_Message("JoinRoom/" + message);
297             }
298         }
299     }
300
301     else if (protocol.equals("ready")) {
302         for (int i = 0; i < room_vc.size(); i++) {
303             RoomInfo r = (RoomInfo) room_vc.elementAt(i);
304             if (r.Room_name.equals(message)) {
305                 r.BroadCast_Room("ready/" + Nickname + "님이 준비버튼을 눌렀습니다.");
306             }
307         }
308     }
309
310 }
311

```

클라이언트들로부터 수신된 메시지를 처리하는 부분이다.

비슷한 부분이 계속 반복되는 중요한 로직만 설명을 해보자면.. StringTokenizer를 사용해 전달받은 문자열을 토큰으로 저장한다. (첫번째 토큰은 protocol, 두번째 프로토콜은 message로) 그리고 각 프로토콜에 따른 동작을 수행한다.

예를 들어 Note 프로토콜은 쪽지 보내기 기능이다. 클라이언트 개인 메시지를 다른 클라이언트에게 보내는 기능이다. 개인 메시지를 'note'로 지정하고, 사용자 'user_vc'를 순회 하면서 해당 사용자가 있는지 확인한 후 메시지를 전송하는 기능이다.

CreateRoom은 새로운 채팅방을 생성한다. 이미 존재하는 방이 있는지(RoomCh) 방의 정보가 저장된 'room_vc'만큼 반복문을 이용해 순회를 해야한다. 그리고 앞서 말한 로직대로 기능을 수행하고(방을 만들고) 또 모든 사용자들에게 알리고.. 의 반복이다.

Chatting은 채팅방 내에서 채팅을 하는 기능이다. 또 'room_vc'만큼 반복문을 이용해 방

이름을 확인하고 해당 방에 메시지를 모든 사용자에게 알린다.

JoinRoom은 특정 채팅방에 사용자가 입장한다.

Ready는 사용자가 준비 상태임을 알린다.

'Broadcast'는 "모든 사용자"에게 메시지를 전송하는 메소드이고

'send_Message'는 개별 사용자에게 메시지를 전송하는 메소드이다 .

```
332 private void Broadcast(String str) {  
333     for (int i = 0; i < user_vc.size(); i++) {  
334         UserInfo u = (UserInfo) user_vc.elementAt(i);  
335         u.send_Message(str);  
336     }  
337 }  
338  
339 private void send_Message(String str) {  
340     try {  
341         dos.writeUTF(str);  
342     } catch (IOException e) {  
343         e.printStackTrace();  
344     }  
345 }  
346 }  
347
```

역시 사용자의 정보는 벡터에 저장되어있으므로 벡터를 이용해 접근해야하는 것을 유의해야한다.(초반에 헤맸었다.)

RoomInfo 클래스다

```

348●   class RoomInfo {
349       private String Room_name;
350       private Vector Room_user_vc = new Vector();
351
352●       RoomInfo(String str, UserInfo u) {
353           this.Room_name = str;
354           this.Room_user_vc.add(u);
355       }
356
357●       public void BroadCast_Room(String str) {
358           for(int i = 0; i < Room_user_vc.size(); i++) {
359               UserInfo u = (UserInfo)Room_user_vc.elementAt(i);
360               u.send_Message(str);
361           }
362       }
363
364●       public void Add_User(UserInfo u) {
365           this.Room_user_vc.add(u);
366       }
367   }
368 }

```

채팅방에 관련된 부분이다. 채팅방의 정보(채팅방 이름, 채팅방 내에 있는 사용자)를 관리하는 부분이다. 생성자를 이용해서 방 이름과 첫 번째 사용자를 받는다.

'BroadCast_Room'은 채팅방 내의 모든 사용자에게 메시지를 알린다.

벡터를 순회하며 각 사용자들에게 메시지를 전송한다.

'Add_User'는 채팅방에 사용자를 추가하는 메소드이다. 전달받은 'UserInfo' 객체를 벡터에 추가한다.

checkAllReady() 메서드

```

312     private void checkAllReady() {
313         boolean allReady = true;
314         for (int i = 0; i < user_vc.size(); i++) {
315             UserInfo u = (UserInfo) user_vc.elementAt(i);
316             if (!u.ReadyCh) {
317                 allReady = false;
318                 break;
319             }
320         }
321         if (allReady) {
322             for (int i = 0; i < user_vc.size(); i++) {
323                 UserInfo u = (UserInfo) user_vc.elementAt(i);
324                 u.ReadyCh = true;
325             }
326             send_Message("allReady"); // 모든 사용자가 준비되었음을 알림
327         }
328     }
329 }

```

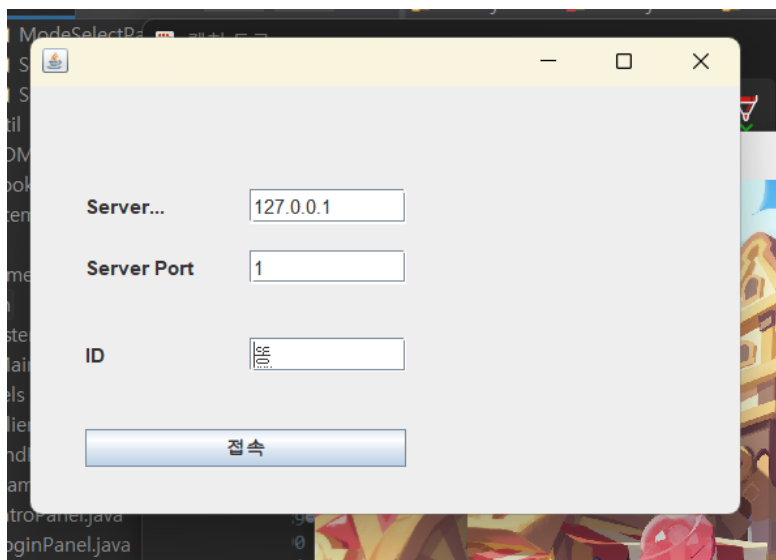
모든 사용자가 준비 상태인지 확인하는 부분이다.

For 반복문을 이용해서 'user_vc'에 있는 모든 사용자를 순회한다. 각 사용자의 'ReadyCh' 변수를 확인하여 모두 true이면 'allReady'를 'true'로 설정한다.

클라이언트 코드는 서버와 연결을 설정한 후, 데이터 스트림을 통해 메시지를 주고 받습니다.

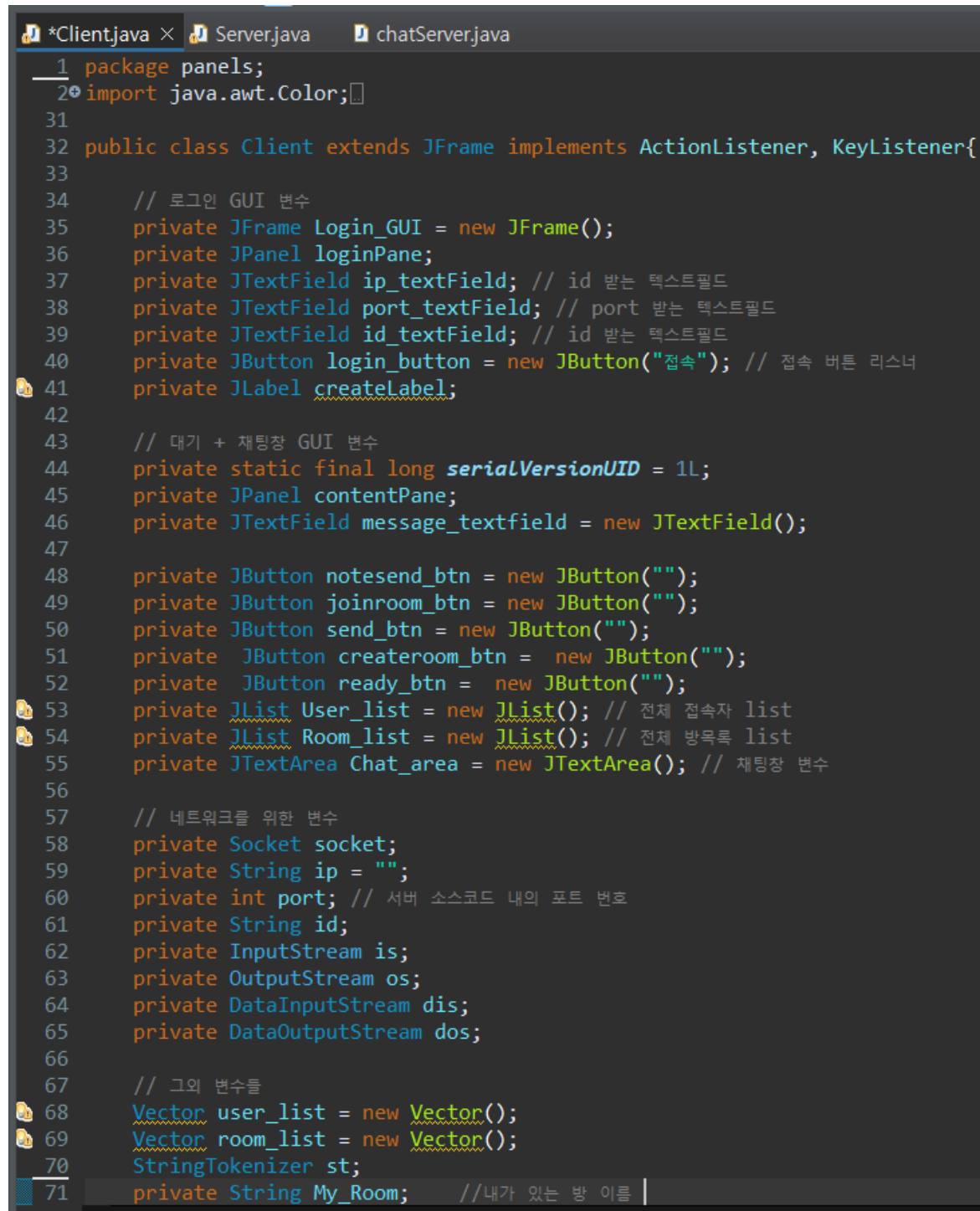
이제 클라이언트 코드이다.

클라이언트 코드를 실행하게 되면 먼저 로그인 창이 뜨게 된다.



다른 컴퓨터에서도 접속 가능하도록 server ip 주소, 채팅방 서버를 열기 위한 server port 번호, 채팅방 + 대가방에서 이름으로쓰일 ID 를 입력받게 로그인 창을 만들었다.

전역 변수 선언 부분



```
1 package panels;
2 import java.awt.Color;
31
32 public class Client extends JFrame implements ActionListener, KeyListener{
33
34     // 로그인 GUI 변수
35     private JFrame Login_GUI = new JFrame();
36     private JPanel loginPane;
37     private JTextField ip_textField; // id 받는 텍스트필드
38     private JTextField port_textField; // port 받는 텍스트필드
39     private JTextField id_textField; // id 받는 텍스트필드
40     private JButton login_button = new JButton("접속"); // 접속 버튼 리스너
41     private JLabel createLabel;
42
43     // 대기 + 채팅창 GUI 변수
44     private static final long serialVersionUID = 1L;
45     private JPanel contentPane;
46     private JTextField message_textfield = new JTextField();
47
48     private JButton notesend_btn = new JButton("");
49     private JButton joinroom_btn = new JButton("");
50     private JButton send_btn = new JButton("");
51     private JButton createroom_btn = new JButton("");
52     private JButton ready_btn = new JButton("");
53     private JList User_list = new JList(); // 전체 접속자 list
54     private JList Room_list = new JList(); // 전체 방목록 list
55     private JTextArea Chat_area = new JTextArea(); // 채팅창 변수
56
57     // 네트워크를 위한 변수
58     private Socket socket;
59     private String ip = "";
60     private int port; // 서버 소스코드 내의 포트 번호
61     private String id;
62     private InputStream is;
63     private OutputStream os;
64     private DataInputStream dis;
65     private DataOutputStream dos;
66
67     // 그외 변수들
68     Vector user_list = new Vector();
69     Vector room_list = new Vector();
70     StringTokenizer st;
71     private String My_Room; //내가 있는 방 이름
```

필요한 변수들을 선언해준다.

채팅창 필드에 메시지를 입력하고 엔터를 누르면 전송되도록 하기 위해 keylistener를 넣었다.

생성자

```
*Client.java × Server.java chatServer.java
72
73 Client() { // 생성자 메소드
74     Login_init(); // 로그인창 화면 구성 메소드
75     start();
76 }
77
```

Login_init() 로그인창과

Start() 액션 버튼 리스너 이다.

Login_init() 메서드이다.

```
*Client.java × Server.java chatServer.java
77
78 private void Login_init() {
79     Login_GUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
80     ImageIcon icon = new ImageIcon("img/새로 추가된 이미지/login_background.png");
81     Image backgroundImage = icon.getImage();
82     Login_GUI.setBounds(100, 100, 800, 500); // 프레임 크기 설정
83     loginPane = new JPanel();
84     loginPane.setBorder(new EmptyBorder(5, 5, 5, 5));
85     Login_GUI.setContentPane(loginPane);
86     loginPane.setLayout(null);
87
88     JLabel backgroundLabel = new JLabel(new ImageIcon
89         (backgroundImage.getScaledInstance(800, 500, Image.SCALE_SMOOTH))); // 이미지 크기 조정하여 설정
90     backgroundLabel.setBounds(0, 0, 800, 500);
91     loginPane.add(backgroundLabel);
92
93     Font labelFont = new Font("Arial", Font.BOLD, 22); // 라벨 글씨 크기 설정
94
95     // 서버 IP 입력 레이블
96     JLabel lblNewLabel = createLabel("Server IP", 130, 150, labelFont);
97     backgroundLabel.add(lblNewLabel);
98
99     // 서버 Port 입력 레이블
100    JLabel lblNewLabel_1 = createLabel("Server Port", 130, 200, labelFont);
101    backgroundLabel.add(lblNewLabel_1);
102
103    // ID 입력 레이블
104    JLabel lblNewLabel_2 = createLabel("ID", 130, 250, labelFont);
105    backgroundLabel.add(lblNewLabel_2);
106
107    // 서버 IP 입력 텍스트 필드
108    JTextField ip_textField = new JTextField();
109    ip_textField.setHorizontalAlignment(JTextField.CENTER); // 가운데 정렬
110    ip_textField.setBounds(300, 150, 200, 30); // 가로 중앙 정렬
111    backgroundLabel.add(ip_textField);
112    ip_textField.setColumns(10);
113}
```

```
*Client.java × Server.java chatServer.java
113
114 // 서버 Port 입력 텍스트 필드
115 port_textField = new JTextField();
116 port_textField.setHorizontalAlignment(JTextField.CENTER); // 가운데 정렬
117 port_textField.setBounds(300, 200, 200, 30); // 가로 중앙 정렬
118 backgroundLabel.add(port_textField);
119 port_textField.setColumns(10);
120
121 // ID 입력 텍스트 필드
122 id_textField = new JTextField();
123 id_textField.setHorizontalAlignment(JTextField.CENTER); // 가운데 정렬
124 id_textField.setBounds(300, 250, 200, 30); // 가로 중앙 정렬
125 backgroundLabel.add(id_textField);
126 id_textField.setColumns(10);
127
128 // 접속 버튼에 이미지 설정
129 ImageIcon loginIcon = new ImageIcon("img/새로 추가된 이미지/loginbutton.png");
130 login_button = new JButton(loginIcon);
131 login_button.setBounds(300, 300, loginIcon.getIconWidth(), loginIcon.getIconHeight()); // 버튼 크기와 위치 설정
132 login_button.setContentAreaFilled(false); // 버튼의 배경 투명하게
133 login_button.setBorderPainted(false); // 버튼의 테두리 투명하게
134 backgroundLabel.add(login_button);
135
136
137 Login_GUI.setVisible(true);
138 }
139
```

앞에 있던 로그인 창의 ui 부분이다.

다음은 로그인창의 접속을 누르면 뜨는 채팅+대기방이다



이상하게도 처음 접속한 유저의 채팅창에 똑 같은 내용이 2번씩 뜨길래 벡터 반복의 오류인 것 같으나.. 오류를 해결하지 못하였다.

그래서 아예 채팅방에 참여를 해야 채팅을 할 수 있게 전송 버튼을 setEnabled로 조정하였다. 채팅방 내에서 반복하여 메시지를 송수신하게 만들었더니 채팅이 2번씩 뜨는 오류가 수정되었다.

Main_init()

```
*Client.java × Server.java chatServer.java
154
155 private void Main_init() {
156     Toolkit kit = Toolkit.getDefaultToolkit();
157     Image img = kit.getImage("img/새로 추가된 이미지/cookie.png");
158     setIconImage(img); // 아이콘 설정
159
160     setTitle("대기 + 채팅방");
161     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
162     setBounds(100, 100, 650, 500);
163
164     contentPane = new JPanel();
165     contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
166     contentPane.setBackground(new Color(0xF3E9DF)); // 배경색 설정
167
168     setContentPane(contentPane);
169     contentPane.setLayout(null);
170
171     JLabel lblNewLabel = new JLabel("전 체 접 속 자");
172     lblNewLabel.setBounds(37, 32, 88, 15);
173     Font font = new Font("배달의민족 주아", Font.PLAIN, 15);
174     lblNewLabel.setFont(font);
175     contentPane.add(lblNewLabel);
176
177
178     User_list.setBounds(28, 57, 97, 113);
179     contentPane.add(User_list);
180
181     notesend_btn.setIcon(new ImageIcon("img/새로 추가된 이미지/쪽지_보내기.png"));
182     notesend_btn.setBounds(28, 180, 97, 40);
183     notesend_btn.setBorderPainted(false);
184     notesend_btn.setContentAreaFilled(false);
185     notesend_btn.setFocusPainted(false);
186     notesend_btn.addActionListener(new ActionListener() {
187     public void actionPerformed(ActionEvent e) {
188     }
189     });
```

```

190
191     contentPane.add(notesend_btn);
192
193     JLabel lblNewLabel_1 = new JLabel("채팅방 목록");
194     lblNewLabel_1.setBounds(37, 235, 88, 15);
195     lblNewLabel_1.setFont(font);
196     contentPane.add(lblNewLabel_1);
197
198
199     Room_list.setBounds(28, 260, 97, 100);
200     contentPane.add(Room_list);
201
202     joinroom_btn.setIcon(new ImageIcon("img/새로 추가된 이미지/채팅방_참여.png"));
203     joinroom_btn.setBounds(20, 370, 110, 40);
204     joinroom_btn.setBorderPainted(false);
205     joinroom_btn.setContentAreaFilled(false);
206     joinroom_btn.setFocusPainted(false);
207     joinroom_btn.addActionListener(new ActionListener() {
208         public void actionPerformed(ActionEvent e) {
209         }
210     });
211     contentPane.add(joinroom_btn);
212
213
214     createroom_btn.setIcon(new ImageIcon("img/새로 추가된 이미지/채팅방_만들기.png"));
215     createroom_btn.setBounds(20, 411, 110, 40);
216     createroom_btn.setBorderPainted(false);
217     createroom_btn.setContentAreaFilled(false);
218     createroom_btn.setFocusPainted(false);
219     createroom_btn.addActionListener(new ActionListener() {
220         public void actionPerformed(ActionEvent e) {
221         }
222     });
223     contentPane.add(createroom_btn);

```

```

226
227 Chat_area.setBounds(153, 59, 454, 350);
228 contentPane.add(Chat_area);
229 //6.13추가된부분
230 Chat_area.setEditable(false);
231 //여기까지
232
233
234
235 message_textfield.setBounds(153, 425, 371, 21);
236 contentPane.add(message_textfield);
237 message_textfield.setColumns(10);
238 /**
239 message_textfield.setEnabled(false);
240
241 /**
242
243
244 send_btn.setIcon(new ImageIcon("img/새로 추가된 이미지/전송.png"));
245 send_btn.setBounds(536, 415, 70, 42);
246 send_btn.setBorderPainted(false);
247 send_btn.setContentAreaFilled(false);
248 send_btn.setFocusPainted(false);
249 send_btn.addActionListener(new ActionListener() {
250     public void actionPerformed(ActionEvent e) {
251     }
252 });
253 contentPane.add(send_btn);
254 send_btn.setEnabled(false);
255
256 ready_btn.setIcon(new ImageIcon("img/새로 추가된 이미지/ready.png"));
257 ready_btn.setBounds(490, 5, 100, 50);
258 ready_btn.setBorderPainted(false);
259 ready_btn.setContentAreaFilled(false);
260 ready_btn.setFocusPainted(false);
261 ready_btn.addActionListener(new ActionListener() {
262     public void actionPerformed(ActionEvent e) {
263     }
264 });
265 contentPane.add(ready_btn);
266 ready_btn.setEnabled(true);

```

```

268 JLabel lblNewLabel_2 = new JLabel("방이름");
269 lblNewLabel_2.setBounds(153, 32, 88, 15);
270 lblNewLabel_2.setFont(font);
271 contentPane.add(lblNewLabel_2);
272
273
274 this.setVisible(true);
275 }

```

Start()

```
*Client.java × Server.java chatServer.java
277 private void start() {
278     login_button.addActionListener(this); // 로그인 버튼 리스너
279     notesend_btn.addActionListener(this); // 쪽지보내기 버튼 리스너
280     joinroom_btn.addActionListener(this); // 채팅방 참여 버튼 리스너
281     createroom_btn.addActionListener(this); // 전송 버튼 리스너
282     send_btn.addActionListener(this); // 방만들기 버튼 리스너
283     ready_btn.addActionListener(this); // 준비 버튼 리스너
284     message_textfield.addKeyListener(this);
285 }
286
```

Network() 메서드이다.

```
*Client.java × Server.java chatServer.java
286
287 private void Network() {
288     try {
289         socket = new Socket(ip, port);
290         if (socket != null) { // 정상적으로 소켓이 연결되었을 경우
291             Connection();
292             // 로그인 창 닫고 메인 창 열기
293             Login_GUI.dispose();
294             Main_init();
295         }
296     } catch (UnknownHostException e) {
297         JOptionPane.showMessageDialog(null, "연결 실패!!", "알림", JOptionPane.ERROR_MESSAGE);
298     } catch (IOException e) {
299         JOptionPane.showMessageDialog(null, "연결 실패!!", "알림", JOptionPane.ERROR_MESSAGE);
300     }
301 }
302
```

서버와 데이터 통신하는 부분이다. dispose()를 이용해서 로그인창이 열리고 -> 채팅창이 뜨게 설정하였다.

Ip와 port번호를 매개 변수로 받아 소켓을 생성한다.

Connection() 메소드 부분이다.

```

304 private void Connection() { // 실질적인 메소드 연결 부분
305     try {
306         is = socket.getInputStream();
307         dis = new DataInputStream(is);
308
309         os = socket.getOutputStream();
310         dos = new DataOutputStream(os);
311     } catch (IOException e) { // 예러저리부분
312         JOptionPane.showMessageDialog(null, "연결 실패!!", "알림", JOptionPane.ERROR_MESSAGE);
313     } // Stream 설정 끝
314
315
316     send_message(id);
317
318     // User_list에 사용자 추가
319     user_list.add(id);
320
321     // 스레드 설정 안 하면 GUI가 죽어버림
322     Thread th = new Thread(new Runnable() {
323     @Override
324     public void run() {
325         while (true) {
326             try {
327                 String msg = dis.readUTF(); // 메시지 수신
328                 System.out.println("서버로부터 수신된 메시지: " + msg);
329                 inmessage(msg);
330
331             } catch (IOException e) {
332                 try {
333                     os.close();
334                     is.close();
335                     dis.close();
336                     socket.close();
337                     JOptionPane.showMessageDialog(null, "서버와 접속이 끊어짐", "알림", JOptionPane.ERROR_MESSAGE);
338                 } catch (IOException e1) {}
339                 break;
340             }
341         }
342     }
343 });
344
345     th.start();
346 }

```

실질적으로 서버가 클라이언트와 연결하는 부분이다.

사실 서버코드와 별 다를게 없다. 소켓으로부터 입출력 스트림을 얻는다.

Send_message를 이용해 서버에 처음 접속할 때 입력한 사용자의 id를 전송한다.

사용자의 id를 받으면 사용자가 접속한 것이니 사용자의 정보를 user_list에 추가한다.

또한 메시지를 계속 받으면서 gui도 돌아가야하니까 스레드를 생성하여 서버로부터 메시지를 수신합니다. 그리고 메시지를 받으면 inmessage를 통해 이벤트에 따라 작업을 수행한다.

Inmessage 메소드

```

*Client.java × Server.java chatServer.java
347
348 private void inmessage(String str) { // 서버로부터 들어오는 모든 메시지
349     st = new StringTokenizer(str, "/");
350
351     String protocol = st.nextToken();
352     String Message = st.nextToken();
353
354     System.out.println("프로토콜: " + protocol);
355     System.out.println("내용: " + Message);
356
357     if (protocol.equals("NewUser")) {
358         user_list.add(Message);
359     }
360
361     else if (protocol.equals("OldUser")) {
362         user_list.add(Message);
363     }
364
365     else if (protocol.equals("Note")) {
366         String note = st.nextToken();
367         System.out.println(Message + "사용자로부터 온 쪽지" + note);
368         JOptionPane.showMessageDialog(null, note, Message + "님으로부터 쪽지", JOptionPane.CLOSED_OPTION);
369     }
370
371     else if (protocol.equals("user_list_update")) {
372         User_list.setListData(user_list);
373     }
374
375     else if (protocol.equals("CreateRoom")) {
376         My_Room = Message;
377
378         /**
379         message_textfield.setEnabled(true);
380         send_btn.setEnabled(true);
381
382         joinroom_btn.setEnabled(false);
383         createroom_btn.setEnabled(false);
384         /**
385     }

```



```

387         else if (protocol.equals("CreateRoomFail")) {
388             JOptionPane.showMessageDialog(null, "방 만들기 실패!!", "알림", JOptionPane.ERROR_MESSAGE);
389         }
390
391         else if (protocol.equals("New_Room")) {
392             room_list.add(Message);
393             Room_list.setListData(room_list);
394         }
395
396         else if (protocol.equals("Chatting")) {
397             String msg = st.nextToken();
398             Chat_area.append(Message + " : " + msg + "\n");
399         }
400
401         else if (protocol.equals("OldRoom")) {
402             room_list.add(Message);
403             Room_list.setListData(room_list);
404         }
405
406         else if (protocol.equals("room_list_update")) {
407             Room_list.setListData(room_list);
408         }
409
410         else if (protocol.equals("JoinRoom")) {
411             My_Room = Message;
412
413             /**
414             message_textfield.setEnabled(true);
415             send_btn.setEnabled(true);
416
417             joinroom_btn.setEnabled(false);
418             createroom_btn.setEnabled(false);
419             /**
420
421             JOptionPane.showMessageDialog(null, "채팅방에 입장했습니다", "알림", JOptionPane.INFORMATION_MESSAGE);
422         }else if (protocol.equals("ready")) {
423             Chat_area.append(Message + "\n");
424             //ready_btn.setEnabled(false);
425         }
426
427     }
428 }

```

서버로부터 들어온 메시지를 처리하는 부분이다. 서버 코드 내의 InMessage 메서드와 똑같다. 'stringTokenizer'를 사용해 프로토콜 메시지에 따라 다양한 이벤트 처리를 실행한다.

send_message() , main 메소드이다.

```

*Client.java × Server.java chatServer.java
430 // """"서버""""에 메세지 보내는 부분
431 private void send_message(String str) {
432     try {
433         dos.writeUTF(str);
434     } catch (IOException e) { // 에러처리부분
435         e.printStackTrace();
436     }
437 }
438
439 public static void main(String[] args) {
440     new Client();
441 }

```

서버에 메시지를 보내는 부분이다.

그리고 메인 메소드이다.

버튼 액션 처리 부분이다.

```

*Client.java × Server.java chatServer.java
442
443 @Override
444 public void actionPerformed(ActionEvent e) {
445     if (e.getSource() == login_button) {
446         System.out.println("로그인 버튼 클릭");
447         if(ip_textField.getText().length() == 0) {
448             ip_textField.setText("IP를 입력해주세요");
449             ip_textField.requestFocus();
450         }
451         else if(port_textField.getText().length() == 0) {
452             port_textField.setText("Port번호를 입력해주세요");
453             port_textField.requestFocus();
454         }
455         else if (id_textField.getText().length() == 0) {
456             id_textField.setText("ID를 입력해주세요");
457             id_textField.requestFocus();
458         }
459         else {
460             ip = ip_textField.getText().trim(); // ip 받는 부분
461             port = Integer.parseInt(port_textField.getText().trim());
462             id = id_textField.getText().trim(); // id 받는 부분
463             Network();
464         }
465     } else if (e.getSource() == notesend_btn) {
466         System.out.println("쪽지 보내기 버튼 클릭");
467         String user = (String) User_list.getSelectedValue();
468         String note = JOptionPane.showInputDialog("보낼 메시지");
469         if (note != null) {
470             send_message("Note/" + user + "/" + note);
471         }
472         System.out.println("받는 사람: " + user + "| 보낼 내용" + note);
473     } else if (e.getSource() == joinroom_btn) {
474         String JoinRoom = (String) Room_list.getSelectedValue();
475         send_message("JoinRoom/" + JoinRoom);
476         System.out.println("채팅방 참여 버튼 클릭");
477     } else if (e.getSource() == createroom_btn) {
478         String roomname = JOptionPane.showInputDialog("방이름");
479         if (roomname != null) {
480             send_message("CreateRoom/" + roomname);
481         }
482         System.out.println("방 만들기 버튼 클릭");
483     } else if (e.getSource() == send_btn) {
484         send_message("Chatting/" + My_Room + "/" + message_textfield.getText().trim()); // "Chatting + 채팅 내용"
485         message_textfield.setText("");
486         message_textfield.requestFocus();
487         System.out.println("전송 버튼 클릭");
488     }
489     else if (e.getSource() == ready_btn) {
490         send_message("ready/" + id);
491         System.out.println(id + ": 준비 버튼 클릭");
492         Chat_area.append(id + "님이 준비 버튼을 눌렀습니다.\n");
493         ready_btn.setEnabled(false);
494     }
495 }

```

서버 코드의 설명과 똑같으니 생략한다.

달라진 점은 준비 버튼을 (ready_btn)을 눌렀을 때 한 번 누르면 setEnabled가 false가 되

어 다시 못 누르게 처리하였다.

그리고 키를 눌렀을 때 반응하는 리스너들이다.

```
497● @Override
498 public void keyTyped(KeyEvent e) {
499     // TODO Auto-generated method stub
500 }
501
502
503● @Override
504 public void keyPressed(KeyEvent e) {
505     // TODO Auto-generated method stub
506 }
507
508
509● @Override
510 public void keyReleased(KeyEvent e) {
511     if (e.getKeyCode() == 10) {
512         send_message("Chatting/" + My_Room + "/" + message_textfield.getText().trim()); // "Chatting" + 채팅 내용
513         message_textfield.setText("");
514         message_textfield.requestFocus();
515     }
516     // TODO Auto-generated method stub
517 }
518
519 }
```

엔터키를 누르면 채팅 텍스트필드에 있는 내용이 전달된다.