Otional Project Omok Game by 2016025514 서현아

1. Expalnation

By using TCP socket programming, which is included in the first project, this project serves omok game.

Client and Server are connected through a TCP socket.

When Server and Client are connected successfully, the omok game is played.

Client can choose which color he will play. Then automatically server's color is choosen.

In turn, each palyer puts baduk stone, and each palyer's baduk stone appears on each other's screen through socket communication.

If five stones of the same color is placed in a row, like the rules of the game we are familiar with, the game ends, and the winner and loser are determined.

The loser can choose the color of the stone, after which the game will resume.

2. Source Code

1. Client의 오목판 class : omok boad screen for client

```
package Omok_Client;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.PrintWriter;

public class Board extends Canvas {
    public static final int BLACK = 1, WHITE = -1;
    public static final int size = 30; //size of one room
    public static final int num = 20; //maximum number of stones can be put
    public static final int x = 30; //start position of x
    public static final int y = 30; //start position of y
    public static final int width = 30; //size of a stone - width
    public static final int height = 30; //size of a stone - height

    private int color; //stone color of client
    private int server_color; //stone color of server
    private String info = "choose color of stone : "; //print string
    private String str_color; //string to print stone color of client
    private boolean enable = false; //activation info of board
```

```
23      private PrintWriter writer; //stream to deliver message to server
24      private int Stone[][] = new int [num][num]; // store stone positions
25
26      public Board(){
27          this.setVisible(true);
28          this.setBackground(new Color(200,200,100));
29          //when client clicks
30          addMouseListener(new MouseAdapter() {
31              public void mousePressed(MouseEvent e) {
32                  //when the board is activated
33                  if(!enable)
34                      return;
35                  //out of board boundary : right & left
36                  else if(e.getX() > x+size*(num-1) || e.getY() > y+size*
    (num-1))
37                      return;
38                  //out of board boundary : up & down
39                  else if(e.getX() < (x-size/2) || e.getY() < (y-size/2))
40                      return;
41                  //already a stone exists
42                  else if(Stone[(e.getX()-x+size/2)/size][(e.getY()-
    y+size/2)/size] != 0)
43                      return;
44                  //put stone
45                  else
46                      Stone[(e.getX()-x+size/2)/size][(e.getY()-
    y+size/2)/size] = color;
47
48                  //send position of client's stone to server
49                  writer.println("[STONE]" + (e.getX()-x+size/2)/size + ","
    + (e.getY() - y + size/2)/size);
50                  info = "client is putting stone.";
51                  repaint();
52
53                  enable = false;
54              }
55          });
56      }
57
58      //initialize board when the game ends
59      void reset(){
60          for(int i = 0; i<num; i++){
61              for(int j = 0; j<num; j++)
62                  Stone[i][j] = 0;
63          }
64
65          //repaint = call paint method
66          repaint();
67          setEnable(false);
```

```
68      }
69
70      public void paint(Graphics g){
71          g.clearRect(0, 0, getWidth(), getHeight());
72
73          g.setColor(Color.RED);
74          //print Info
75          g.drawString(info, 30, 20);
76
77          //draw line
78          for(int i = 0; i<num; i++){
79              //choose color of line
80              g.setColor(Color.BLACK);
81              //horizontal line
82              g.drawLine(x, y + size*i, x + size*(num-1), y + size*i);
83              //vertical line
84              g.drawLine(x + size*i, y, x + size*i, y + size*(num-1));
85          }
86
87          //put stone
88          for(int i = 0; i <num; i++){
89              for(int j = 0; j<num; j++){
90                  //black stone
91                  if(Stone[i][j] == BLACK){
92                      g.setColor(Color.BLACK);
93                      g.fillOval((x-size/2) + i*size, (y-size/2) + j*size,
    width, height);
94                  }
95                  //white stone
96                  else if(Stone[i][j] == WHITE){
97                      g.setColor(Color.WHITE);
98                      g.fillOval((x-size/2) + i*size, (y-size/2) + j*size,
    width, height);
99                  }
100             }
101         }
102     }
103
104     //message for choosing stone color
105     public void stoneSelect(){
106         //choose black
107         if(JOptionPane.showOptionDialog(this, "Black Or White?", "Choose
    color of stone", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
    null, new String[]{"Black", "White"}, "Black") ==0){
108             setColor(BLACK, WHITE);
109             writer.println("[COLOR]" + WHITE + "," + BLACK);
110         }
111         //choose white
112         else{
```

```
            setColor(WHITE, BLACK);
            writer.println("[COLOR]" + BLACK + "," + WHITE);
        }
    }

    //change board's activation info
    public void setEnable(boolean enable){
        this.enable = enable;
    }

    //set stone's color
    public void setColor(int color, int server_color){
        //set client's stone color
        this.color= color;
        //set server's stone color
        this.server_color = server_color;

        //black stone : first attack
        if(color == BLACK){
            //activate board
            setEnable(true);
            info = "Stone Color : Black - first attack";
            str_color = "Black";
        }
        //white stone : second attack
        else{
            //don't activate board
            setEnable(false);
            info = "Stone Color : White - second attack";
            str_color = "White";
        }

        repaint();
    }

    //change Info to print out
    public void setInfo(String info){
        this.info = info;
    }

    //put server's stone
    public void putServer(int x, int y){
        Stone[x][y] = server_color;
        info = "My turn (" + str_color + ") - Put you stone";
        repaint();
    }

    //manage connection with server
    public void setWriter(PrintWriter writer){
```

```
        this.writer = writer;
    }

}
```

2. Client class : creates object of board. Marks client's playing and server's stone color and playing. Accepts server's game conclusion message and do proper actions by the message.

```
package Omok_Client;

import javax.swing.*;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

public class Client extends Frame {

    //object of omok game board
    private Board board = new Board();

    Socket socket = null;
    //input stream
    private BufferedReader reader;
    //output stream
    private PrintWriter writer;

    //constructor
    public Client(String name){
        super(name);
        add(board);

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent w){
                System.exit(0);
            }
        });
    }

    //connection
    private void connect() {
        try {
            socket = new Socket("127.0.0.1", 0516);
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);
```

```
41
42              board.setWriter(writer);
43              board.stoneSelect();
44
45              String msg;
46
47              while ((msg = reader.readLine()) != null) {
48                  //when server puts a stone
49                  if (msg.startsWith("[STONE]")) {
50                      msg = msg.substring(7);
51                      //now it's client's turn
52                      board.setEnable(true);
53                      board.putServer(Integer.parseInt(msg.substring(0,
msg.indexOf(","))), Integer.parseInt(msg.substring(msg.indexOf(",") +
1)));
54                  }
55
56                  //when server chooses stone's color
57                  else if (msg.startsWith("[COLOR]")) {
58                      msg = msg.substring(7);
59                      //set server's stone's color
60                      board.setColor(Integer.parseInt(msg.substring(0,
msg.indexOf(","))), Integer.parseInt(msg.substring(msg.indexOf(",") +
1)));
61                  }
62
63                  //server notices client's lose
64                  else if (msg.startsWith("[LOSE]")) {
65                      msg = msg.substring(6);
66                      JOptionPane.showMessageDialog(null, msg);
67                      //clinet will choose stone's color
68                      board.setInfo("Choose Stone");
69                      board.reset();
70
71                      //reprint screen of choosing stone's color
72                      board.stoneSelect();
73                  }
74
75                  //server notices client's win
76                  else if (msg.startsWith("[WIN]")) {
77                      msg = msg.substring(5);
78                      JOptionPane.showMessageDialog(null, msg);
79                      //server will choose stone's color
80                      board.setInfo("Server Is Choosing Stone");
81                      board.reset();
82                  }
83              }
84          } catch (Exception e) {
85              System.out.println(e.getMessage());
```

```
        } finally {
            try {
                socket.close();
            } catch (Exception e) {
            }
        }
    }

    //main method
    public static void main(String[] args) {
        Client client = new Client("Omok Game : Client");
        client.setBounds(500,50,650,670);
        client.setVisible(true);
        client.connect();
    }
}
```

3. Server의 오목판 class : omok board screen for server

```
package Omok_Server;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.PrintWriter;

public class Board extends Canvas {
    public static final int BLACK = 1, WHITE = -1;
    public static final int size = 30; //size of one room
    public static final int num = 20; //maximum number of stones can be put
    public static final int x = 30; //start position of x
    public static final int y = 30; //start position of y
    public static final int width = 30; //size of a stone - width
    public static final int height = 30; //size of a stone - height

    private int color; //stone color of server
    private int client_color; //stone color of client
    private String info = "waiting for connection"; //print string for stuatus
    private String str_color; //string to print server's stone color
    private boolean enable = false; //activation info of board
    private PrintWriter writer; //stream to deliver message to client
    private int Stone[][] = new int [num][num]; // store stones' positions

    public Board(){
        this.setVisible(true);
```

```
28          this.setBackground(new Color(200,200,100));
29          //click mouse
30          addMouseListener(new MouseAdapter() {
31              public void mousePressed(MouseEvent e) {
32                  //if board is inactive
33                  if(!enable)
34                      return;
35
36                  //out of board boundary : right & left
37                  else if(e.getX() > x+size*(num-1) || e.getY() > y+size*
(num-1))
38                      return;
39
40                  //out of board boundary : up & down
41                  else if(e.getX() < (x-size/2) || e.getY() < (y-size/2))
42                      return;
43
44                  //already a stone exists
45                  else if(Stone[(e.getX()-x+size/2)/size][(e.getY()-
y+size/2)/size] != 0)
46                      return;
47
48                  //put stone
49                  else
50                      Stone[(e.getX()-x+size/2)/size][(e.getY()-
y+size/2)/size] = color;
51
52                  //send stone's position to client
53                  writer.println("[STONE]" + (e.getX() - x + size/2)/size +
"," + (e.getY() - y + size/2)/size);
54
55                  //check server's status >> game ends & server win
56                  if(check(color) == true){
57                      repaint();
58                      //print server's winning message
59                      JOptionPane.showMessageDialog(null, "Server Win");
60                      //send losing message to client
61                      writer.println("[LOSE] You Lose");
62                      //it's server's turn to choose stone
63                      info = "Server Is Choosing Stone";
64                      reset();
65                      return;
66                  }
67
68                  info = "Server is putting stone";
69                  repaint();
70                  enable = false;
71              }
72          });
```

```
73      }
74
75      //checking win/lose method
76      boolean check(int color){
77          for(int i = 0; i<num-4; i++){
78              for(int j = 0; j<num; j++){
79                  if(Stone[i][j] == color && Stone[i+1][j] == color &&
Stone[i+2][j] == color && Stone[i+3][j] == color && Stone[i+4][j] ==
color)
80                      return true;
81              }
82          }
83
84          for(int i = 0; i<num; i++){
85              for(int j = 0; j<num-4; j++){
86                  if(Stone[i][j] == color && Stone[i][j+1]==color &&
Stone[i][j+2]==color && Stone[i][j+3] == color && Stone[i][j+4] == color)
87                      return true;
88              }
89          }
90
91          for(int i = 19; i>3; i--){
92              for(int j = 0; j<num-4; j++){
93                  if(Stone[i][j] == color && Stone[i-1][j+1] == color &&
Stone[i-2][j+2] == color && Stone[i-3][j+3] == color && Stone[i-4][j+4]
== color)
94                      return true;
95              }
96          }
97
98          for(int i = 0; i <num-4; i++){
99              for(int j = 0; j<num-4; j++){
100                 if(Stone[i][j] == color && Stone[i+1][j+1] == color &&
Stone[i+2][j+2] == color && Stone[i+3][j+3] == color && Stone[i+4][j+4]
== color)
101                     return true;
102             }
103         }
104         return false;
105     }
106
107     //when the game ends, reset the game
108     void reset(){
109         for(int i = 0; i<num; i++){
110             for(int j = 0; j<num; j++){
111                 Stone[i][j] = 0;
112             }
113         }
114         repaint();
```

```
115         //make the board unactivated.
116         setEnable(false);
117     }
118 
119     public void paint(Graphics g){
120         g.clearRect(0, 0, getWidth(), getHeight());
121 
122         //print info
123         g.setColor(Color.RED);
124         g.drawString(info, 30, 20);
125 
126         //draw line
127         for(int i = 0; i<num; i++){
128             //choose color of line
129             g.setColor(Color.BLACK);
130             //horizontal line
131             g.drawLine(x, y + size*i, x + size*(num-1), y + size*i);
132             //vertical line
133             g.drawLine(x + size*i, y, x + size*i, y + size*(num-1));
134         }
135 
136         //put stone
137         for(int i = 0; i <num; i++){
138             for(int j = 0; j<num; j++){
139                 //black stone
140                 if(Stone[i][j] == BLACK){
141                     g.setColor(Color.BLACK);
142                     g.fillOval((x-size/2) + i*size, (y-size/2) + j*size,
width, height);
143                 }
144                 //white stone
145                 else if(Stone[i][j] == WHITE){
146                     g.setColor(Color.WHITE);
147                     g.fillOval((x-size/2) + i*size, (y-size/2) + j*size,
width, height);
148                 }
149             }
150         }
151     }
152 
153     //message for choosing stone color
154     public void stoneSelect(){
155         if(JOptionPane.showOptionDialog(this, "Black Or White?", "Choose
color of stone", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null, new String[]{"Black", "White"}, "Black") ==0){
156             setColor(BLACK, WHITE);
157             writer.println("[COLOR]" + WHITE + "," + BLACK);
158         }
159         else{
```

```
            setColor(WHITE, BLACK);
            writer.println("[COLOR]" + BLACK + "," + WHITE);
        }
    }

    public void setEnable(boolean enable){
        this.enable = enable;
    }

    public void setWriter(PrintWriter writer){
        this.writer = writer;
    }

    public void setColor(int color, int client_color){
        this.color= color;
        this.client_color = client_color;

        if(color == BLACK){
            setEnable(true);
            info = "Stone Color : Black - first attack";
            str_color = "Black";
        }
        else{
            setEnable(false);
            info = "Stone Color : White - second attack";
            str_color = "White";
        }

        repaint();
    }


    public void setInfo(String info){
        this.info = info;
    }

    //put client's stone
    public void putClient(int x, int y){
        Stone[x][y] = client_color;
        info = "My turn (" + str_color + ") - Put you stone";
        repaint();
        //check client's win/lose
        if(check(client_color) == true){
            repaint();
            JOptionPane.showMessageDialog(null, "Server Lose");
            writer.println("[WIN] Client Win");
            info = "Choose Stone";
            reset();
            stoneSelect();
```

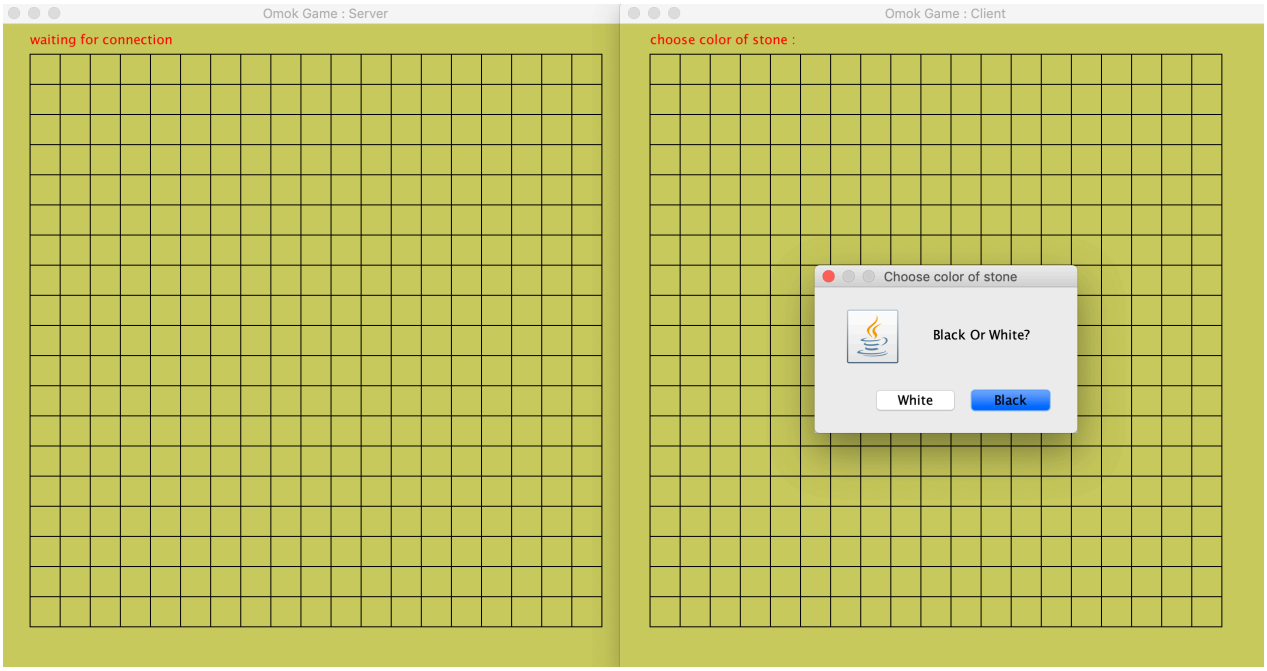
```
209             }
210 
211         }
212 
213 }
```

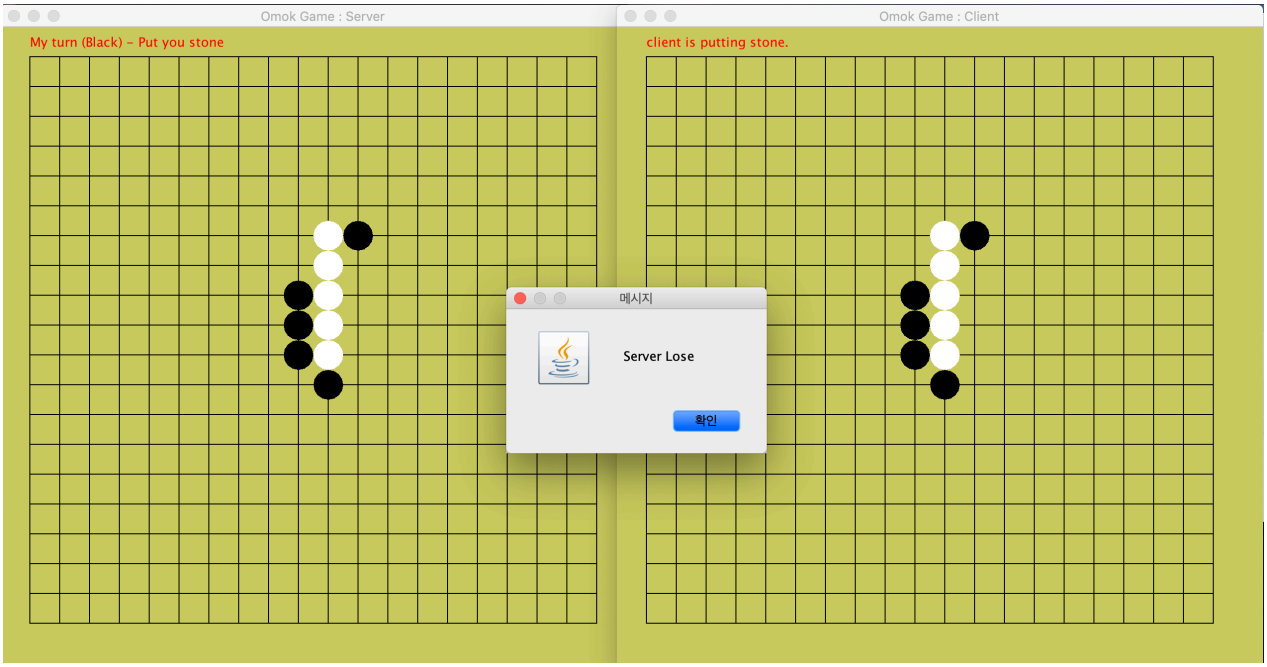
4. Server class : creates object of board. Marks server's playing and clinet's stone color and playing. Accepts client's game conclusion message and do proper actions by the message.

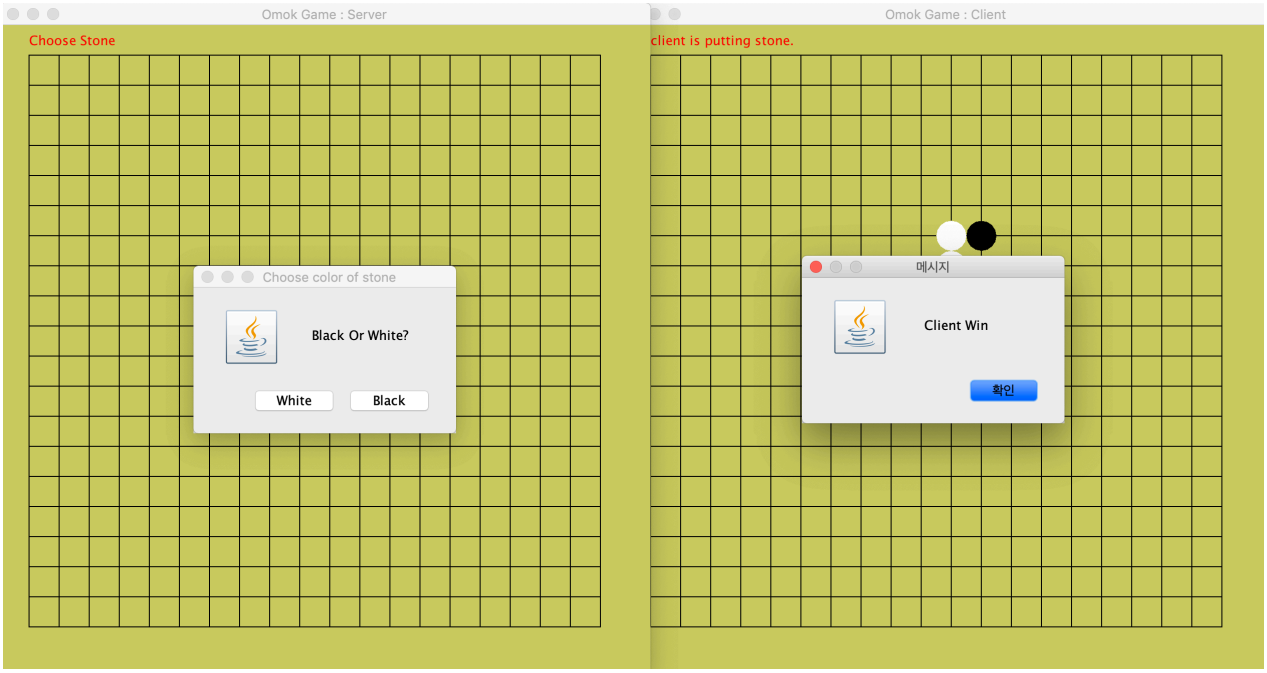
```
1  package Omok_Server;
2  
3  import java.awt.*;
4  import java.awt.event.WindowAdapter;
5  import java.awt.event.WindowEvent;
6  import java.io.BufferedReader;
7  import java.io.InputStreamReader;
8  import java.io.PrintWriter;
9  import java.net.ServerSocket;
10 import java.net.Socket;
11 
12 public class Server extends Frame {
13 
14     //object of omok game board
15     private Board board = new Board();
16 
17     ServerSocket serverSocket = null;
18     Socket socket = null;
19 
20     private BufferedReader reader;
21     private PrintWriter writer;
22 
23     //contructor
24     public Server(String name){
25         super(name);
26         add(board);
27 
28         addWindowListener(new WindowAdapter() {
29             public void windowClosing(WindowEvent w){
30                 System.exit(0);
31             }
32         });
33     }
34 
35     //connection
36     public void connect(){
37         try{
38             //put client's port num
39             serverSocket = new ServerSocket(0516);
40 
```

```
            socket = serverSocket.accept();
            reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            writer = new PrintWriter(socket.getOutputStream(), true);

            board.setWriter(writer);
            board.setInfo("상대 Is Choosing Stone");

            String msg;

            while((msg = reader.readLine()) != null){
                //when server puts a stone
                if(msg.startsWith("[STONE")){
                    msg = msg.substring(7);
                    //it is server's turn
                    board.setEnable(true);
                    board.putClient(Integer.parseInt(msg.substring(0,
msg.indexOf(","))), Integer.parseInt(msg.substring(msg.indexOf(",") +1)));
                }

                //when client chooses stone's color
                else if(msg.startsWith("[COLOR]")){
                    msg = msg.substring(7);
                    board.setColor(Integer.parseInt(msg.substring(0,
msg.indexOf(","))), Integer.parseInt(msg.substring(msg.indexOf(",") +1)));
                }
            }
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
        finally {
            try{
                serverSocket.close();
                socket.close();
            }
            catch (Exception e) {}
        }
    }

    //main method
    public static void main(String[] args) {
        Server server = new Server("Omok Game : Server");
        server.setBounds(0,0,650,670);
        server.setVisible(true);
        server.connect();
    }
}
```

3. Result (playing game)






Omok Game : Server
Choose Stone
Choose color of stone
Black Or White?
White
Black
Omok Game : Client
client is putting stone.
메시지
Client Win
확인