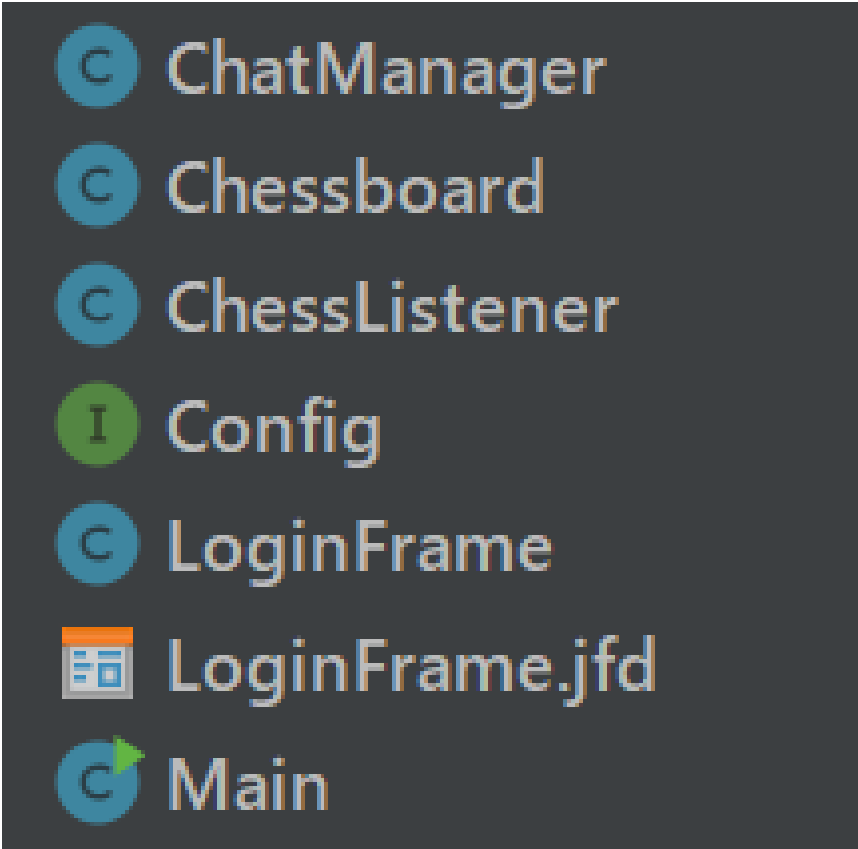


作成したプログラムの概要

- ・このゲームルームは、ユーザがオンラインのチャットラウンジに入り、ルームを選びまたはルームを作って、ほかのユーザと五目並べのゲームをプレイするプログラムである。
- ・プログラム全体はClientとServerに分け、複数のユーザがClientを実行して一つのServerに接続する方式となる。
- ・ユーザはGUIでClientを操作する。
- ・Java Swing とJava awtでGUIを実現する

作成したプログラムの概要



A dark gray rectangular box containing a list of source files for the client. Each item is preceded by a circular icon: blue with 'C' for classes, green with 'I' for interfaces, and a JFX icon for a JavaFX FXML file. The files listed are ChatManager, Chessboard, ChessListener, Config, LoginFrame, LoginFrame.jfd, and Main.


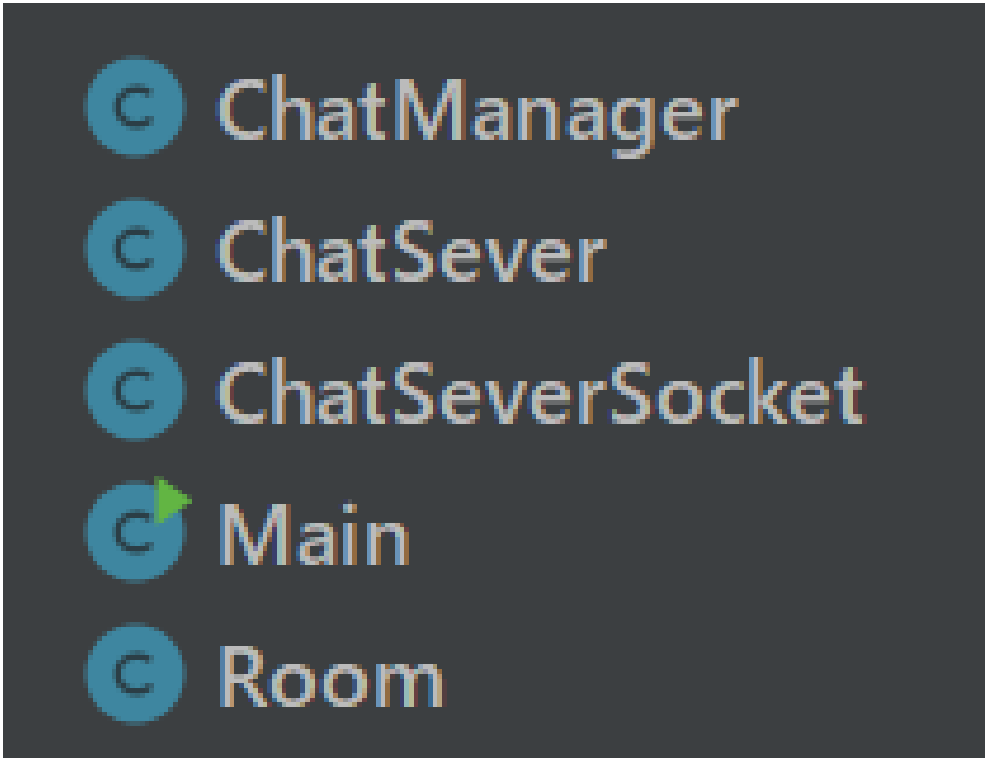
- C ChatManager
- C Chessboard
- C ChessListener
- I Config
- C LoginFrame
-  LoginFrame.jfd
- C Main

図1 Clientのソースファイル



A dark gray rectangular box containing a list of source files for the server. Each item is preceded by a circular icon: blue with 'C' for classes, and a green arrow for the main class. The files listed are ChatManager, ChatSever, ChatSeverSocket, Main, and Room.

- C ChatManager
- C ChatSever
- C ChatSeverSocket
- C → Main
- C Room

図2 Serverのソースファイル

作成したプログラムの概要

Serverのクラスの紹介

Main:

CharServerのインスタンスのスレッドをスタートする。

ターミナルからメッセージを読み込み、すべてのクライアントに送信する。

ChatSever:

クライアント側からのSocket接続を待ち、接続を確認したらChatServerSocketのインスタンスを作って、そのスレッドをスタートし、CharManagerの中のvectorにaddする。

ChatServerSocket:

サーバーのSocketのクラスである。データを受送信でき、受信した命令を処理する。

ChatManager:

サーバーのSocketを管理するクラス。接続したクライアントに指向するSocketを管理するvectorとユーザが作ったRoomのvectorを持つ。また、ChatServerSocketが受信した命令を処理するいろんなメソッドを持つ。

Room:

ユーザが作った部屋の本体。中には部屋に入ったユーザへのsocketを管理するvectorを持つ。また、メッセージを送信するメソッドsendをもつ。

作成したプログラムの概要

Clientのクラスの紹介

Main:

GUIのウィンドウのインスタンスをスタートし、そのインスタンスをChatManagerに渡す。

ChatManager:

クライアントのSocketを管理して受送信をコントロールし、GUIのウィンドウの動きと関連する。

LoginFrame:

GUIのウィンドウのソースファイルである。ChatMangerの動きを関連する。

四つの画面がり、ユーザの操作によって切替できる。

Chessboard:

五目並べゲームの盤面の本体である。ユーザのアクションを受け取って処理するクラスChessListenerのインスタンスをもつ。

ChessListener:

ユーザのアクションを受け取って、コマをセットし、セットしたコマの情報を相手に送り、どちらが勝ったのかを判断する機能を持つ。

Config (インターフェース):

Chessboardが盤面を描画する際に使う初期値を設定できる。

作成したプログラムの概要

ソケットの間のインタラクション

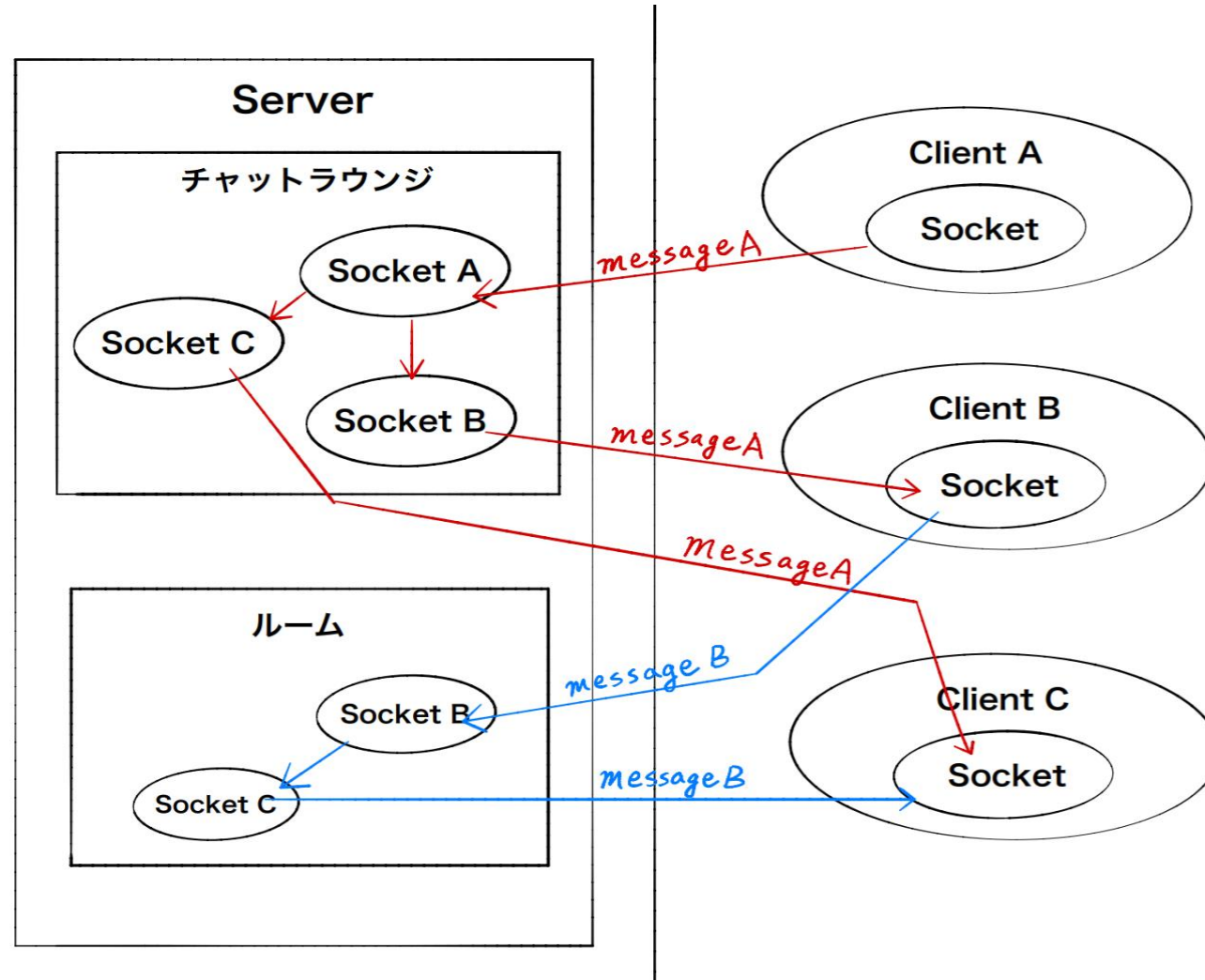


図3 ソケットの間のインタラクションの紹介

プログラムの動作の説明

The screenshot shows a login window with the following fields and annotations:

- IP:** A text box containing "12345". A red arrow points to it from the text: "ユーザのIPアドレス。プログラムの中に使われていないため12345と入力しておけばいい".
- Server IP:** A text box containing "localhost". A red arrow points to it from the text: "ServerのIPアドレス
Server側のPORT番号は
10086とプリセットした".
- Username:** A text box containing "Linetia". A red arrow points to it from the text: "ユーザの名前を設定する".
- Login:** A blue button with the text "Login".
- Footer:** The text "JFormDesigner Evaluation" is visible at the bottom of the window.

図4 Login画面の紹介

プログラムの動作の説明

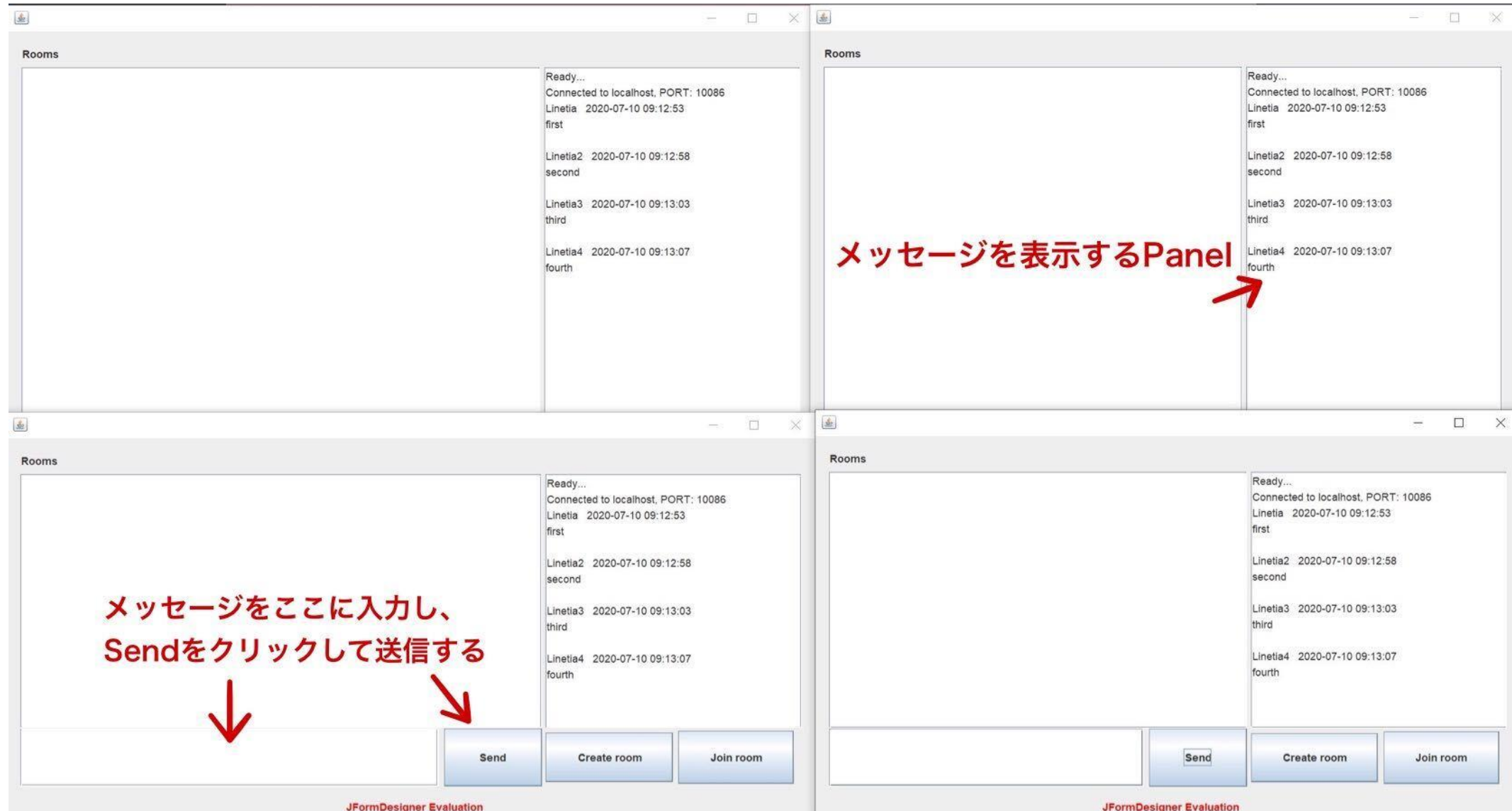


図5 チャットラウンジ画面の紹介

プログラムの動作の説明

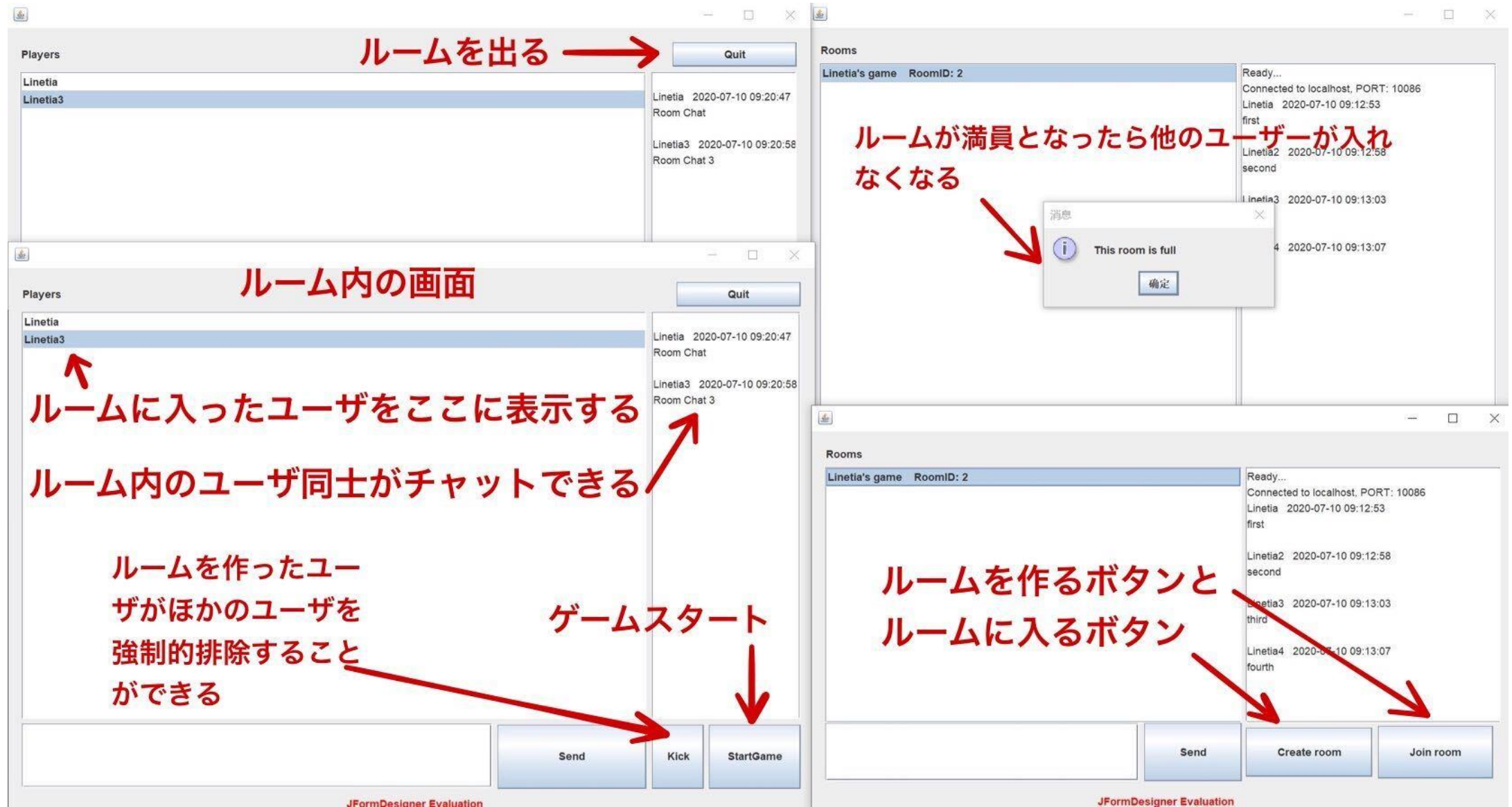


図6 ルーム画面の紹介

プログラムの動作の説明

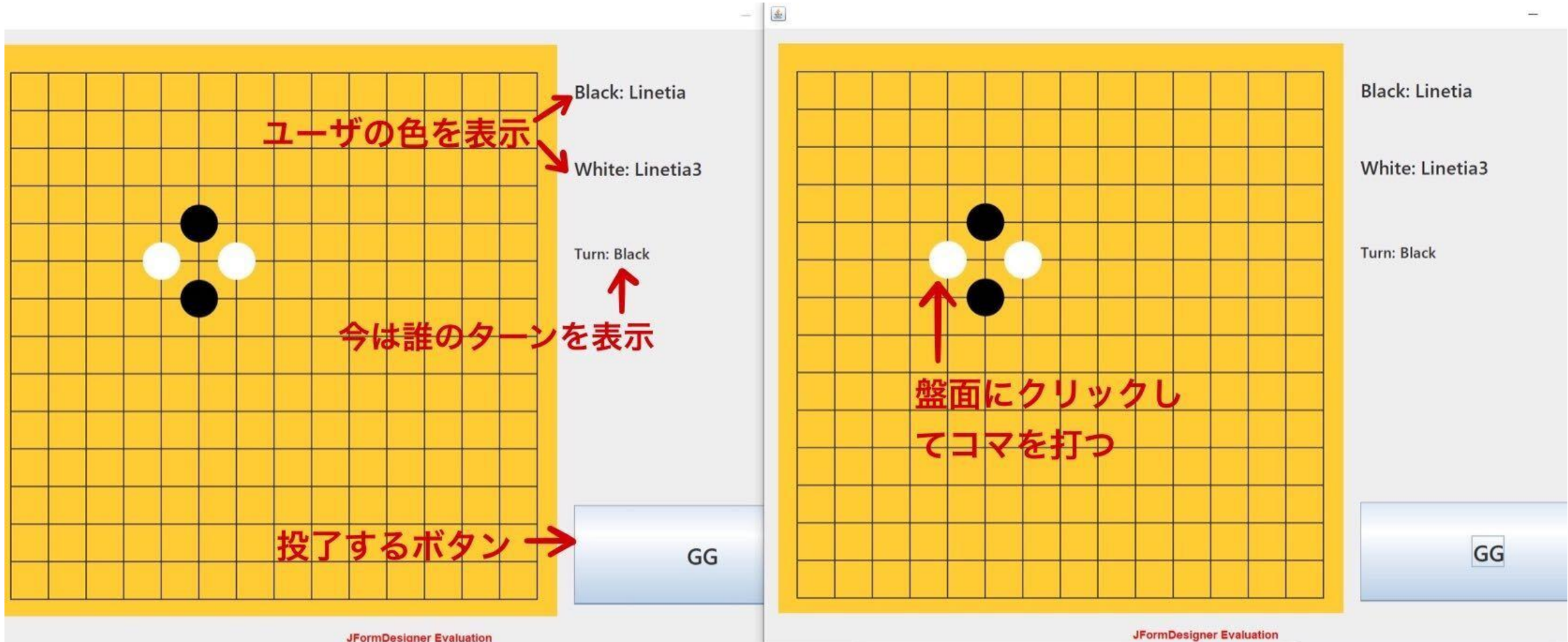


図7 ゲーム画面の紹介

プログラムの動作の説明

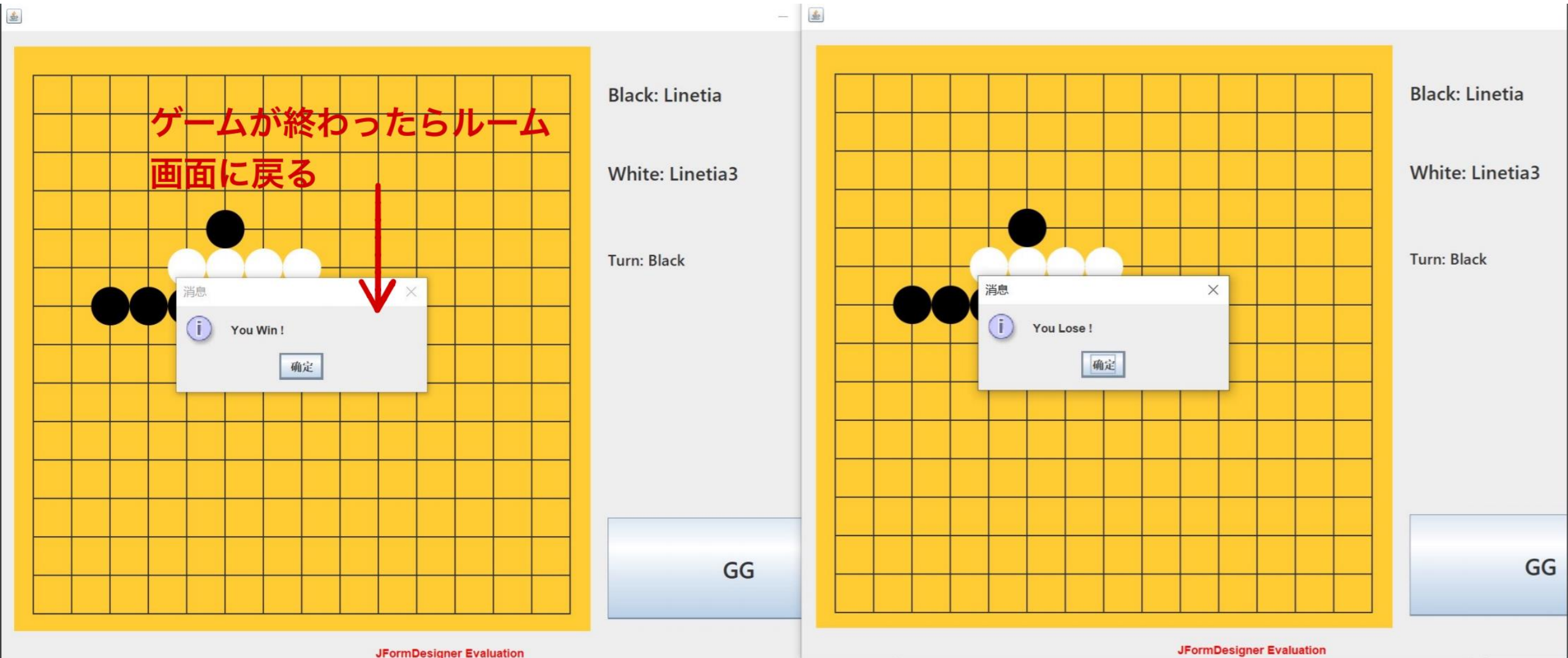


図8 ゲームが終わる画面の紹介

工夫したところ

命令の受送信

オンラインゲームルームのそれぞれの機能を実現するために、ソケット同士が受送信する十種類の命令を定義した。

例えばラウンジにはユーザA、B、Cがいると仮定する。Aがルームを作るボタンを押したら、Aのソケットが/createroomをサーバーに送信する。サーバー側のソケットが/createroomを受信したら、ChatManagerのメソッドcreateroom()を実行し、ルームを実際に作って、ラウンジにいるA、B、Cに/createroom1#A0を送信する。A、B、Cのソケットがそれを受信したら、1#A0からルームの情報を切り取って、GUI上にルームを表示させる。

工夫したところ

GUI画面の編集

GUI画面を作るために、IntelliJ IDEA上のplugin、JFormDesignerを使った。

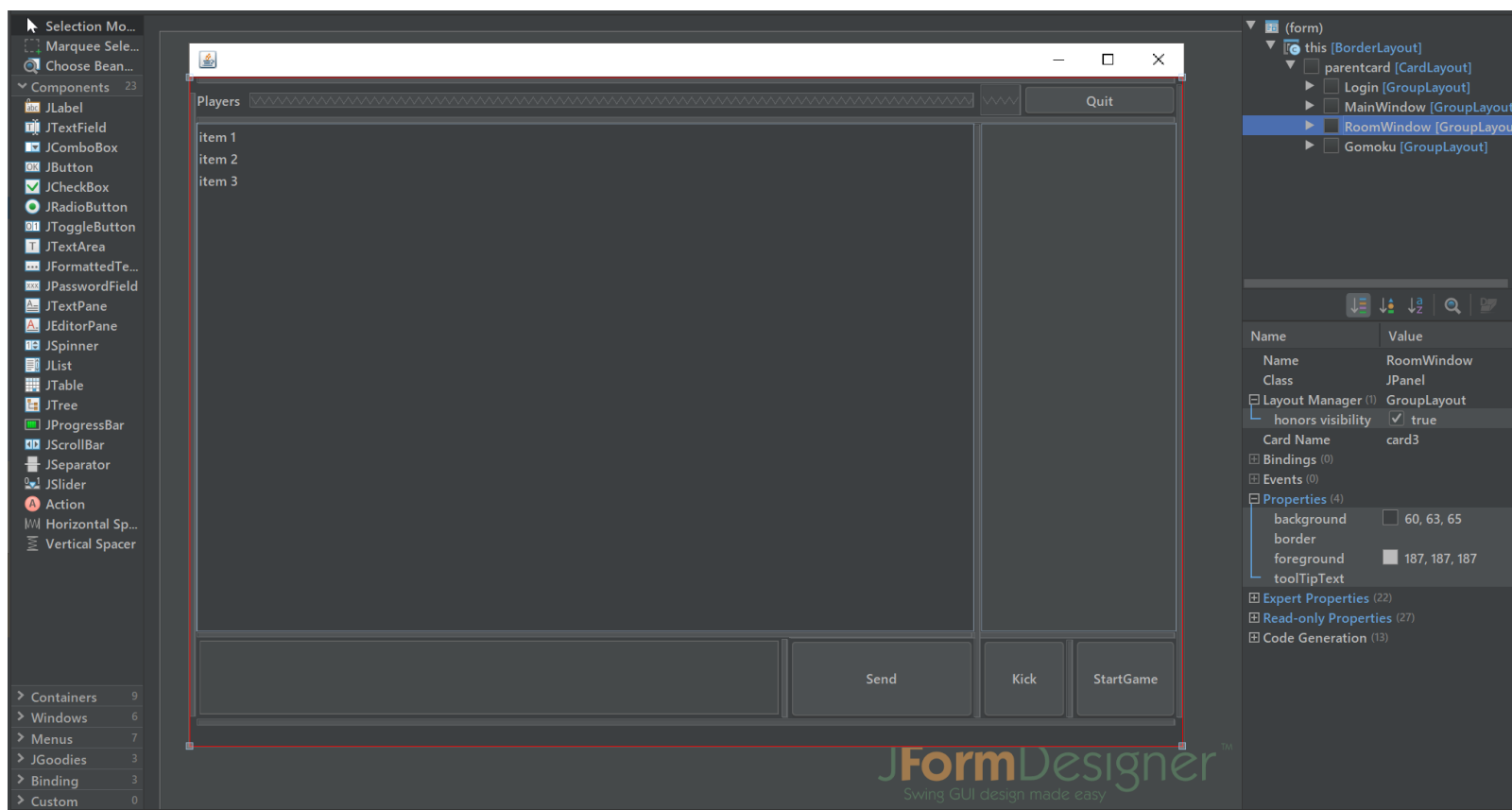


図9 JFormDesignerによるGUI画面の編集

工夫したところ

オンライン五目並べゲームの実現

Java awtのクラスGraphics用いて盤面を描画し、メソッドmouseclicked()でユーザがクリックした座標を読み取り、適切な位置であったら、その座標の情報を相手に送信して、自分のと相手の画面上のその位置にクラスGraphics2Dでコマを描画させる。例としてmouseclickedの定義を以下に示す。

```
// マウスを左クリックすると
public void mouseClicked(MouseEvent e) {
    // どこにclickしたかのを読み取る
    x = e.getX();
    y = e.getY();

    for(int r=0;r<Config.ROWS;r++)
    {
        for(int c=0;c<Config.COLUMNS;c++)
        {
            if(koma[r][c] == 0) // その位置にはコマがあるかどうかをチェックする
            {
                // 適切な位置にclickしたかのをチェックする
                if((Math.abs(x-Config.X0-c*Config.SIZE) < Config.SIZE/3.0)
                    && (Math.abs(y-Config.Y0-r*Config.SIZE) < Config.SIZE/3.0))
                {
                    // 自分の番かどうかをチェックする
                    if(myTurn != null)
                    {
                        if(whoTurn[0].equals(myTurn))
                        {
                            ChatManager.getCM().setkoma(x, y); // 相手にsetkomaの命令を送る
                            setkoma(x, y, r, c);
                            changeTurn();
                            ChatManager.getCM().changeWindowTurn(whoTurn[0]);
                            return;
                        }
                    }
                }
            }
        }
    }
}
```

図10 メソッドmouseclickedの定義