□경진대회 결과보고서

- 11분반 6조 - **※해당란에 "∨" 표기하여 주세요.**

<table>
<tr><td>단과대학명</td><td colspan="4">공과대학</td></tr>
<tr><td>과 목 명</td><td colspan="4">융합디지털미디어 (11분반)</td></tr>
<tr><td rowspan="4">학 번</td><td>1560058</td><td rowspan="4">성 명</td><td colspan="2">황주호</td></tr>
<tr><td>1560028</td><td colspan="2">오승화</td></tr>
<tr><td>1560050</td><td colspan="2">전진원</td></tr>
<tr><td>1560007</td><td colspan="2">김선준</td></tr>
<tr><td>주 전 공</td><td>학부(과)</td><td>정보통신융합공학부</td><td>전공</td><td>정보통신</td></tr>
<tr><td>작품명</td><td colspan="4">OMG(Oh-Mok Game)</td></tr>
<tr><td>작품 설명</td><td colspan="4">바둑판에 두 사람이 번갈아 돌을 놓아 가로나 세로, 대각선으로 다섯 개의 연속된 돌을 놓으면 이기는 놀이이다. 19x19 크기의 바둑판에서 하기도 하나 정식 경기는 보통 15x15 크기의 판을 이용한다. 잡거나 움직이는 돌이 없으므로 종이와 펜을 이용하여 진행할 수 있다. - 출처 : 위키 백과 -</td></tr>
<tr><td>연구방법</td><td colspan="4">- 처음 시작시 흑돌과 백돌을 랜덤으로 선택된 후 이용자가 19x19 바둑판에 순차적으로 돌을 두어 자기 자신의 돌이 5개가 연속적으로 나오면 게임에서 승리한다.</td></tr>
</table>

위와 같이 최종 보고서를 제출 합니다.

2018년 12월 06일

제 출 자 황주호 (인 또는 서명)

지도 교수 정윤수 (인 또는 서명)

목원대학교 교무처장 귀하

1. 작품명

OMG (Oh-Mok Game)

2. 과제의 목표

수업 시간때 배운 것을 활용하여 만들 수 있는 작품을 목표로 하였고, 배운것들을 초대한 활용하여 완성도 있는 작품을 만드는 것이 저희조의 최종 목표이며, 한명의 낙오자 없이 역할을 효율적으로 분담하여 조원 전원이 참여하여 각각 개인의 프로그램 개발 능력 또한 향상될 것입니다.

3. 작품 내용

1. 서버가 실행이 되어있는 상태에서, 클라이언트를 실행하게 되면 클라이언트가 서버에 접속을 하게 된다.
2. 서버에 연결이 성공적으로 이루어지면, 닉네임을 정하고 대기실에 입장할 수 있다.
3. 대기실에 입장되면 채팅이 가능하고, 원하는 방번호를 입력해 방에 들어갈 수 있다.
4. 위와 같은 방법으로 다른 클라이언트도 접속하여, 특정 방에 두명이 접속하면 게임을 시작할 수 있다. 또한 게임 도중에도 채팅을 할 수 있다.
5. 게임이 시작되면, 난수를 이용하여 흑돌 백돌을 선정하고, 흑돌이 먼저 선공으로 시작한다.
6. 게임 진행 중에 기권을 할 수 있으며, 해당 판이 종료되면 다시 게임을 시작할 수 있다.

4. 과제 수행 일정

DATE	개발 일정
11.17.	주제선정 및 역할 분담
11.18.	게임 알고리즘 및 소켓 네트워킹에 대한 조사
11.19. ~ 11.23.	소켓 네트워킹 구성 [채팅 / 대기실 부분]
11.24. ~ 11.25.	소켓 네트워킹 구성 [오목 게임 부분]
11.26. ~ 11.30.	GUI를 이용한 게임의 전체적인 인터페이스 구성
12.01. ~ 12.03	전체적인 프로그램 오류 수정
12.05.	보고서 작성 및 발표자료 준비

5. 과제 참가자의 역할 분담 계획

* 참가자별로 분담할 내용을 정리

조장			
이름	황주호	학번	1560058
역할	소켓 네트워킹 및 오류 수정		

조원			
이름	김선준	학번	1560007
역할	GUI 인터페이스		

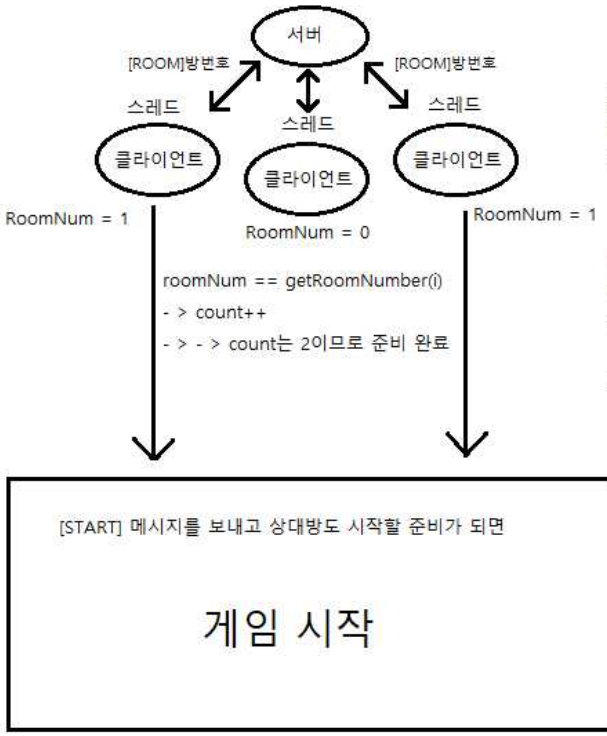
조원			
이름	전진원	학번	1560050
역할	채팅 프로그램 구성		

조원			
이름	오승화	학번	1560028
역할	오목 알고리즘 담당		

6. 알고리즘

서버와 클라이언트는 문자열을 주고 받는다.
앞에 붙는 단어를 인식하여 해당 문자열이 의미하는 명령을 실행한다.

[NAME]~ : userName 설정 [EXIT]~ : 퇴장 [FULL] : 방 인원 초과 [MSG]~ : 메시지
[ROOM]~ : 방 번호 설정 [ENTER]~ : 입장 [STONE] ~ + ~ : 오목게임등 ...

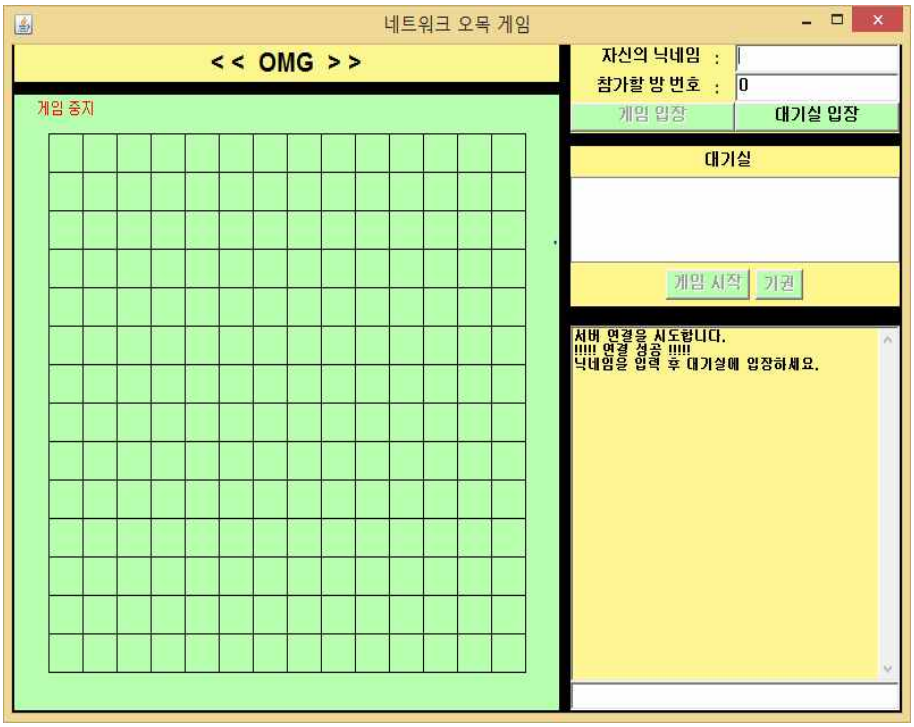


클라이언트 A가 "[ROOM]방번호" 문자열을
서버로 보내면 서버에서 해당 방번호를 클라이언트
A와 연결된 스레드의 방번호를 지정하는 변수에 대입

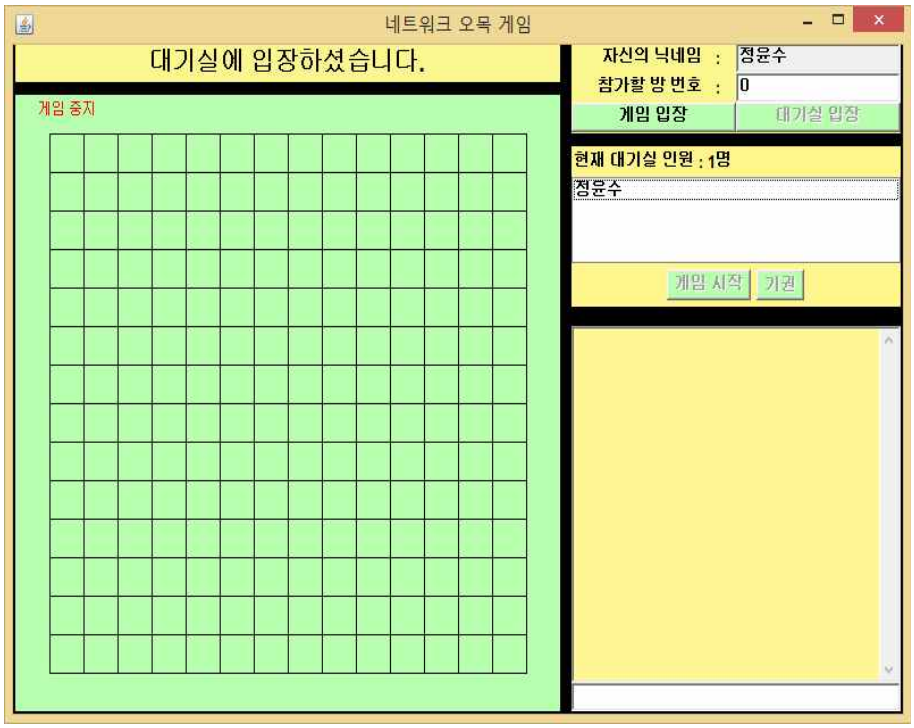
i번째 스레드의 방번호를 받아와
방번호가 RoomNum과 같을 경우 Count를
1 증가시키는 작업을 스레드의 개수 만큼 반복하고

Count가 2라면 게임이 시작될 준비가 된다.

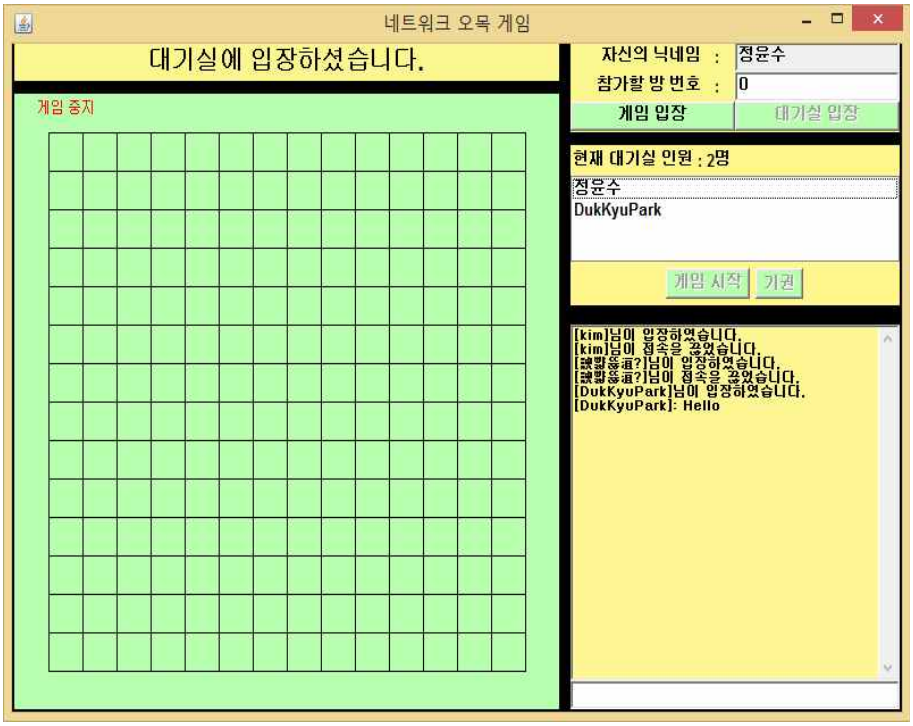
7.실행



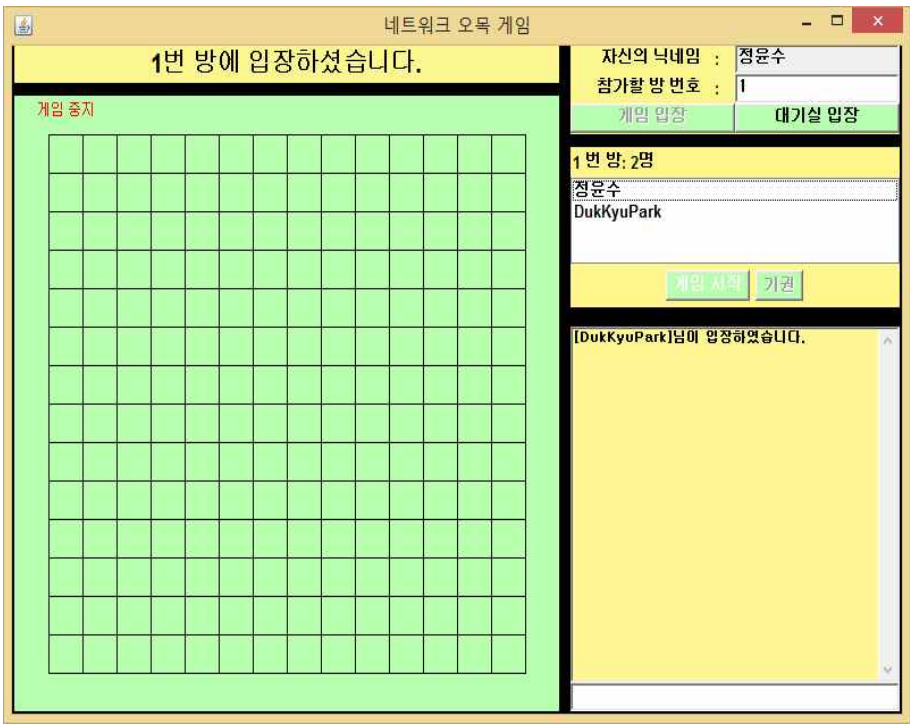
처음 실행 화면



닉네임 입력 후 대기실 입장 버튼 클릭으로 대기실 입장

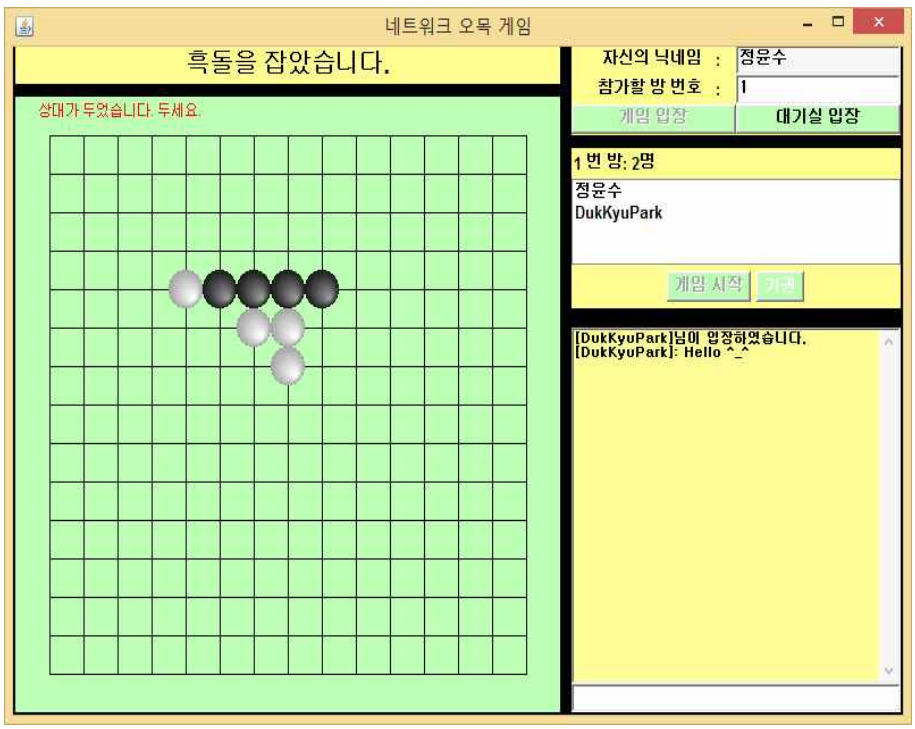


대기실에서 클라이언트 간에 채팅을 한다.

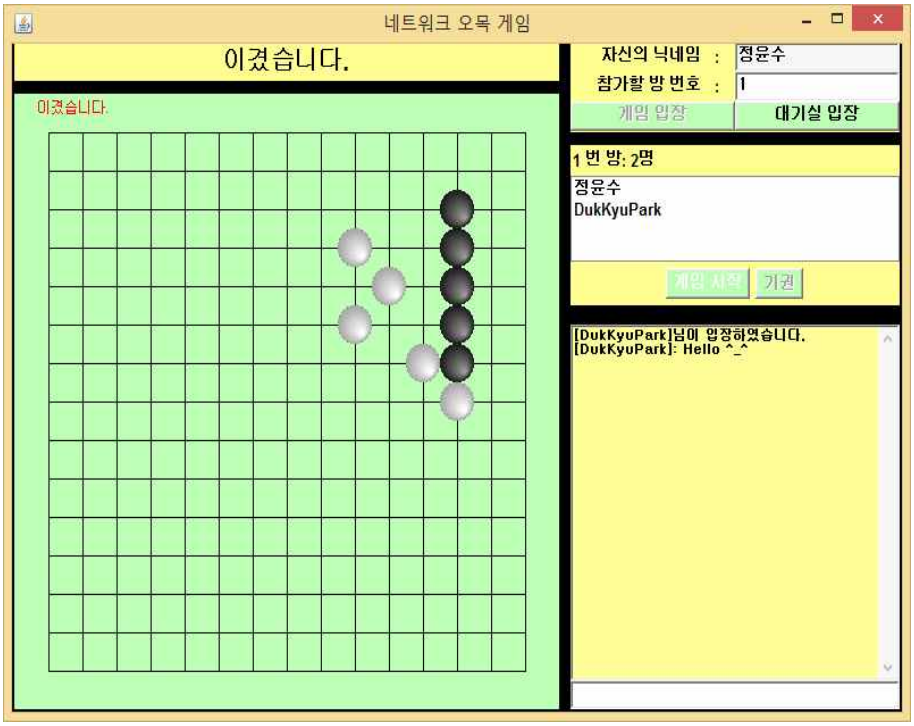


참가할 방 번호에 번호를 입력 후 게임 입장 버튼을 눌러 방에 입장한다.

방에 입장 후 상대방이 들어오면 게임 시작 버튼을 눌러 게임을 진행한다.



랜덤으로 돌 색을 부여받으며 흑돌이 선공으로 게임을 진행한다.
게임 도중 기권 버튼을 눌러 게임을 패배할 수도 있다.



오목 룰에 따라 승패가 결정되면 승패 문구가 출력되고 게임이 끝난다.
끝난 후 버튼 클릭으로 게임을 재시작 하거나, 대기실로 나갈 수 있다.

8. 소스

SERVER PART

```
import java.net.ServerSocket;
import java.net.*;
import java.io.*;
import java.util.*;

public class SERVER {

	private ServerSocket server;

	private Random rnd = new Random(); // 흑돌 백돌을 정하기 위핸 난수입니다.

	private BManager bMan = new BManager(); // 메시지를 전달하는
관리자입니다.

	public SERVER() {
	}

	void startServer() { // 서버를 실행합니다.

	try {

	server = new ServerSocket(0707); // 서버 소켓을 생성합니다.

	System.out.println("서버소켓이 생성되었습니다.");

	while (true) { // 무한반복합니다.

	// 클라이언트와 연결되는 스레드를 얻습니다.

	Socket socket = server.accept();

	// 스레드를 생성합니다.

	Omok_Thread ot = new Omok_Thread(socket);

	// 스레드를 실행합니다.
	ot.start();

	// Vector를 상속받는 BManager 객체인 bMan에 스레드를 추가한다.

	bMan.add(ot);

	// 접속자 수를 표시해줍니다.

	System.out.println("접속자 수: " + bMan.size());

	}

	} catch (Exception e) {

	System.out.println(e);

	}

	}
```

```
public static void main(String[] args) {
SERVER server = new SERVER();
server.startServer();
}
// 클라이언트와 통신하는 스레드 클래스를 생성합니다.
class Omok_Thread extends Thread {
private int roomNumber = -1; // 방 번호를 설정하는데에 쓰입니다.
private String userName = null; // 유저 이름을 설정하는데에 쓰입니다.
private Socket socket; // 소켓
// 게임이 준비되었는지를 지정합니다, true이면 게임을 시작할 준비가 되었음을
의미합니다.
private boolean ready = false;
private BufferedReader rd; // 입력 스트림인 BufferedReader입니다.
private PrintWriter pw; // 출력 스트림 PrintWrite입니다.
Omok_Thread(Socket socket) { // 생성자를 만듭니다.
this.socket = socket;
}
Socket getSocket() { // 소켓을 반환하는 메소드입니다.
return socket;
}
int getRoomNumber() { // 해당 쓰레드의 방 번호를 반환하는 메소드입니다.
return roomNumber;
}
String getUserName() { // 유저 이름을 반환하는 메소드입니다.
return userName;
}
boolean isReady() { // 준비가 되어있는지를 반환해줍니다.
return ready;
}
public void run() { // 스레드가 실행 시 실제로 실행되는 부분입니다.
try {
// 소켓에서 값을 읽어올 때 쓰이는 부분입니다.
rd = new BufferedReader(
new InputStreamReader(socket.getInputStream()));
// 소켓에 값을 보낼 때 쓰이는 부분입니다.
```

```
        pw = new PrintWriter(socket.getOutputStream(), true);
        String msg; // 클라이언트의 메시지입니다.
        while ((msg = rd.readLine()) != null) {
        // msg를 읽어 왔는데 "[NAME]"으로 시작된다면
        if (msg.startsWith("[NAME]")) {
        userName = msg.substring(6); // 앞부분을 자르고 userName을 정한다.
        }
        // msg를 읽어 왔는데 "[ROOM]"으로 시작되면
        else if (msg.startsWith("[ROOM]")) {
        int roomNum = Integer.parseInt(msg.substring(6));
        // 방 번호를 앞부분을 자르고 나머지를 정수로 전환하여 대입합니다.
        if (!bMan.isFull(roomNum)) { // 방이 꽉 차있지 않다면(2명이 꽉찬
상태입니다)
        // 현재 방의 다른 유저에게 유저의 퇴장을 알립니다.
        if (roomNumber != -1) // 방의 번호가 초기값이 아니라면
        bMan.sendToOthers(this, "[EXIT]" + userName); // sendToOthers 메소드를
호출합니다.
        // 유저의 방 번호를 위에서 읽어온 새로운 방 번호로 지정합니다.
        roomNumber = roomNum;
        // 유저에게 입장이 가능하다는 것을 알려줍니다.
        pw.println(msg); // msg를 소켓을 통해 전송합니다.
        // 유저에게 새로 입장한 방에 있는 유저 이름 리스트를 전송합니다.
        pw.println(bMan.getNamesInRoom(roomNumber));
        // 새 방에 있는 다른 유저에게 유저의 입장을 알려줍니다.
        bMan.sendToOthers(this, "[ENTER]" + userName);
        }
        else
        pw.println("[FULL]"); // 유저에게 방이 사람이 꽉찼음을 알려줍니다.
        }
        // "[STONE]" 메시지를 상대에게 전송합니다.
        else if (roomNumber >= 1 && msg.startsWith("[STONE]"))
        bMan.sendToOthers(this, msg);
        // 대화 메시지를 방에 전송합니다.
        else if (msg.startsWith("[MSG]"))
        bMan.sendToRoom(roomNumber,
```

```
"[" + userName + "] : " + msg.substring(5));
// "[START]" 메시지라면
else if (msg.startsWith("[START]")) {
ready = true; // 게임을 시작할 준비를 의미하는 ready를 true로 바꿉니다.
// 다른 유저도 게임을 시작한 준비가 되었으면
if (bMan.isReady(roomNumber)) {
// 흑과 백을 정하고 유저와 상대편에게 전송합니다.
int a = rnd.nextInt(2); // 1~2사이의 정수 난수를 a에 대입합니다.
if (a == 0) {
pw.println("[COLOR]BLACK"); // 0 이면 해당 문자를 전송합니다.
bMan.sendToOthers(this, "[COLOR]WHITE");
}
else {
pw.println("[COLOR]WHITE"); // 1이면 해당 문자를 전송합니다.
bMan.sendToOthers(this, "[COLOR]BLACK");
}
}
}
// 유저가 게임을 중지하는 메시지를 보내면
else if (msg.startsWith("[STOPGAME]"))
ready = false;
// 유저가 게임을 기권하는 메시지를 보내면
else if (msg.startsWith("[DROPGAME]")) {
ready = false;
// 상대편에게 유저의 기권을 알립니다.
bMan.sendToOthers(this, "[DROPGAME]");
}
// 유저가 이겼다는 메시지를 보내면
else if (msg.startsWith("[WIN]")) {
ready = false;
// 유저에게 메시지를 보냅니다.
pw.println("[WIN]");
```

```
        // 상대편에는 졌음을 알립니다.
        bMan.sendToOthers(this, "[LOSE]");
        }
        }
        } catch (Exception e) {
        } finally {
        // 연결을 닫아주는 부분입니다.
        try {
        bMan.remove(this);
        if (rd != null)
        rd.close();
        if (pw != null)
        pw.close();
        if (socket != null)
        socket.close();
        rd = null;
        pw = null;
        socket = null;
        System.out.println(userName + "님이 접속을 끊었습니다.");
        System.out.println("접속자 수: " + bMan.size());
        // 유저가 접속을 끊었음을 같은 방에 알려줍니다..
                bMan.sendToRoom(roomNumber, "[DISCONNECT]" + userName);
        } catch (Exception e) {
        }
        }
        }
        }
        class BManagerextends Vector{// 메시지를 전달하는 클래스입니다. Vector를
상속받는 중요한 클래스입니다.
        BManager() {
        }
        void add(Omok_Thread ot) { // 스레드를 Vector에 추가합니다.
        super.add(ot);
        }
        void remove(Omok_Thread ot) { // 스레드를 Vector에서 제거합니다.
        super.remove(ot);
        }
        Omok_Thread getOT(int i) { // i번째 스레드를 반환합니다.
```

```
        return (Omok_Thread) elementAt(i);

        }

        Socket getSocket(int i) { // i번째 스레드의 소켓을 반환합니다.

        return getOT(i).getSocket();

        }

        // i번째 스레드와 연결된 클라이언트에게 메시지를 전송합니다.

        void sendTo(int i, String msg) {

        try {

        PrintWriter pw = new PrintWriter(getSocket(i).getOutputStream(),
true);

        pw.println(msg); // msg를 보냅니다.

                } catch (Exception e) {
        }

        }

        int getRoomNumber(int i) { // i번째 스레드의 방 번호를 반환합니다.

        return getOT(i).getRoomNumber();

        }

        synchronized boolean isFull(int roomNum) { // 방이 찼는지 알아보는
메소드입니다.

        if (roomNum == 0)
        return false; // 대기실은 인원 제한이 없기 때문에 대기실이면 그냥
빠져나옵니다.

        // 다른 방은 2명 이상 입장할 수 없습니다.

        int count = 0; // int형의 count를 초기값 0으로 선언합니다.

        for (int i = 0; i < size(); i++) // 반복문을 메소드의 개수 (Vector의 요소
개수)만큼 반복합니다.

        if (roomNum == getRoomNumber(i)) // 만약 방 번호와 i번 스레드의 방 번호가
같다면

        count++; // count를 1씩 증가시킵니다.

        if (count >= 2) // 만약 count가 2보다 크다면 true를 반환합니다.
        return true;

        return false; // 아니면 false를 반환합니다.

        }

        // 해당하는 번호의 방에 msg를 전송합니다.

        void sendToRoom(int roomNum, String msg) {

        for (int i = 0; i < size(); i++)

        if (roomNum == getRoomNumber(i))
```

```
        sendTo(i, msg);
        }
        // ot와 같은 방에 있는 다른 유저에게 msg를 전달합니다.
        void sendToOthers(Omok_Thread ot, String msg) {
        for (int i = 0; i < size(); i++)
        if (getRoomNumber(i) == ot.getRoomNumber() && getOT(i) != ot) // i번
스레드의 방번호와 ot의 방번호가 같으며 ot 자기 자신이 아닐 경우
        sendTo(i, msg); // i번 스레드에게 msg를 보내는 메소드를 실행합니다.
        }
        // 게임을 시작할 준비가 되었는가를 반환합니다.
        // 두 명의 유저 모두 준비된 상태이면 true를 반환합니다.
        synchronized boolean isReady(int roomNum) {
        int count = 0; // count를 선언합니다.
        for (int i = 0; i < size(); i++) // 반복문을 스레드의 개수만큼 반복합니다.
        if (roomNum == getRoomNumber(i) && getOT(i).isReady()) // i번 스레드와
roomNum이 같으며 i번스레드가 준비가 되어있다면
        count++; // count를 1 증가시킵니다.
        if (count == 2) // count가 2라면 true를 반환합니다.
        return true;
        return false;
        }
        // roomNum방에 있는 유저들의 이름을 반환합니다.
        String getNamesInRoom(int roomNum) {
        StringBuffer sb = new StringBuffer("[PLAYERS]"); // 문자열 버퍼를
만듭니다.
        for (int i = 0; i < size(); i++) // 메소드의 개수만큼 반복합니다.
        if (roomNum == getRoomNumber(i)) // i번 메소드와 roomNum이 같다면 sb에
i번 메소드의 유저이름과 "\t"를 문자열 뒤에 붙여줍니다.
        sb.append(getOT(i).getUserName() + "\t");
        return sb.toString(); // 문자열 버퍼를 문자열로 변환하여 반환합니다.
        }
        }
}
```

CLIENT PART

```
import java.awt.*;

import java.net.*;

import java.io.*;

import java.util.*;

import java.awt.event.*;

class BOARDextends Canvas { // 오목판을 구현하는 클래스

	public static final int BLACK = 1, WHITE = -1; // 흑과 백을 나타내는 상수

	// true이면 유저가 돌을 놓을 수 있는 상태를 의미하며 false이면 유저가 돌을
놓을 수 없는 상태를 의미합니다.
	private boolean enable = false;

	private boolean running = false; // 게임이 진행 중인가를 나타내는 변수

	private int[][] omokpan; // 오목판 배열을 선언합니다.

	private int size; // size는 격자의 가로 또는 세로 개수입니다. 이
게임에서는 15로 지정합니다.

	private int color = BLACK; // 유저의 돌 색깔

	private int pixel; // 격자의 크기입니다.

	private String Stat_Text = "게임 시작전.."; // 게임의 진행 상황을 나타내는
문자열

	private PrintWriter pw; // 상대편에게 메시지를 전달하기 위한 스트림

	private Graphics gboard, gbuff; // 캔버스와 버퍼를 위한 그래픽스 객체

	private Image buff; // 더블 버퍼링을 위한 버퍼

	Image BStone_img =
Toolkit.getDefaultToolkit().getImage(System.getProperty("user.dir") +
"\\images\\Black.png");
	Image WStone_img =
Toolkit.getDefaultToolkit().getImage(System.getProperty("user.dir") +
"\\images\\White.png");

	BOARD(int s, int c) { // 오목판의 생성자입니다. (s=15, c=30)

	this.size = s;
	this.pixel = c;

	omokpan = new int[size + 2][]; // 오모판의 크기를 정합니다.

	for (int i = 0; i < omokpan.length; i++)

	omokpan[i] = new int[size + 2];

	setBackground(new Color(184, 255, 176)); // 오목판의 배경색을 정합니다.

	setSize(size * (pixel + 1) + size, size * (pixel + 1) + size); //
오목판의 크기를 계산한다.
```

```
// GUI에서의 오목판의 마우스 이벤트 처리를 위한 부분입니다.
addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent me) { // 마우스를 누르면
if (!enable)
return; // 유저가 누를 수 없는 상태이면 빠져나옵니다.
// 마우스 좌표를 omokpan의 좌표로 계산합니다.
int x = (int) Math.round(me.getX() / (double) pixel);
int y = (int) Math.round(me.getY() / (double) pixel);
// 돌을 놓을 수 없는 위치를 클릭했다면 이벤트에서 빠져나옵니다.
if (x == 0 || y == 0 || x == size + 1 || y == size + 1)
return;
// 이미 해당 좌표에 돌이 놓여져 있다면 이벤트에서 빠져나옵니다.
if (omokpan[x][y] == BLACK || omokpan[x][y] == WHITE)
return;
// 상대편에게 놓은 돌의 좌표를 전송합니다.
pw.println("[STONE]" + x + " " + y);
omokpan[x][y] = color;
// 게임에서 이겼는지를 판별하는 부분입니다.
if (check(new Point(x, y), color)) {
Stat_Text = "승리!";
pw.println("[WIN]");
} // 이긴게 아니라면 계속해서 게임을 진행합니다.
        else
Stat_Text = "상대방의 턴입니다..";
repaint(); // 오목판을 다시 새로 그립니다.
// 상대방의 턴이기 때문에 유저가 둘 수 없는 상태로 만듭니다.
// 상대편이 두면 enable이 true가 되어 유저가 둘 수 있게 됩니다.
enable = false;
}
});
}
public boolean isRunning() { // 게임의 현재 진행 상태를 반환하는
메소드입니다.
return running;
}
```

```
        public void startGame(String col) { // 게임 시작을 담당하는 메소드입니다.
        running = true;
        if (col.equals("BLACK")) { // 흑돌이 선택되었을 때 선공을 줍니다.
        enable = true;
        color = BLACK;
        Stat_Text = "게임이  시작되었습니다. 흑돌이므로 선공을 시작합니다.";
        }
        else { // 백돌이 선택되었을 때 후공으로 상대를 기다립니다.
        enable = false;
        color = WHITE;
        Stat_Text = "게임이 시작되었습니다. 백돌이므로 상대방을 기다립니다.";
        }
        }
        public void stopGame() { // 게임 중단을 담당하는 메소드입니다.
        reset(); // 오목판을 리셋합니다.
        pw.println("[STOPGAME]"); // 상대방에게 게임을 멈춘다는 문자열을 보냅니다.
        enable = false; // 돌을 둘 수 없습니다.
        running = false; // 게임이 중단되었음을 의미합니다.
        }
        public void putOpponent(int x, int y) { // 상대방이 돌을 놓는 것을
담당하는 메소드입니다.
        omokpan[x][y] = -color; // 매개변수로 받아온 x, y 좌표에 -color값을
대입합니다.
        Stat_Text = "상대방이 돌을 두었습니다. 당신의 차례입니다, 돌을
놓아주세요.";
        repaint(); // 판을 다시 그립니다.
        }
        public void setEnable(boolean enable) {
        this.enable = enable;
        }
        public void setWriter(PrintWriter pw) {
        this.pw = pw;
        }
        public void update(Graphics g) { // repaint를 호출하면 자동으로
호출됩니다.
```

```
        paint(g); // paint를 호출합니다.
        }
        public void paint(Graphics g) { // 오목판 화면을 그리는 메소드입니다.
        if (gbuff == null) { // 버퍼가 없다면 버퍼를 생성합니다.
        buff = createImage(getWidth(), getHeight());
        gbuff = buff.getGraphics();
        }
        drawBoard(g); // 오목판을 그리는 메소드를 호출합니다..
        }
        public void reset() { // 오목판을 초기화하는 메소드입니다.
        for (int i = 0; i < omokpan.length; i++)
        for (int j = 0; j < omokpan[i].length; j++)
        omokpan[i][j] = 0; // 오목판 2차원 배열을전부 0으로 초기화시킵니다.
        Stat_Text = "게임 중지"; // 게임 중지라는 텍스트를 화면에 띄웁니다.
        repaint(); // 오목판을 다시 그립니다.
        }
        private void drawLine() { // 오목판에 선을 그어주는 메소드입니다.
        gbuff.setColor(Color.black); // 검은색으로 긋습니다.
        for (int i = 1; i <= size; i++) {
        gbuff.drawLine(pixel, i * pixel, pixel * size, i * pixel);
        gbuff.drawLine(i * pixel, pixel, i * pixel, pixel * size);
        } // pixel 사이즈만큼 간격을 주어 선을 긋습니다.
        }
        private void drawBlack(int x, int y) { // 흑 돌을 그리는 메소드입니다.
        Graphics2D gbuff = (Graphics2D) this.gbuff;
        gbuff.drawImage(BStone_img, x * pixel - pixel / 2, y * pixel - pixel /
2, pixel, pixel, this);
        // 이미지를 매개변수 x, y를 받아와서 x,y 좌표에 삽입합니다.
        }
        private void drawWhite(int x, int y) { // 백 돌을 그리는 메소드입니다.
        gbuff.drawImage(WStone_img, x * pixel - pixel / 2, y * pixel - pixel /
2, pixel, pixel, this);
        // 이미지를 매개변수 x, y를 받아와서 x,y 좌표에 삽입합니다.
        }
        private void drawStones() { // 오목판에 놓여진 돌들을 전부 그려주는
메소드입니다.
```

```
        for (int x = 1; x <= size; x++)
        for (int y = 1; y <= size; y++) {
        if (omokpan[x][y] == BLACK) // 해당 좌표가 1[블랙]이면 검은돌 그리기를 호출합니다.
        drawBlack(x, y);
        else if (omokpan[x][y] == WHITE) // 해당 좌표가 -1[화이트]이면 흰돌 그리기를 호출합니다.
        drawWhite(x, y);
        }
        }
        synchronized private void drawBoard(Graphics g) { // 오목판을 그리는 메소드입니다.
        // 버퍼에서 오목판을 먼저 그리고 버퍼에 그린 이미지를 오목판에 옮겨 그립니다.
        gbuff.clearRect(0, 0, getWidth(), getHeight());
        drawLine(); // 선을 그립니다.
        drawStones(); // 돌을 그립니다.
        gbuff.setColor(Color.red); // 붉은색으로 설정합니다.
        gbuff.drawString(Stat_Text, 20, 15); // 상태 메시지를 띄웁니다.
        g.drawImage(buff, 0, 0, this); // 버퍼에 그린 것을 옮겨 그립니다.
        }
        private boolean check(Point p, int col) { // 오목게임의 승리 알고리즘입니다.
        if (count(p, 1, 0, col) + count(p, -1, 0, col) == 4)
        return true;
        if (count(p, 0, 1, col) + count(p, 0, -1, col) == 4)
        return true;
        if (count(p, -1, -1, col) + count(p, 1, 1, col) == 4)
        return true;
        if (count(p, 1, -1, col) + count(p, -1, 1, col) == 4)
        return true;
        return false;
        }
        private int count(Point p, int dx, int dy, int col) { // 오목게임의 돌을 세는 알고리즘입니다.
        int i = 0;
```

```
        for (; omokpan[p.x + (i + 1) * dx][p.y + (i + 1) * dy] == col; i++)
        ;

        return i;

        }
} // 오목판을 구현하는 클래스의 끝입니다.

public class CLIENTextends Frame implements Runnable, ActionListener {

        // 방에 접속한 인원의 수를 보여주는 레이블입니다

        private Label pInfo = new Label(" > 대기실"); // 라벨의 이름을
표시합니다.

        private java.awt.List pList = new java.awt.List(); // 현재 같은
방(대기실)의 인원 명단을 보여주는 리스트입니다.

        private Button startButton = new Button("게임 시작"); // 게임을 시작하는
버튼입니다.

        private Button stopButton = new Button("기권"); // 기권을 하는
버튼입니다.

        private Button enterButton = new Button("게임 입장"); // 게임 입장
버튼입니다.

        private Button exitButton = new Button("대기실 입장"); // 대기실로
입장하는 버튼입니다.

        // 각종 컴포넌트입니다.

        private TextArea msgView = new TextArea("", 1, 1, 1); // 메시지를
표시해주는 영역인 TextArea 컴포넌트입니다.

        private TextField sendBox = new TextField(""); // 송신할 메시지를 적을
수 있는 영역인 TextField 컴포넌트입니다.

        private TextField nameBox = new TextField(); // 유저의 이름을 적을 수
있는 영역인 TextField 컴포넌트입니다.

        private TextField roomBox = new TextField("0"); // 방 번호을 적을 수
있는 영역인 TextField 컴포넌트입니다.

        // 여러가지 정보를 보여주는 레이블입니다.

        private Label infoView = new Label("< <  OMG  > >", 1);

        private BOARD board = new BOARD(15, 30); // 오목판 객체를 생성합니다.

        private BufferedReader br; // 소켓에게서 받아올 때 쓰이는 입력
스트림입니다.

        private PrintWriter pw; // 소켓으로 내보내줄 때 쓰이는 출력 스트림입니다.

        private Socket socket; // 소켓

        private int roomNumber = -1; // 방 번호입니다.

        private String userName = null; // 유저 이름입니다.

        private Font f1 = new Font("NanumGothic", Font.BOLD, 20); // 나눔고딕,
굵게, 20포인트의 폰트입니다.
```

```
        private Font f2 = new Font("NanumGothic", Font.BOLD, 13); // 나눔고딕, 굵게, 13포인트의 폰트입니다.

        private Font f3 = new Font("NanumGothic", Font.BOLD, 11); // 나눔고딕, 굵게, 11포인트의 폰트입니다.

        public CLIENT(String title) { // 생성자를 만듭니다.

        super(title);

        setLayout(null); // 레이아웃을 사용하지 않고 배치하기 위해 쓰입니다.

        // 각종 컴포넌트를 생성하고 직접 배치합니다.

        this.setBackground(Color.BLACK); // 배경을 검은색으로 지정합니다.

        msgView.setEditable(false); // msgView TextArea를 수정을 할 수 없도록 잠급니다.

        infoView.setBounds(10, 30, 480, 30); // infoView의 크기와 배치되는 위치를 설정합니다.

        infoView.setBackground(new Color(252, 247, 142)); // 맨 위 텍스트바 색상을 변경합니다.

        infoView.setFont(f1); // 폰트를 f1으로 지정합니다.

        board.setLocation(10, 70); // board의 위치를 설정합니다.

        add(infoView); // infoView 컴포넌트를 배치합니다.

        add(board); // board 컴포넌트를 배치합니다.

        Panel p = new Panel(); // 새로은 패널 p를 생성합니다.

        p.setBackground(new Color(252, 247, 142)); // 패널의 색상을 변경합니다.

        p.setLayout(new GridLayout(3, 3)); // 패널의 레이아웃을 설정합니다.

        p.add(new Label("자신의 닉네임   :   ", 2)); // 패널에 레이블을 하나 생성합니다.
        p.add(nameBox); // 패널에 namebox 컴포넌트를 하나 생성합니다.

        p.setFont(f2); // 패널의 폰트를 f2로 변경합니다.

        p.add(new Label("참가할 방 번호   :   ", 2)); // 패널에 레이블을 하나 생성합니다.
        p.add(roomBox); // roomBox 컴포넌트를 생성합니다.

        p.add(enterButton); // enterButton 컴포넌트를 생성합니다.
        p.add(exitButton); // exitButton 컴포넌트를 생성합니다.

        enterButton.setEnabled(false); // enterButton을 비활성화 시킵니다.

        p.setBounds(500, 30, 290, 70); // 패널 p의 위치와 크기를 설정합니다.

        Panel p2 = new Panel(); // 패널 p2를 생성합니다.

        p2.setBackground((new Color(255, 247, 142))); // 패널 p의 배경색을 설정합니다.

        p2.setLayout(new BorderLayout()); // 패널 p의 레이아웃을 설정합니다.

        Panel p2_1 = new Panel(); // 패널 p2_1를 생성합니다.
```

```
        p2_1.add(startButton); // 패널 p2_1에 startButton을 생성합니다.
        p2_1.add(stopButton); // 위와 마찬가지로 stopButton을 생성합니다.

        startButton.setBackground(new Color(184, 255, 176)); // startButton의
배경색을 설정합니다.
        stopButton.setBackground(new Color(184, 255, 176));// stopButton의
배경색을 설정합니다.
        startButton.setForeground(Color.white);// startButton의 색을
설정합니다.
        stopButton.setForeground(Color.WHITE); // stopButton의 색을 설정합니다.

        enterButton.setBackground(new Color(184, 255, 176)); // EnterButton의
색을 설정합니다.
        exitButton.setBackground(new Color(184, 255, 176));// exitButton의 색을
설정합니다.

        p2.add(pInfo, "North"); // pInfo 컴포넌트를 북쪽(위)에 배치합니다.
        p2.add(pList, "Center"); // pList 컴포넌틀를 중앙에 배치합니다.
        p2.add(p2_1, "South"); // p2_1 패널을 남쪽(아래)에 배치합니다.

        p2.setFont(f2); // p2의 폰트를 f2로 지정합니다.

        startButton.setEnabled(false); // 버튼을 비활성화합니다.
        stopButton.setEnabled(false); // 버튼을 비활성화합니다.

        p2.setBounds(500, 110, 290, 125); // 대기실 Label 크기 설정

        msgView.setBackground(new Color(255, 247, 147));

        Panel p3 = new Panel();

        p3.setLayout(new BorderLayout());

        p3.add(msgView, "Center");

        p3.add(sendBox, "South");

        p3.setBounds(500, 250, 290, 300); // 채팅창 TextArea의 크기를 설정합니다.

        p3.setFont(f3);

        add(p);
        add(p2);
        add(p3);

        // 각종 컴포넌트들의 이벤트 리스너를 등록한다.

        sendBox.addActionListener(this);

        enterButton.addActionListener(this);

        exitButton.addActionListener(this);

        startButton.addActionListener(this);

        stopButton.addActionListener(this);

        // 윈도우 닫기를 처리합니다.

        addWindowListener(new WindowAdapter() {

        public void windowClosing(WindowEvent we) {

        System.exit(0);
```

```
        }});

        }

        // 컴포넌트들의 액션 이벤트들을 처리하는 부분입니다.

        public void actionPerformed(ActionEvent ae) {

        if (ae.getSource() == sendBox) { // 만약 이벤트가 일어나는 곳이 메시지
입력 상자이면

        String msg = sendBox.getText(); // senBox에 적인 문자를 msg에 저장합니다.

        if (msg.length() == 0) // msg의 길이가 0이라면 빠져나옵니다.
        return;

        if (msg.length() >= 30) // msg의 길이가 30보다 크다면 msg를 30글자 까지
자릅니다.
        msg = msg.substring(0, 30);

        try {

        pw.println("[MSG]" + msg); // 자른 문자에 [MSG]라는 문자들을 붙여서
소켓으로 보내줍니다.

        sendBox.setText(""); // 메시지 입력 상자를 비웁니다.

        } catch (Exception ie) {
        }

        }

        else if (ae.getSource() == enterButton) { // 만약 이벤트가 일어나는 곳이
입장하기 버튼이면

        try {

        if (Integer.parseInt(roomBox.getText()) < 1) { // roomBox에 적힌 글자가
1보다 작다면

        infoView.setText("방번호가 잘못되었습니다. 1이상"); // infoView에 해당
문자들을 출력하고 빠져나옵니다.

        return;

        }

        pw.println("[ROOM]" + Integer.parseInt(roomBox.getText())); // 앞에
[ROOM]을 붙이고 유저가 적은 문자를 정수로 변환해 같이 묶어서
                // 소켓으로 전송합니다.

        msgView.setText(""); // 텍스트 상자를 비웁니다.

        } catch (Exception ie) {

        infoView.setText("...입력에 오류가 존재합니다..."); // 오류가 있으면 해당
문자를 출력합니다.

        }

        }

        else if (ae.getSource() == exitButton) { // 만약 이벤트가 일어나는 곳이
대기실로 버튼이면
```

```
        try {
        goToWaitRoom(); // goToWaitRoom 이라는 메소드를 실행합니다.
        startButton.setEnabled(false); // startButton과 stopButton을 비활성화
합니다.
        stopButton.setEnabled(false);
        } catch (Exception e) {
        }
        }
        else if (ae.getSource() == startButton) { // 만약 이벤트가 일어나는 곳이
게임 시작 버튼이면
        try {
        pw.println("[START]"); // [START]라는 문자열을 소켓으로 보냅니다.
        infoView.setText("상대의 결정을 기다립니다."); // 해당 문자를 출력합니다.
        startButton.setEnabled(false); // startButton을 비활성화합니다.
        } catch (Exception e) {
        }
        }
        else if (ae.getSource() == stopButton) { // 만약 이벤트가 일어나는 곳이
기권 버튼이면
        try {
        pw.println("[DROPGAME]"); // [DROPGAME]이라는 문자열을 내보냅니다.
        endGame("당신은 게임을 기권하였습니다. 당신은 패배하였습니다..."); //
endGame 메소드를 호출합니다.
        } catch (Exception e) {
        }
        }
        }
        void goToWaitRoom() { // 대기실로 버튼을 누르면 호출되는 메소드입니다.
        if (userName == null) { // userName이 초기값(NULL)이라면
        String name = nameBox.getText().trim(); // nameBox에서
화이트스페이스(스페이스, 탭, 엔터 같은 공백)을 제거한 문자열을 name에 대입합니다.
        if (name.length() <= 2 || name.length() > 10) { // name의 길이가 2글자
이하이거나 10글자를 넘는다면
        infoView.setText("이름은 3글자 이상, 9글자 이하여야 합니다!"); // 해당
문자열을 출력합니다.
        nameBox.requestFocus(); // 키 입력을 받을 컴포넌트를 강제로 설정합니다. 즉
nameBox를 입력받도록 강제합니다.
        return;
```

```
        }

        userName = name; // userName을 name으로 지정합니다.

        pw.println("[NAME]" + userName); // 앞에 [NAME]을 추가한 문자열을
소켓에게 전송합니다.

        nameBox.setText(userName); // nameBox에 유저의 이름을 띄웁니다.

        nameBox.setEditable(false); // nameBox를 수정할 수 없도록 막습니다.

        }

        msgView.setText(""); // msgView를 공백으로 바꿉니다.

        pw.println("[ROOM]0"); // [ROOM]0를 소켓으로 보냅니다.

        infoView.setText("대기실에 입장하셨습니다. 즐거운 하루 되세요!"); // 해당
메세지를 infoView에 뛰웁니다.

        roomBox.setText("0"); // roomBox의 문자를 0으로 바꿉니다.

        enterButton.setEnabled(true); // enterButton을 활성화합니다.

        exitButton.setEnabled(false); // exitButton을 비활성화합니다.

        }

        public void run() {

        String msg; // 서버로부터의 메시지를 대입합니다.

        try {

        while ((msg = br.readLine()) != null) { // msg에 소켓으로부터 수신한 값을
대입하며 msg가 null이 아닌 동안 반복합니다.

        if (msg.startsWith("[STONE]")) { // 받아온 msg의 문자의 시작이
[STONE]이면 상대편이 놓은 돌의 좌표에 대한 값임을 의미합니다.

        String temp = msg.substring(7); // 앞의 7글자 [STONE]를 제거한 값을
temp에 대입합니다. 그려면 문자열 temp는 "A B"만 남습니다.      (A, B는 숫자입니다.)

        int x = Integer.parseInt(temp.substring(0, temp.indexOf(" "))); //
x값에 temp의 처음시작부터 " "(스페이스)전까지의
                // 문자들을 정수로 변환하여 대입합니다. 즉 정수 A를
                // 대입받습니다.

        int y = Integer.parseInt(temp.substring(temp.indexOf(" ") + 1)); //
y값에 temp의 " "(스페이스)가 있는 지점 +
                // 1만큼의 위치 아래로 전부 버린 후 그 값을 정수화
                // 하여 대입합니다 즉 정수 B를 대입합니다.

        board.putOpponent(x, y); // 위에서 받은 x, y값을 상대방이 돌을 놓는
메소드의 매개변수로 대입하여 호출합니다.

        board.setEnable(true); // 유저가 돌을 놓을 수 있도록 합니다.

        }

        else if (msg.startsWith("[ROOM]")) { // 받아온 msg의 문자의 시작이
[ROOM]이면 방 입장에 대한 값임을 의미합니다.

        if (!msg.equals("[ROOM]0")) { // 대기실(0)이 아닌 방이라면
```

```
        enterButton.setEnabled(false); // enterButton을 비활성화하고

        exitButton.setEnabled(true); // exitButton을 활성화합니다.

        infoView.setText(msg.substring(6) + "번 방에 입장하셨습니다."); // 받아온
문자열의 앞 6글자 [ROOM]을 제거한 값에 문자열을 더해 출력합니다. ex)"3번 방에
입장하셨습니다."

        }

        else
        infoView.setText("대기실에 입장하셨습니다."); // 아니면 infoView에 해당
문자를 출력합니다.

        roomNumber = Integer.parseInt(msg.substring(6)); // roomNumber에 msg의
앞 6글자 [ROOM]를 제거한 값을 정수로 형변환하여 대입합니다.

        if (board.isRunning()) { // 게임이 진행중인지를 boolean형으로 반환해주는
메소드를 실행합니다. 진행중이면 ture / 아니라면 false입니다.

        board.stopGame(); // 게임을 중지시킵니다.

        }

        }
        else if (msg.startsWith("[FULL]")) { // 받아온 msg의 문자의 시작이
[FULL]이라면 방이 찬 상태를 의미하는 값입니다.

        infoView.setText("방이 차서 입장할 수 없습니다."); // infoView에 해당
문자를 출력합니다.

        }

        else if (msg.startsWith("[PLAYERS]")) { // 받아온 msg의 문자의 시작이
[PLAYERS]라면 방에 있는 유저들의 명단을 의미합니다.

        nameList(msg.substring(9)); // msg의 앞 9글자 [PLAYERS]를 제거한 값을
nameList에 추가하는 메소드를 호출합니다.

        }

        else if (msg.startsWith("[ENTER]")) { // 받아온 msg의 문자의 시작이
[ENTER]이라면 방 입장에 대한 값입니다.

        pList.add(msg.substring(7)); // msg의 앞 7글자 [ENTER]를 제거한 값을
pList에 추가합니다.

        playersInfo(); // 플레이어들에 대한 정보를 계산 후 보여줍니다.

        msgView.append("[" + msg.substring(7) + "]님이 입장하였습니다.\n"); //
msgView(채팅창)에 해당 문자들을 밑으로 추가합니다.

        }

        else if (msg.startsWith("[EXIT]")) { // 받아온 msg의 문자의 시작이
[EXIT]라면 유저의 퇴장에 대한 값입니다.

        pList.remove(msg.substring(6)); // pList에서 msg의 앞 6글자 [EXIT]를
제거한 값에 대한 리스트를 제거합니다.

        playersInfo(); // 인원수를 다시 계산하여 보여줍니다.

        msgView.append("[" + msg.substring(6) +

        "]님이 다른 방으로 입장하였습니다.\n"); // 해당 문자를 msgView(채팅창)에
```

밑으로 추가합니다.

```
        if (roomNumber != 0) // 만약 대기실이 아닐때라면

        endGame("상대방이 게임에서 나갔습니다."); // endGame메소드를 호출합니다.

        }

        else if (msg.startsWith("[DISCONNECT]")) { // 받아온 msg의 문자의 시작이
[DISCONNECT]라면 유저의 연결 종료에 대한 값입니다.

        pList.remove(msg.substring(12)); // msg의 앞에서 12글자 [DISCONNECT]를
제거한 값에 대한 pList의 값을 제거합니다.

        playersInfo(); // 인원수를 계산 후 보여줍니다.

        msgView.append("[" + msg.substring(12) + "]님이 접속을 끊었습니다.\n");
// 채팅창에 해당 문자를 출력합니다.

        if (roomNumber != 0) // 만약 대기실이 아니라면

        endGame("상대가 나갔습니다."); // endGame메소드를 호출합니다.

        }

        else if (msg.startsWith("[COLOR]")) { // 받아온 msg의 문자의 시작이
[COLOR]라면 돌의 색을 부여하는것에 대한 값입니다.

        String color = msg.substring(7); // msg의 앞에 7글자 [COLOR]를 제거한
값을 color 문자열에 대입합니다.

        board.startGame(color); // 게임을 시작하는 메소드에 매개변수로 color을
대입하여 실행합니다.

        if (color.equals("BLACK")) // 만약 color문자열이 "BLACK"과 같다면

        infoView.setText("흑돌로 시작합니다.."); // 해당 문자를 출력합니다.

        else

        infoView.setText("백돌로 시작합니다.."); // 아니라면 해당문자를
출력합니다.

        stopButton.setEnabled(true); // 기권 버튼을 활성화합니다.

        }

        else if (msg.startsWith("[DROPGAME]")) // 받아온 msg의 문자의 시작이
[DROPGAME]이라면 상대가 기권한다는 것에 대한 값 입니다.

        endGame("상대가 기권하였습니다."); // endGame을 출력합니다.

        else if (msg.startsWith("[WIN]")) // 받아온 msg의 문자의 시작이
[WIN]이라면 승리에 대한 값입니다.

        endGame("당신이 이겼습니다. 축하드립니다!");

        else if (msg.startsWith("[LOSE]")) // 받아온 msg의 문자의 시작이
[LOSE]이라면 승리에 대한 값입니다.

        endGame("당신이 졌습니다. 힘내세요!"); // 약속된 메시지가 아니면 메시지
영역에 보여준다.

        else
        msgView.append(msg + "\n"); // 위에 있는 어떠한 경우도 아니라면 채팅창에
```

그대로 띄웁니다.

```
        }
        } catch (IOException ie) {
        msgView.append(ie + "\n"); // 에러가 있으면 채팅창에 띄웁니다.
        }
        msgView.append("접속이 끊겼습니다.");
        }
        private void endGame(String msg) { // 게임을 종료시키는 메소드입니다.
        infoView.setText(msg); // infoView에 받아온 msg를 출력합니다.
        startButton.setEnabled(false); // startButton을 비활성화합니다.
        stopButton.setEnabled(false); // stopButton을 비활성화합니다.
        try {
        Thread.sleep(2000); // 2초동안 스레드를 잠재웁니다.
        } catch (Exception e) {
        }
        if (board.isRunning()) // 만약 게임이 진행중이라면, 게임을 종료시킵니다.
        board.stopGame();
        if (pList.getItemCount() == 2) // 만약 pList의 목록에 있는 값의 수가
2개이면
        startButton.setEnabled(true); // startButton을 활성화시킵니다.
        }
        private void playersInfo() { // 방 접속자의 수를 보여주는 메소드입니다.
        int count = pList.getItemCount(); // count에 pList의 값의 개수를
대입합니다.
        if (roomNumber == 0) // 방이 대기실(0)이라면
        pInfo.setText("현재 대기실 인원 : " + count + "명"); // pInfo에 해당
문자를 출력합니다.
        else
        pInfo.setText(roomNumber + " 번 방: " + count + "명"); // 아니라면
pInfo에 해당 문자를 출력합니다.
        // 게임 시작 버튼의 활성화 상태를 점검한다.
        if (count == 2 && roomNumber != 0) // count가 2이며, 방이 대기실(0)이
아닐 때,
        startButton.setEnabled(true); // 시작버튼을 활성화합니다.
        else
        startButton.setEnabled(false); // 아니라면 시작버튼을 비활성화합니다.
        }       // 유저 리스트에서 유저들을 추출하여 pList에 추가합니다.
        private void nameList(String msg) {
        pList.removeAll(); // pList에서 모든 값을 제거합니다.
```

```
        StringTokenizer st = new StringTokenizer(msg, "\t"); //
StringTokenizer란 첫번째 매개변수로 들어온 문자열(msg)을 두번째 매개변수로 들어온
                // 문자열("\t")을 구획문자로 하여 토큰 단위로 끊어주는 기능을
합니다. 예) Hello My
                // Friends를 " "를 구획문자로 하여 토큰 단위로 끊으면 Hello, My,
Friends로 나누어집니다.

        while (st.hasMoreElements()) // st가 아직 토큰 요소가 더 가지고 있다면

        pList.add(st.nextToken()); // pList에 st의 다음 토큰을 추가하는 것을
반복합니다.

        playersInfo(); // 방의 접속자 수를 보여주는 메소드를 호출합니다.

        }

        private void connect() { // 연결에 대한 메소드입니다.

        try {

        msgView.append("서버 연결을 시도합니다.\n"); // 채팅창에 출력합니다.

        socket = new Socket("192.168.0.2", 0707); // 해당하는 서버 아이피에
접속합니다.

        msgView.append("!!!!! 연결 성공 !!!!!\n"); // 채팅창에 출력합니다.

        msgView.append("닉네임을 입력 후 대기실에 입장하세요.\n"); // 채팅창에
출력합니다.

        br = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // 소켓에게서 값을 받아오는
bufferedReader를 생성합니다.

        pw = new PrintWriter(socket.getOutputStream(), true); // 소켓에게 값을
전달하는 PrintWriter을 생성합니다.

        new Thread(this).start(); // 객체 자기자신을 새로운 메소드로 생성하여
실행합니다.

        board.setWriter(pw);

        } catch (Exception e) {

        msgView.append(e + "\n\n..... 연결 실패 .....\n"); // 채팅창에
출력합니다.

        }

        }

        public static void main(String[] args) {

        CLIENT client = new CLIENT("네트워크 오목 게임");

        client.setSize(800, 560);// 클라이언트 창의 크기입니다
        client.setLocation(175, 50);// 클라이언트 창 열리는 위치입니다
        client.setVisible(true);

        client.connect();

        }

}
```