1. Platform:

1.1 Why I use TCP Socket

Both Communication and Game I using TCP socket. Because UDP is not reliable, and this means that it may missing something when they communication with server or clients. The UDP socket is suitable for transmit the picture, live video stream or others. So, If I design a video chat, I will use UDP.

2. The operation of the programming:

2.1 Start:

The Game has 2 Servers (Sockets), one is for Game, and another one for Communication (Chat). When Server start, the Game socket will use port 6666, and the Communication will use port 7777.

```
ServerSocket serverSocket = new ServerSocket(LISTENING_PORT);
```

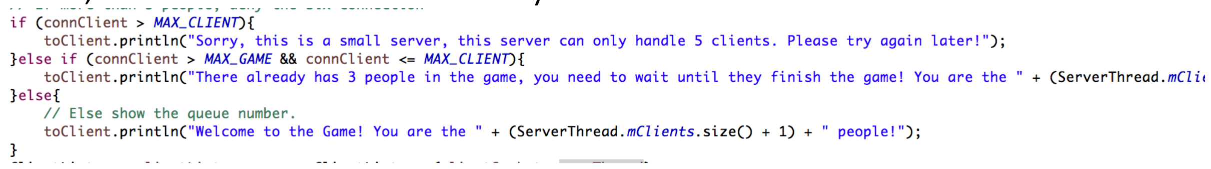
2.2 Connect:

Then, the client will connect to the Game server, and begin the game.

```
socket = new Socket(SERVER_HOSTNAME, SERVER_PORT);
```

2.3 Handle Clients:

As the requirement, the server can only handle 5 clients. So that, the server will check how much clients that connect to the server. If greater than 3 and less than 5, they will send a notification the clients, and put them in the waiting list. If greater than 5, if will send a notification and deny the connection.



```
if (connClient > MAX_CLIENT){
    toClient.println("Sorry, this is a small server, this server can only handle 5 clients. Please try again later!");
}else if (connClient > MAX_GAME && connClient <= MAX_CLIENT){
    toClient.println("There already has 3 people in the game, you need to wait until they finish the game! You are the " + (ServerThread.mCli
}else{
    // Else show the queue number.
    toClient.println("Welcome to the Game! You are the " + (ServerThread.mClients.size() + 1) + " people!");
}
```

2.4 Register:

The Client(Player) can register anytime, but they must register before the game. If they do not register, they cannot play the game. The register only need people input the name.

2.5 Waiting List:

Before Game, the server will check how many clients that connect to the server. (AS 2.3 Mentioned)

If there only have one client, it will wait until the second client connect. (Because it is 'multiple player game', so I assume that the game need at least 2 players)

If there have two clients, the game will be run as two clients' module.

If there have three clients, the game will be run as three clients' module.

If there have greater than 3 and less than 5 clients, the client 4 and 5 will go to waiting list.

If there have greater than 6 clients, the connection will be denied. (AS 2.3 Mentioned)

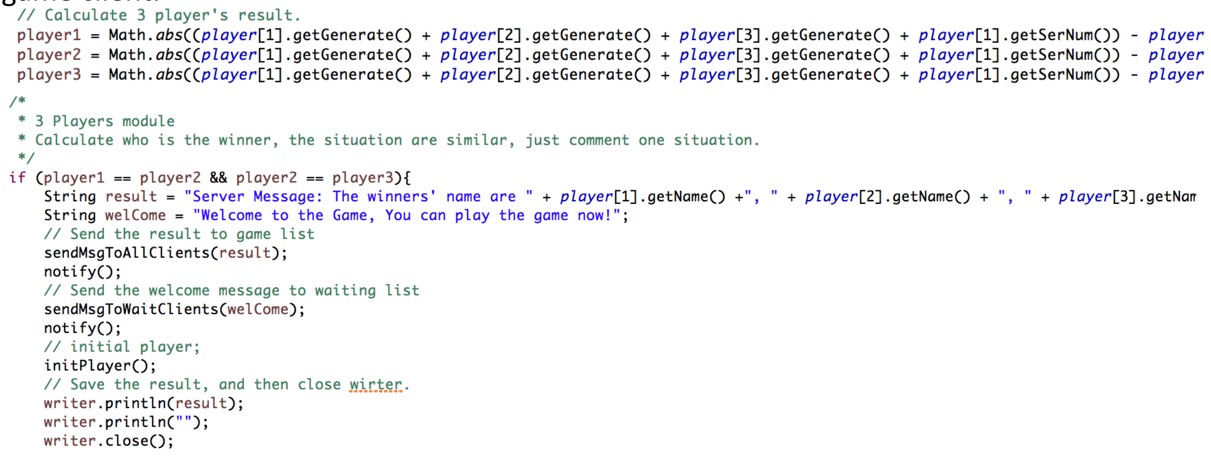
The clients who are in the waiting list, they **can** register but they **cannot** start the game.

2.6 Game Process:

Clients(Players) input number that they generate, and input the number that they guess. And the clients' name, generate number and guess number to the server. And server will finish register the game.

```
player[1] = new Player(usrName, generNum, guessNum, serverRandom);
player[2] = new Player(usrName, generNum, guessNum, serverRandom);
player[3] = new Player(usrName, generNum, guessNum, serverRandom);
```

Then, They server will calculate who is the winner, and send the notification to all game client.



```
 // Calculate 3 player's result.
 player1 = Math.abs((player[1].getGenerate() + player[2].getGenerate() + player[3].getGenerate() + player[1].getSerNum()) - player
 player2 = Math.abs((player[1].getGenerate() + player[2].getGenerate() + player[3].getGenerate() + player[1].getSerNum()) - player
 player3 = Math.abs((player[1].getGenerate() + player[2].getGenerate() + player[3].getGenerate() + player[1].getSerNum()) - player

/*
 * 3 Players module
 * Calculate who is the winner, the situation are similar, just comment one situation.
 */
if (player1 == player2 && player2 == player3){
    String result = "Server Message: The winners' name are " + player[1].getName() +", " + player[2].getName() + ", " + player[3].getNam
    String welCome = "Welcome to the Game, You can play the game now!";
    // Send the result to game list
    sendMsgToAllClients(result);
    notify();
    // Send the welcome message to waiting list
    sendMsgToWaitClients(welCome);
    notify();
    // initial player;
    initPlayer();
    // Save the result, and then close wirter.
    writer.println(result);
    writer.println("");
    writer.close();
```

Then, it will send the notification to the clients who are in the wait list. And the clients can begin the game.

```
 // Send the notification to waiting list clients
 public synchronized void sendMsgToWaitClients(String aMsg) {
     // Get the client's size
     // Using for loop to send all clients
     for (int i = 3; i < mClients.size(); i++) {
         Socket socket = mClients.get(i);
         try {
             java.io.OutputStream out = socket.getOutputStream();
             out.write(aMsg.getBytes());
             out.flush();
         } catch (IOException ioe) {
             deleteClient(socket);
         }
     }
 }
```

2.7 Communication

Because it is a multiple player guessing game. So, I assume during the game, the player cannot communicate, otherwise, they will cheat. Once they finish the game, they will automatically go to the communication room to chat. The Communication server is similar with Game server.

2.8 Log

Both Communication and echo game's players information, generate number, guess number and result will be stored in the result.txt (Game Result and Player Information) and communication.txt (communication content, sender's name, date and time)

```
PrintWriter writer = new PrintWriter(new FileWriter("result.txt", true));
PrintWriter writer = new PrintWriter(new FileWriter("communication.txt", true));
```