浙江大学计算机科学与技术学院

Java 程序设计课程报告

2020—2021 学年秋冬学期

题目	聊天室软件开发
学号	3180102095
学生姓名	臧可
所在专业	计算机科学与技术
所在班级	计科 1804

目　录

1 引言

本次开发的是一个聊天室软件，用 Java 语言编写，编译环境是 Eclipse（安装插件 WindowBuilder）和 JDK1.8.0_202。用 WindowBuilder 辅助设计了用户界面用 GUI，使用网络编程，能支持多组用户同时使用。

1．1 设计目的

聊天室可以支持群聊和私聊。具体功能如下：

(1) 每个登录用户可以选择一个用户名，拒绝重复的用户名。

(2) 登录后可以选择不同的接收方发送消息，支持群聊（选择所有用户）和私聊（选择其他在线用户）

(3) 不支持自己和自己聊天

(4) 登录一个客户端服务器会显示当前客户端数量，同理客户端下线也会更新并显示当前客户端数量。

(5) 未登录不能发送消息。服务器断连发送消息会提示。

1．2 设计说明

本程序用 Java 语言编写，编译环境是 Eclipse（安装插件 WindowBuilder）和 JDK1.8.0_202。

2 总体设计

2．1 功能模块设计

本程序需实现的主要功能有：

(1) 服务器可以监视到当前所有客户端状态

(2) 用户可以群聊

(3) 用户可以私聊

程序的总体功能如图 1 所示：

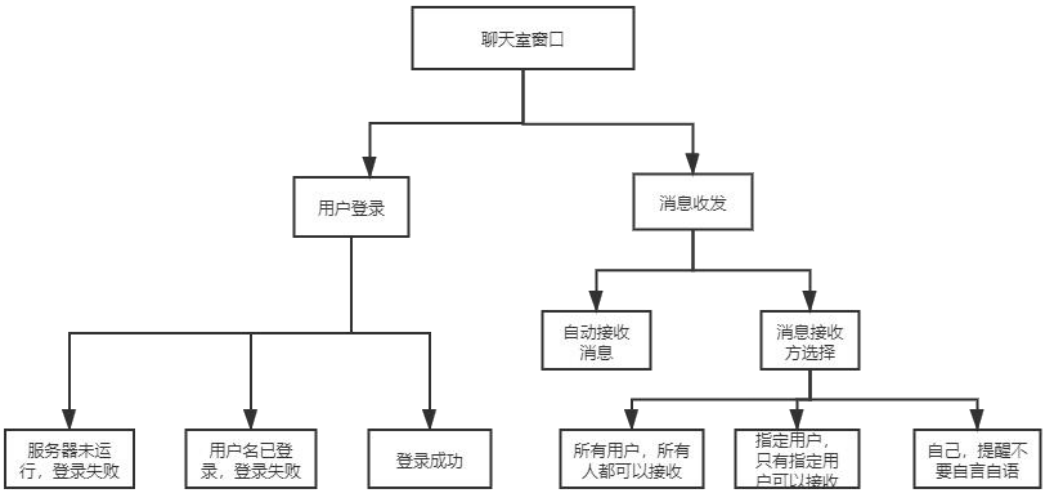

图 1 程序总体功能图

2. 2 流程图设计

服务器和客户端的总体流程如图 2 所示：

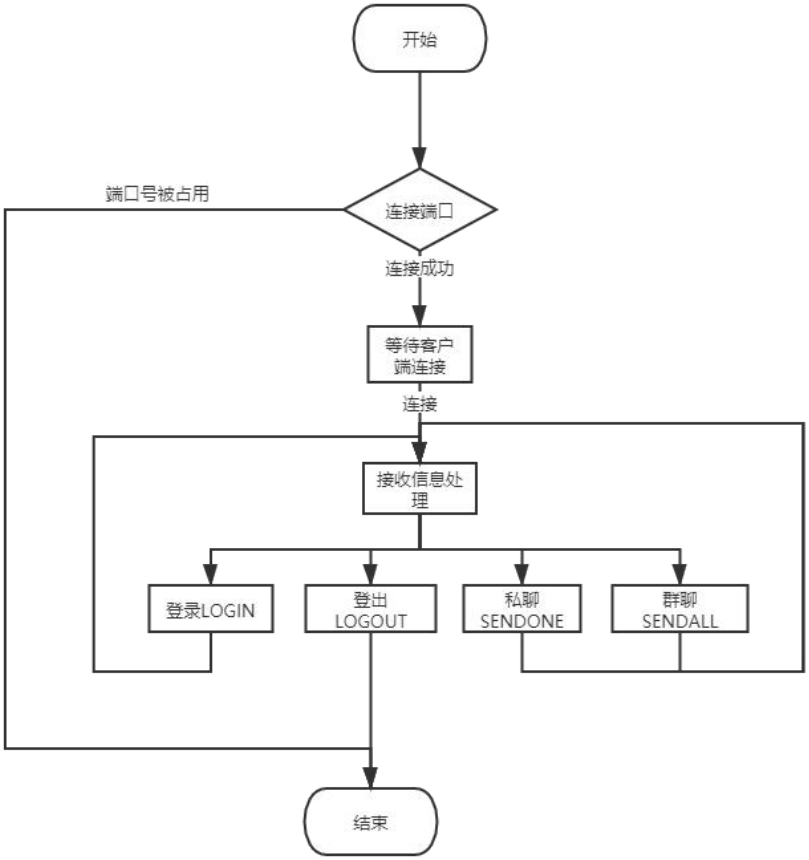

图 2 服务器流程图

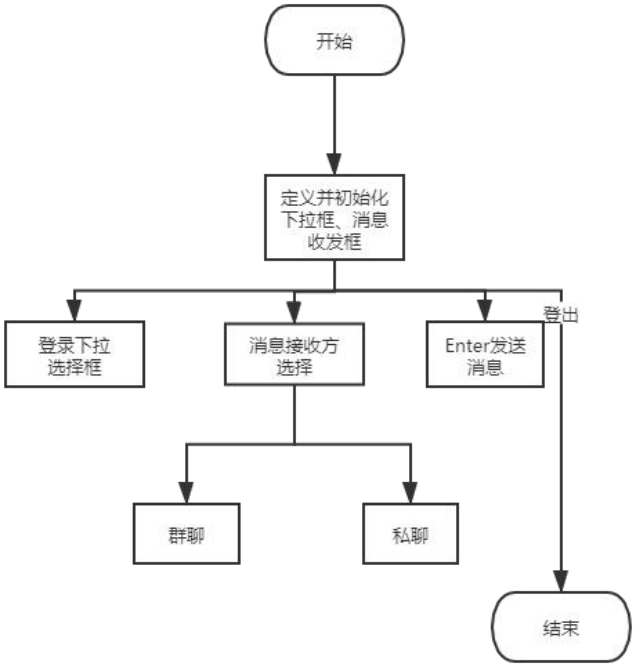

图 3 客户端流程图

3　详细设计

3．1　服务器设计

服务器写在 ServerClient.java 中。

启动服务器。new ServerSocket(8888)连接端口 8888，如果端口被占用则会抛出异常，不被占用则启动服务器。当客户端连接成功的时候通过全局变量 clientNumber 来打印输出当前客户端的数量。

```
public void start() {
        try {
            ss = new ServerSocket(8888);
            serverFlag = true;
            System.out.println("启动服务器...");
        } catch (BindException e) {
            System.out.println("端口8888被占用");
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            //Receive client connection
            while (serverFlag) {
                Socket socket = ss.accept();
                ServerClient c = new ServerClient(socket);
                clientNumber++;
                System.out.println(clientNumber + "个客户端已经连接");
                new Thread(c).start();
            }
        } catch (IOException e) {
            System.out.println("客户端关闭");
        } finally {
            try {
                if (ss != null)
                    ss.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

```
                    }
```

ServerClient 的构造函数，s 为接收到的客户端，input、ouput 分别存储输入输出消息。当前连接状态 connected 置为 true。

```
public ServerClient(Socket s) {
            this.s = s;
            try {
                input = new DataInputStream(s.getInputStream());
                output = new DataOutputStream(s.getOutputStream());
                connected = true;
            } catch (IOException e) {
                e.printStackTrace();
            }
                    }
```

重写 run()。如果当前客户端和服务器连接成功，则进入收发消息的循环。Msg[]以{类型，“用户”，“发送信息”}的格式存储。用 switch 语句捕获发送的指令类型，“LOGIN”是客户端选择用户登录时的消息处理，“LOGOUT”是用户关闭客户端后的消息处理，“SENDONE”是用户选择私聊时的消息处理，“SENDALL”是用户选择群聊时的消息处理。

```
@Override
        public void run() {
            try {
                while (connected) {
                    String msg[] = input.readUTF().split("#");// msg={指令,"用户","
要发送的信息"}
                    switch (msg[0]) {
                    case "LOGIN":
                        String userName = msg[1];
                        if (clients.containsKey(userName)) {//If the user is already
in the Client
                            output.writeUTF("FAIL");
                            System.out.println("拒绝了一个重复连接");
                            closeConnect();
                        } else {
                            output.writeUTF("SUCCESS");
                            clients.put(userName, this);
                            //send the info of all logged user to the new user
                            StringBuffer allUsers = new StringBuffer();
```

```
                            allUsers.append("ALLUSERS#");
                            for (String user : clients.keySet())
                                allUsers.append(user + "#");
                            output.writeUTF(allUsers.toString());
                            //Send the newly logged-in user information to other users
                            String newLogin = "LOGIN#" + userName;
                            sendMsg(userName, newLogin);
                            this.userName = userName;
                        }
                        break;
                    case "LOGOUT":
                        clients.remove(this.userName);
                        String logoutMsg = "LOGOUT#" + this.userName;
                        sendMsg(this.userName, logoutMsg);
                        System.out.println("用户" + this.userName + "已下线...");
                        clientNumber--;
                        if (clientNumber != 0)
                            System.out.println(clientNumber + "个客户端已经连
接...");
                        else
                            System.out.println("当前无客户端连接");
                        closeConnect();
                        break;
                    case "SENDONE"://send message to one user
                        ServerClient c = clients.get(msg[1]);//Get the connection of
the target user
                        String msgToOne = "";
                        if (c != null) {
                            msgToOne = "SENDONE#" + this.userName + "#" + msg[2];
                            c.output.writeUTF(msgToOne);
                            c.output.flush();
                        }
                        break;
                    case "SENDALL"://send message to all user
                        String msgToAll = "";
                        msgToAll = "SENDALL#" + this.userName + "#" + msg[1];
                        sendMsg(this.userName, msgToAll);
                        break;

                    }

                }
            } catch (IOException e) {
                System.out.println("Client closed...");
```

```
                connected = false;
            } finally {

                try {
                    if (input != null)
                        input.close();
                    if (output != null)
                        output.close();
                    if (fout != null)
                        fout.close();
                    if (s != null)
                        s.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }

            }
                    }
```

登出的时候调用 closeConnect()函数。把 connected 状态设置为 false，关闭正在运行的。

```
public void closeConnect() {
            connected = false;
            try {
                if (input != null)
                    input.close();
                if (output != null)
                    output.close();
                if (s != null)
                    s.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
                    }
```

群发送消息的时候调用 sendMsg（）函数。对于当前用户列中每个不是自己的用户发送 msg。

```
public void sendMsg(String fromUser, String msg) {
            String tempUser = "";
            try {
                for (String toUser : clients.keySet()) {
                    if (!toUser.equals(fromUser)) {
                        tempUser = toUser;
```

```
                        DataOutputStream out = clients.get(toUser).output;
                        out.writeUTF(msg);
                        out.flush();
                    }
                }
            } catch (IOException e) {
                System.out.println("用户" + tempUser + "已经离线！！！");
            }
                    }
```

3.2 客户端 UI 设计

Main 函数中放 UI 美化代码和 jframe 默认布局代码。

```
public static void main(String args[]) {
    //Render UI
    try {
           for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
               if ("Nimbus".equals(info.getName())) {
                   javax.swing.UIManager.setLookAndFeel(info.getClassName());
                   break;
               }
           }
       }catch(Exception e) {
        System.out.println(e);
       }
       try {
        Client2 frame = new Client2();
            frame.setVisible(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
   }
```

主要 UI 布局在 Client2 的构造函数中。sendArea 是待发送消息的输入文本框，contentArea 是接收消息的显示文本框。给 sendArea 添加 KeyListener,输入完内容后按 Enter 键就可以输出。userLog 和 userOnline 分别是登录用户和在线用户的下拉选择框，初始化给 userLog 添加所有用户，添加一个 addItemListener，选择完用户后开始调用函数连接服务器进行登录。

```
public Client2() {
```

```
        contentPane = new JPanel();
         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
         contentPane.setLayout(null);
        contentPane.setLayout(null);
        setContentPane(contentPane);
        setResizable(false);
        setTitle("聊天室");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         setBounds(100, 100, 1000, 800);

        JLabel lbl_uname = new JLabel("\u5F53\u524D\u7528\u6237");
        lbl_uname.setFont(new Font("宋体", Font.PLAIN, 24));
        lbl_uname.setBounds(825, 520, 138, 31);
        contentPane.add(lbl_uname);

        JLabel lblNewLabel = new JLabel("\u63A5\u53D7\u65B9");
        lblNewLabel.setFont(new Font("宋体", Font.PLAIN, 24));
        lblNewLabel.setBounds(825, 10, 106, 31);
        contentPane.add(lblNewLabel);

         sendArea = new JTextArea();
         sendArea.setFont(new Font("Monospaced", Font.PLAIN, 24));
        sendArea.setBounds(10, 520, 800, 219);
        contentPane.add(sendArea);
        sendArea.setLineWrap(true);
        sendArea.setWrapStyleWord(true);//Activate line break and continuous word
function
        sendArea.addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if ('\n' == e.getKeyCode())//press Enter key
                    if (clientThread != null) {
                        clientThread.sendMsg();
                    }
            }
        });

        contentArea = new JTextArea();
        contentArea.setFont(new Font("Monospaced", Font.PLAIN, 24));
        contentArea.setBounds(10, 10, 800, 500);
        contentPane.add(contentArea);

        userLog = new JComboBox<>();
        userLog.setFont(new Font("宋体", Font.PLAIN, 24));
```

```
        userLog.setBounds(825, 552, 128, 31);
        contentPane.add(userLog);
        userLog.addItem("请选择");
        userLog.addItem("小红");
        userLog.addItem("小白");
        userLog.addItem("小猫");
        userLog.addItem("小明");
        userLog.addItem("小王");
        userLog.addItemListener(e -> {
            if (e.getStateChange() == ItemEvent.SELECTED) {
                    System.out.println("选中的项：" + userLog.getSelectedItem());//debug
用
                if (userLog.getSelectedIndex() == 0) {
                    return;
                }
                clientThread = new RecvThread((String) userLog.getSelectedItem());//Get
login user info
                new Thread(clientThread).start();
            }
        });

        userOnline = new JComboBox<>();
        userOnline.setFont(new Font("宋体", Font.PLAIN, 24));
        userOnline.setBounds(825, 50, 128, 31);
        contentPane.add(userOnline);
         userOnline.addItem("所有用户");

        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                if (clientThread != null)
                    clientThread.exit();
                System.exit(0);
            }
        });
    }
```

3．3　客户端功能的设计

RecvThread 构造函数初始化 username，存储当前登陆的用户名。

```
public RecvThread(String userName) {
            this.userName = userName;
        }
```

login（）函数处理用户登录。通过 new Socket 监听本机 8888 端口，绑定输入输出流。用 sendMsg 字符串记录当前用户的登录信息发送给客户端，用 recv 字符串记录客户端返回的信息。根据返回信息调用 showMsg（）函数显示登录结果。登录成功话 isLogin 设为 true，用于 rnuu（）函数内操作执行的判断。

```
//Log
        public void login() {
            try {
                s = new Socket("127.0.0.1", 8888);//Monitor port 8888 of the host
                in = new DataInputStream(s.getInputStream());
                out = new DataOutputStream(s.getOutputStream());
                String sendMsg = "LOGIN#" + userName;
                out.writeUTF(sendMsg);
                out.flush();
                //Information returned by the server
                String recv = in.readUTF();
                if (recv.equals("FAIL")) {//If the user is already in the Client2
                    showMsg("该用户已登录！");
                    userLog.setEnabled(true);
                    exit();
                    return;
                } else if (recv.equals("SUCCESS")) {
                    showMsg("登录成功！");
                    userLog.setEnabled(false);
                    isLogin = true;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

重写 run()函数。首先调用 login（）函数登录。读取客户端返回的信息，根据不同的返回信息调用 showMsg 执行相应的操作显示相应信息在接收消息窗口。异常抛出用于判断用户是否登出，用户登出也会关闭窗口。

```
@Override
        public void run() {
            login();
            try {
                while (isLogin) {
                    String msgs[] = in.readUTF().split("#");
                    switch (msgs[0]) {
                        case "LOGIN":
                            userOnline.addItem(msgs[1]);//add online user
                            break;
                        case "ALLUSERS":
                            for (int i = 1; i < msgs.length; i++) {
                                if (!"".equals(msgs[i]))
                                    userOnline.addItem(msgs[i]);
                            }
                            break;
                        case "SENDONE":
                                showMsg(msgs[1] + ": " + msgs[2]);
                            break;
                        case "SENDALL":
                                showMsg(msgs[1] + "(世界): " + msgs[2]);
                            break;
                        case "LOGOUT":
                            showMsg("用户" + msgs[1] + "已下线");
                            userOnline.removeItem(msgs[1]);
                            break;
                    }
                }
            } catch (SocketException e) {
                System.out.println(userName + "已退出...");
            } catch (IOException e) {
                isLogin = false;//Return data is abnormal, log out
                e.printStackTrace();
            }
        }
    }
```

通过 showMsg 来显示相应的 msg 信息。把 msg 的值传递 contentAera 显示，并且设置自动换行。

```
//show message
    public void showMsg(String msg) {
     contentArea.append(msg + "\n");
        contentArea.setCaretPosition(contentArea.getText().length());//Automatically
scroll to the last line of the text area
    }
```

4 测试与运行

4．1 程序测试

在程序代码编译运行成功后，本次实验设计的聊天室能够正常运行，实现群聊和单独聊天的功能。因为时间原因，很遗憾没有进行用户登录的设计（只来及写一个 UI），本学期课程结束后有时间应该会进行一个功能的完善。

4．2 程序运行

首先运行 Server.java，启动服务器。命令行执行 java -jar ChatServer.jar，弹出防火墙提示，允许访问。

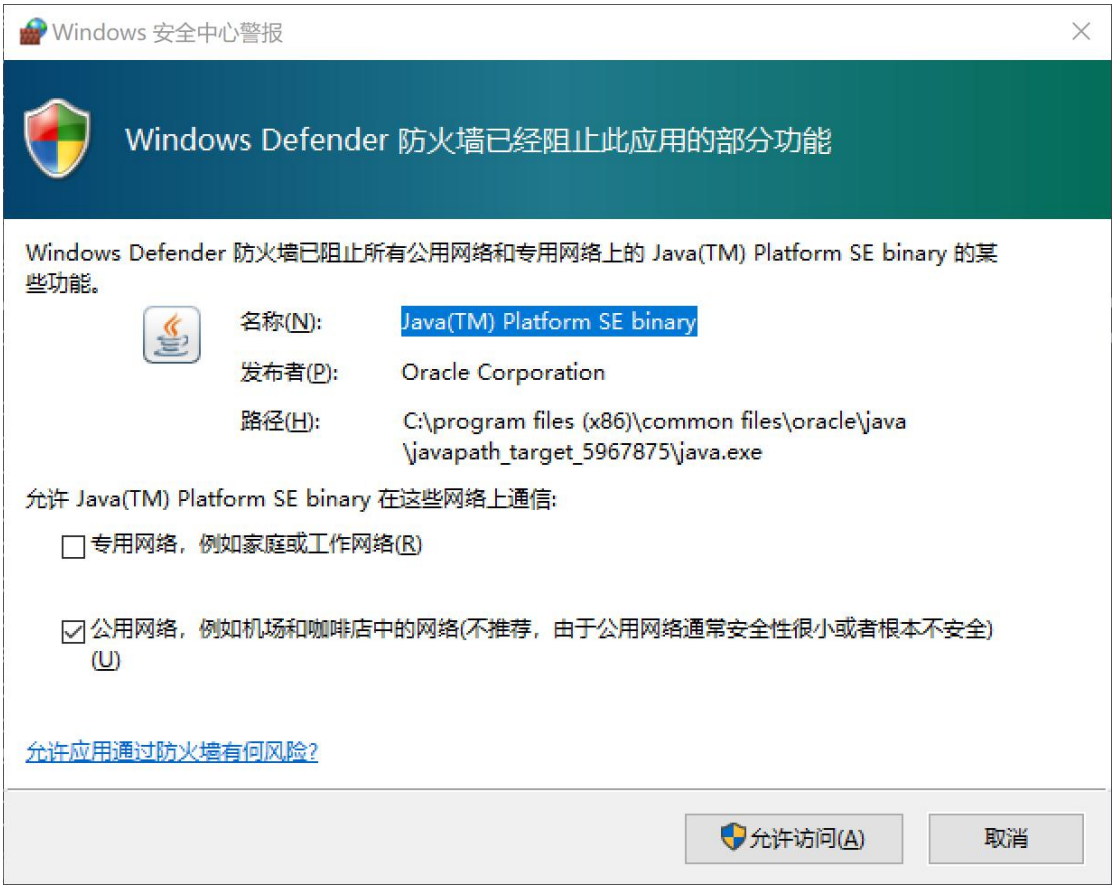

图 4 防火墙提示

允许访问后命令行显示服务器正在启动中

```
PS C:\JAVA\workspace2> java -jar ChatServer.jar
启动服务器...
```

图 5 启动服务器

另开一个窗口输入 java -jar .\ChatClient.jar 运行 Client2.java,弹出初始窗口如下图所示：

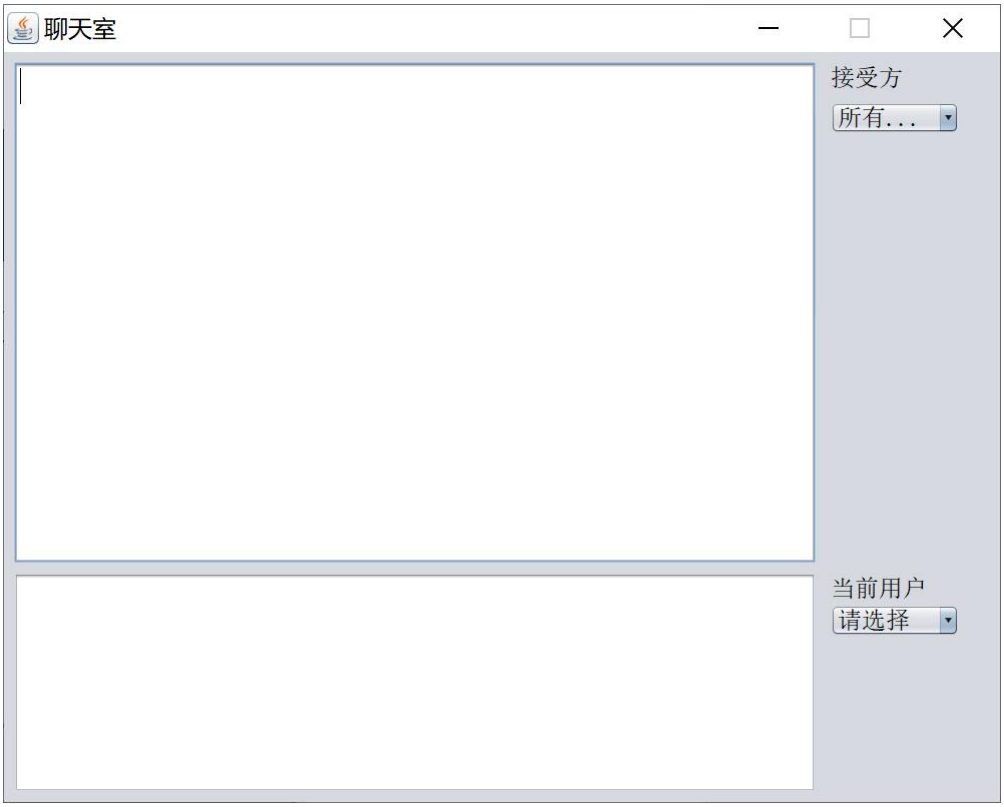

图 6 初始聊天室窗口

点击右下角的下拉框选择用户名登录，登录成功聊天室窗口会有提示：

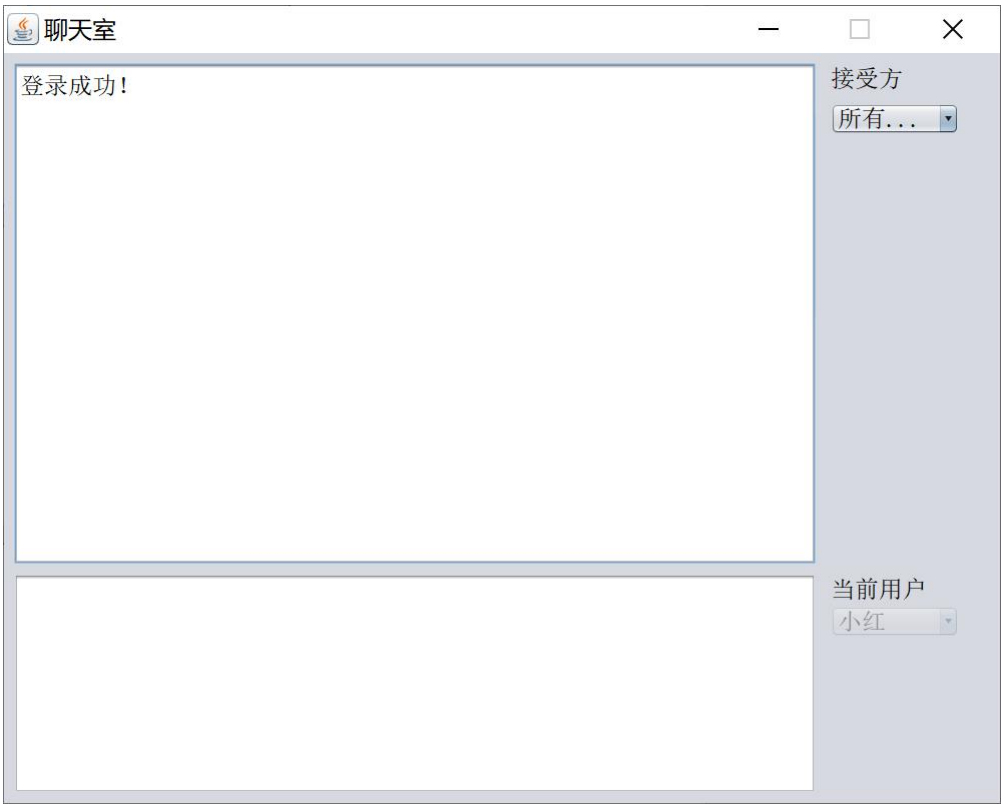

图 7 登录成功

同时 Server 的命令行窗口会显示当前连上的客户端数量：

```
PS C:\JAVA\workspace2> java -jar ChatServer.jar
启动服务器...
1个客户端已经连接
```

图 8 客户端数量

如果该用户名已登录，会拒绝重复登录：

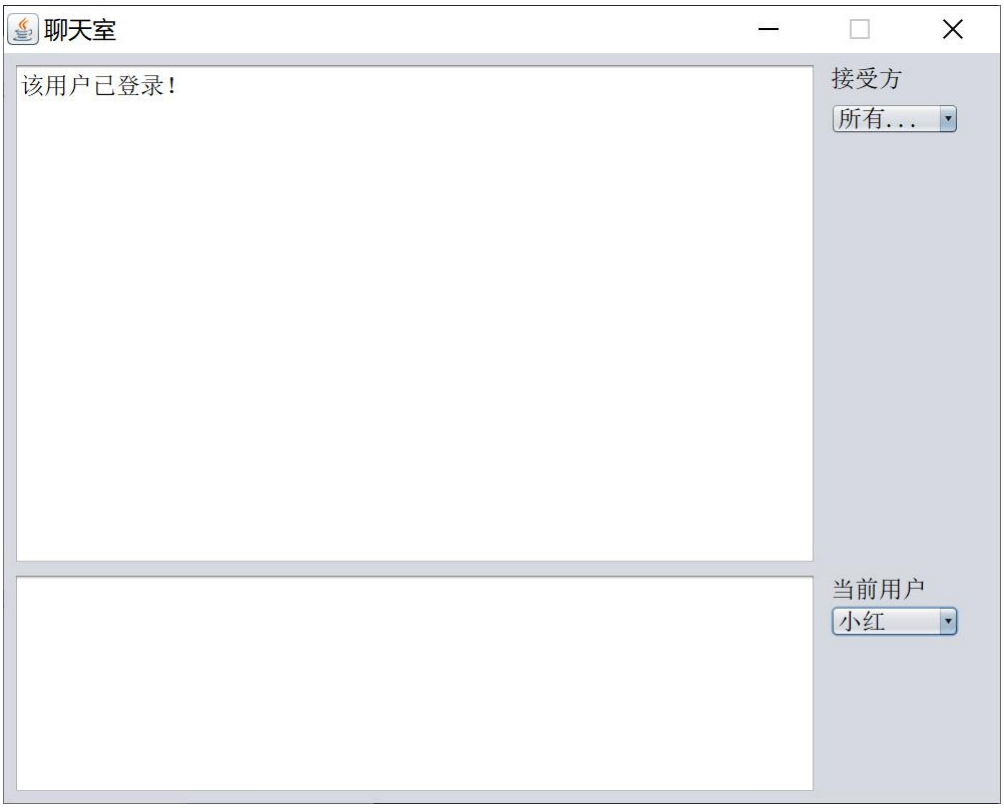

图 9 拒绝重复请求

默认状态下是对所有人发消息，尝试发送消息，并且再开 2 个 ps 窗口，运行另 2 个客户端观察消息接收情况：

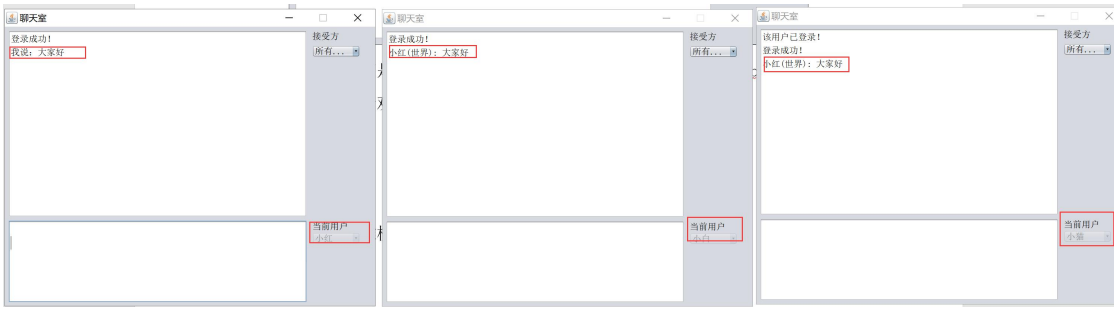

图 10 群聊

选择右上角接收方可以私聊，测试结果如下图：

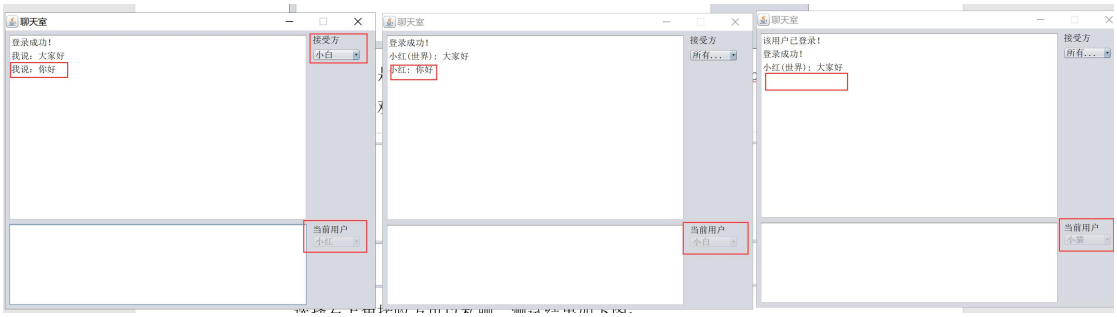

图 11 私聊

如果用户选择自己窗口将会提醒请不要自言自语：

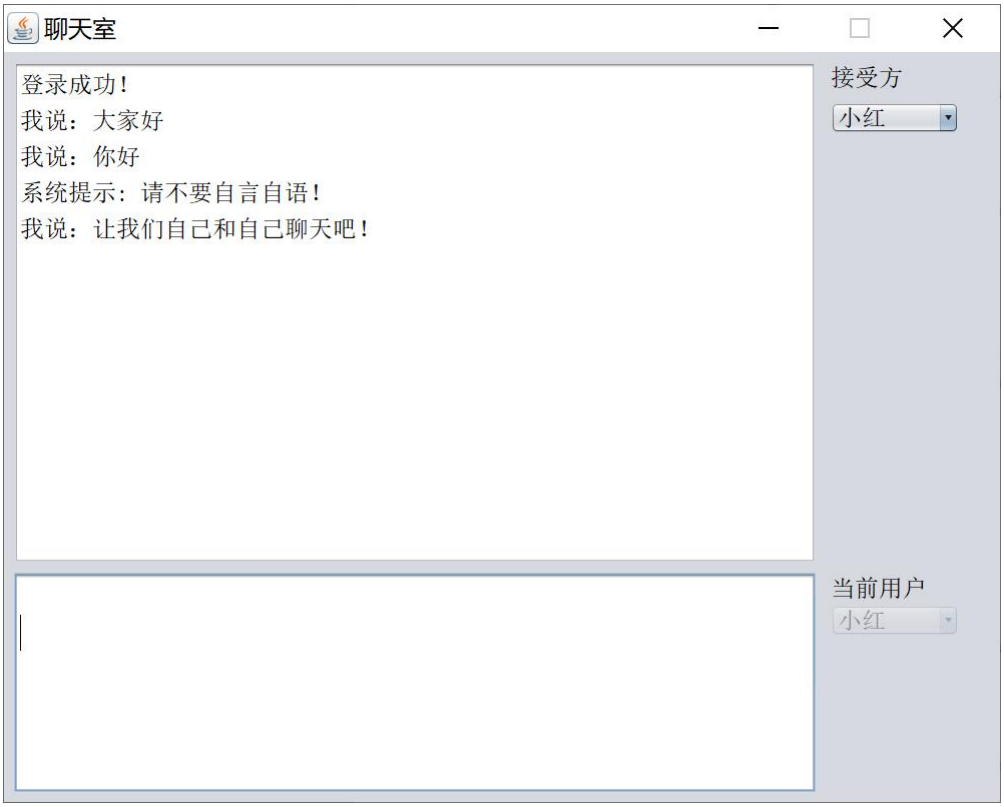

图 12 请不要自言自语

关闭窗口即可退出，退出后所有聊天室窗口会提示：

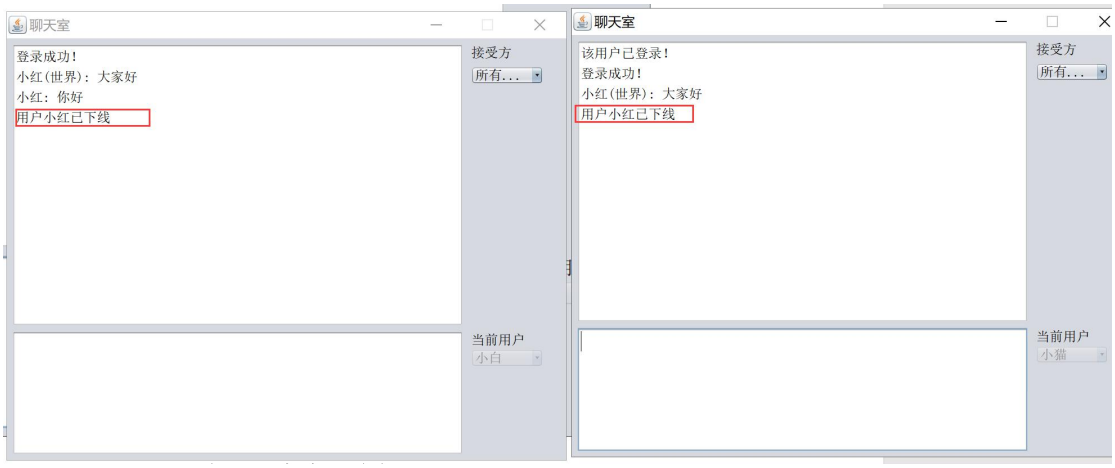

图 13 登出

服务器会显示当前在线的客户端数量。

5. 总结

本次课程作业因为是在期末阶段，完成较为匆忙，很多预期想要实现的功能并没有时间实现，只是完成了最基础的功能。因为个人算是第一次接触 Java 网络编程和 UI，在一些非常基础的函数调用上出了很多 bug，花了很多时间去调试才解决。

通过编写本次聊天室，我对于 Java 网络编程和 UI 都有了一定的掌握，现在可以熟练地把工程复现一遍了。最大的感受就是 Java 学习中最重要的还是实践，知道这个函数应该怎么用和真正会用之间还是有很大的差距的。在平时的知识点中需要注意细节，编程时才会少犯低级错误。编程的时候感受到 Java 的报错比起 c 语言都更为详细友好，有的报错不需要去搜索直接阅读英文就可以解决。

非常感谢我舍友的帮助。一开始我设计出来的 UI 很丑（明明花了很多时间去设计），后来在她的指导下安装了 WindowBuilder 的插件，可以进行一个可视化的设计，加上自己有 Qt 和安卓程序的开发经验，在编写 UI 的时候省了很多力气，并且最终页面也漂亮多了。

总而言之，本次实验成果做出来后，即便非常简陋，但是想到自己的付出和已经掌握的知识，还是很有成就感的。

参考文献

[1] Java 实例 - ServerSocket 和 Socket 通信实例
(https://www.runoob.com/java/net-serversocket-socket.html).
[2] UI 框架-UIManager(https://www.jianshu.com/p/65339e872699).
[3] Java WindowBuilder 安装及基本使用的教程
(https://www.jb51.net/article/187354.htm)