프로그램 개요

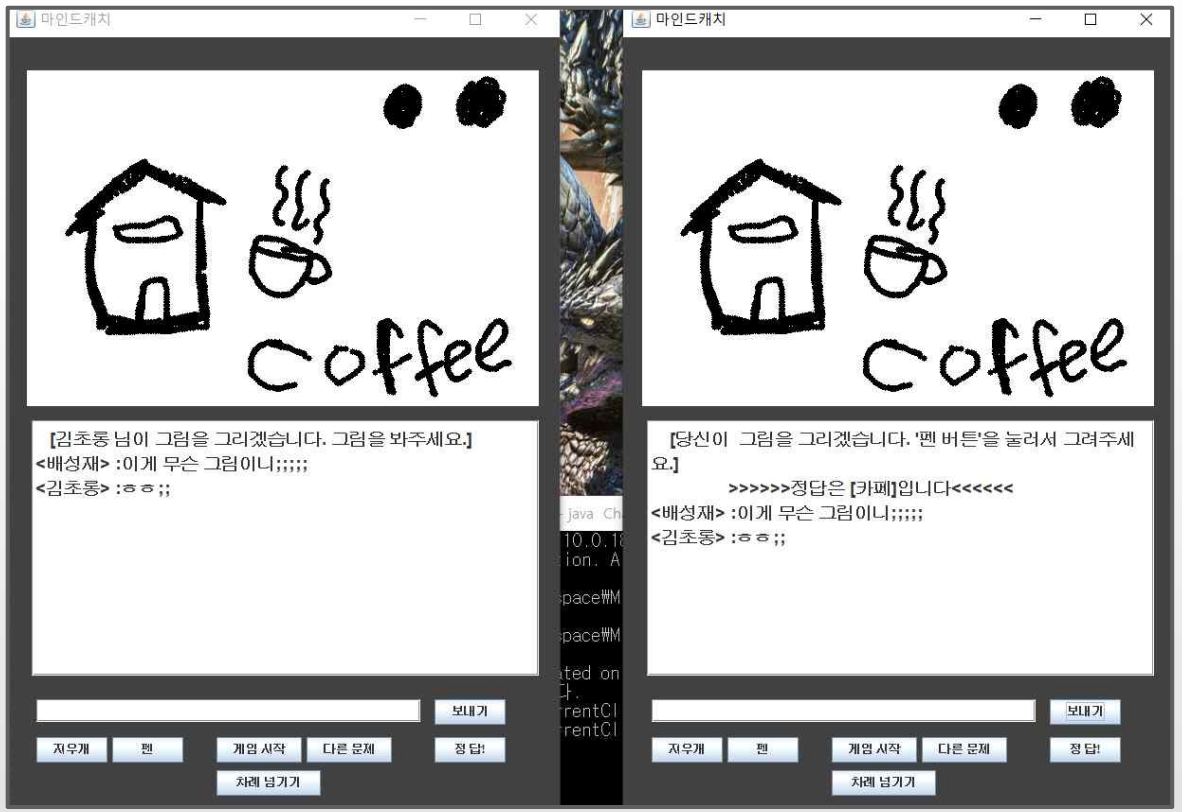
SSL키를 활용한 캐치마인드와 채팅

- 하나의 서버를 통해서 여러명의 사용자가 그림을 그리는 게임과 채팅이 가능한 프로그램.

- 자바 Swing을 이용해서 GUI를 만들었고 SSL 키를 통해 서버와 클라이언트 간에 연결이 가능하게 만들었습니다.

- RMI func을 활용해서 게임의 기능들을 넣어 두었습니다.

IDE : 이클립스
개발환경 : JDK 1.8.0_151



실행과정과 소스코드 소개

SSL키를 활용한 캐치마인드와 채팅

서버 기능을 하는 MindCatchServer.java와 클라이언트 기능을 하는 MindCatchClient.java로 나누어 두었습니다.

SSL키는 'keytool'을 이용해서 만들어 두었습니다. keytool을 통해 인증서를 만들었다면 서버와 클라이언트에게 키를 넘겨주도록하고, 키를 통해서만 연결이 가능하도록 SSLFactory 클래스를 통한 소켓 연결을 해두었습니다.

```
kStore = KeyStore.getInstance("JKS");
kStore.load(new FileInputStream(ksName), keyStorePass);

kmFactory = KeyManagerFactory.getInstance("SunX509");
kmFactory.init(kStore, keyPass);

sslContext = SSLContext.getInstance("TLS");
sslContext.init(kmFactory.getKeyManagers(), null, null);

sslFactory = sslContext.getServerSocketFactory();
sslservSock = (SSLServerSocket) sslFactory.createServerSocket(Port);

System.out.println ("Server started: socket created on " + Port);

System.out.println("마인드캐치 서버 동작했습니다.");
```

MindCatchServer.java

서버에서 해당 키를 통해 SSLFactory를 만들었고, 이 객체를 통해서 SSLServerSocekt 소켓을 만들어 같은 SSL키를 가진 클라이언트가 접속하기를 기다립니다.

서버에서는 클라언트가 들어오면 쓰레드로 생성해서 클라이언트의 요청을 기다려줍니다.

```
6명이 참가할 수 있습니다.
Server started: socket created on 8000
마인드캐치 서버 동작했습니다.
```

실행과정과 소스코드 소개

MindCatchClient.java

```
System.setProperty("javax.net.ssl.trustStore", "trustedcerts");
System.setProperty("javax.net.ssl.trustStorePassword", "123456");

sslFactory = (SSLSocketFactory) SSLSocketFactory.getDefault();
chatSocket = (SSLSocket) sslFactory.createSocket(Server, Port);

String[] supported = chatSocket.getSupportedCipherSuites();
chatSocket.setEnabledCipherSuites(supported);

RMI_func = (Game_func)Naming.lookup("rmi://"+"127.0.0.1"+"/"+"ser");

EventQueue.invokeLater(new Runnable() {
    public void run() {
        try {
            UI = new Client_UI(chatSocket, RMI_func);
            UI.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
```

```
class Client_UI extends JFrame{

    private Container   container;  // M

    private GrimPane    grimPanel;
    private JTextField  textField;
    private JTextPane   chatPanel;

    private Game_func   RMI_func;
    private SSLSocket   chatSocket;
    private StringBuilder   allText;

    private String inputText = "";
    private String MyName = "";
    private PrintWriter out = null;
    private boolean icandraw = true;
    private boolean gameRunning = false;
```

클라이언트에서는 ip주소와 해당 포트번호를 통해 서버에 들어갔다면, 그 서버와 SSLSocket통신을 위해서 SSL키와 그 비밀번호가 맞는지 확인을 하여 SSLFactory를 통해 SSLSocket을 만들어 줍니다.

SSL통신으로 접속에 성공을 했다면 클라이언트를 위한 GUI를 생성해서 보여줍니다.

실행과정과 소스코드 소개

서버와 클라이언트는 서로 메시지를 주고받으며 요청을 하게되는데 서로간의 OutputStream과 InputStream을 구성하게 됩니다.

클라이언트는 "시그널 번호 # 내용"으로 서버에게 메시지를 보내고, 서버에서는 이를 구문분석해서 필요한 기능을 클라이언트들이 사용할 수 있게 메시지를 보냅니다.

MindChatServer에서 메시지 주고받기

```
while((inputLine = in.readLine())!=null) {
    Token = new StringTokenizer(inputLine, "#");
    String Signal=null, message=null;
    if(Token.hasMoreElements())
        Signal = Token.nextToken();
    if(Token.hasMoreElements())
        message = Token.nextToken();
    if(Signal==null || message==null) continue;

    if(Signal.equalsIgnoreCase("100")) {
        if(clientName == null) {
            clientName = message;
            continue;
        }
        chatServer.putClient(getClientID(), message, 1);
        continue;
    }
    else if(Signal.equalsIgnoreCase("200")) {
        chatServer.putClient(getClientID(), message, 2);
        continue;
    }
    else if(Signal.equalsIgnoreCase("300")) {
        chatServer.putClient(getClientID(), message, 3);
        continue;
    }
    else if(Signal.equalsIgnoreCase("400")) {
```

서버에게

여러 클라이언트들에게

MindChatClient에서 메시지 주고받기

```
while((readSome = in.readLine())!=null) {
    Token = new StringTokenizer(readSome, "#");
    String temp = Token.nextToken();
    String message = Token.nextToken();

    if(temp.equalsIgnoreCase("100")) {      // 일반적 대화
        allText.append(message+"\n");
        chatPanel.setText(allText.toString());
        Token = null;
        continue;
    }
    else if(temp.equalsIgnoreCase("200")) { // 누군가 정답을 맞추었을 때
        Token = new StringTokenizer(message, "*");
        String name = Token.nextToken();
        String ans = Token.nextToken();

        allText.append("      ["+name+"님이 정답을 맞추셨습니다!]\n");
        chatPanel.setText(allText.toString());
        allText.append("     [정답은 \""+ans+"\" 입니다.]\n");
        chatPanel.setText(allText.toString());

        gameRunning = false;
        icandraw = true;
        Token = null;
        continue;
    }
    else if(temp.equalsIgnoreCase("300")) { // 누군가 오답을 냈을 때
```

실행과정과 소스코드 소개

RMI func를 이용하도록 rmiregistry를 실행하고, 이를 바탕으로 게임에 필요한 자바 메소드를 서버로부터 호출받을 수 있습니다.

```
public interface Game_func extends Remote {

    // 게임의 전반적인 내용물을 담고있습니다. 유저들이 게임에 대한 정보를 얻는데 있어서
    public int Gamer_Num() throws RemoteException;
    public int TurnPass() throws RemoteException;
    public String SetAns() throws RemoteException;
    public boolean isAns(String ans) throws RemoteException;
    public int WhoStart() throws RemoteException;
    public int Start() throws RemoteException;
}
```

```
>start rmiregistry
```

MindCatchServer

```
RMI_func = new Game_Impl(clients);
```

서버에서 Game_func 인터페이스를 상속받은 Game_Impl 클래스를 객체로 만들어 두면.

MindCatchClient

```
RMI_func = (Game_func)Naming.lookup("rmi://"+"127.0.0.1"+"/"+"ser");
```

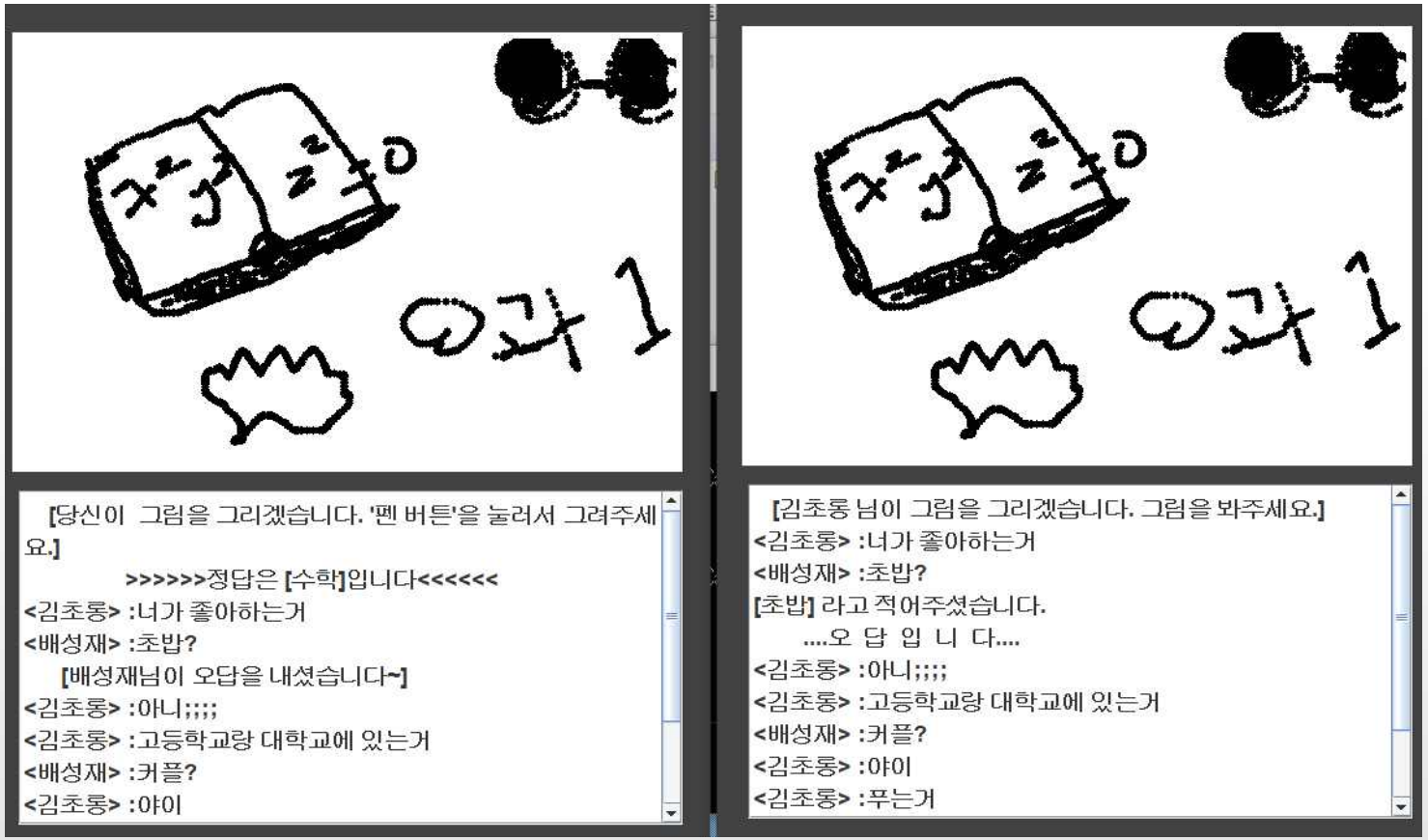
클라이언트에서 해당 객체를 찾아서 서버에 있는 메소드를 사용할 수 있게된다.

실행과정과 소스코드 소개

연결과 RMI가 준비가 되었다면 상대방과 채팅과 그림을 그리면서 게임을 할 수 있습니다.

그림을 그릴때마다 서버에서 클라이언트들에게 해당 점의 위치를 알려주어 모든 클라이언트의 그림판에 그림이 그려지게 됩니다.

마인드캐치 게임을 시작하면 그림을 그리는 사람에게 문제를 제시하고, 나머지 사람들은 펜을 사용하지 못하며 채팅으로만 소통하여 정답을 맞추어가면 됩니다.



실행화면 전체

```
6명이 참가할 수 있습니다.
Server started: socket created on 8000
마인드캐치 서버 동작했습니다.
Client connected: 10620, CurrentClient: 1
Client connected: 10623, CurrentClient: 2
Client connected: 10626, CurrentClient: 3
```