

**程序设计综合训练实验报告**

|  |  |
| --- | --- |
| 学生姓名 | 白明强 王云鹏 |
|  |  |
| 学生学号 | 1201170606  8213180228 |
|  |  |
| 指导教师 | 向遥 |
|  |  |
| 专业班级 | 物联网工程1802 |
|  |  |
| 完成日期 | 2020.7.26 |

**计算机学院**

**目 录**

[实验一 2](#_Toc8229)

[一、 目的与要求 2](#_Toc4506)

[二、 操作环境 2](#_Toc3216)

[三、 基本技术原理 1](#_Toc30233)

[四、 概要设计 1](#_Toc4338)

[五、 实验内容与实现 3](#_Toc17920)

[六、 遇到的问题与解决方案 6](#_Toc12858)

**七、 实验结果........................................................................................................................13**

[八、实验源代码 16](#_Toc14031)

[九、实验总结 39](#_Toc19189)

[十、参考资料 40](#_Toc28611)

# 实验一

## 目的与要求

题目：用JavaSocket编程开发聊天室

实践目的或任务：通过指导学生上机实践，对JavaSocket编程、Java多线程、Java图形用户界面进行掌握。

实践基本要求**：**

1.了解实验目的及实验原理；

2.编写程序，并附上程序代码和结果图；

3.总结在编程过程中遇到的问题、解决办法和收获。

实践的内容或要求**：**

1. 用Java图形用户界面编写聊天室服务器端和客户端，支持多个客户端连接到一个服务器。每个客户端能够输入账号。

2. 可以实现群聊（聊天记录显示在所有客户端界面）。

3. 完成好友列表在各个客户端上显示。

4. 可以实现 私人聊天，用户可以选择某个其他用户，单独发送信息。

5. 服务器能够群发系统消息，能够强行让某些用户下线。

6. 客户端的上线下线要求能够在其他客户端上面实时刷新。

## 操作环境

硬件：



软件：

操作系统：64位，Windows10家庭中文版

编辑器：visual studio code

java版本：java12.0.2

## 基本技术原理

### JavaSocket原理：

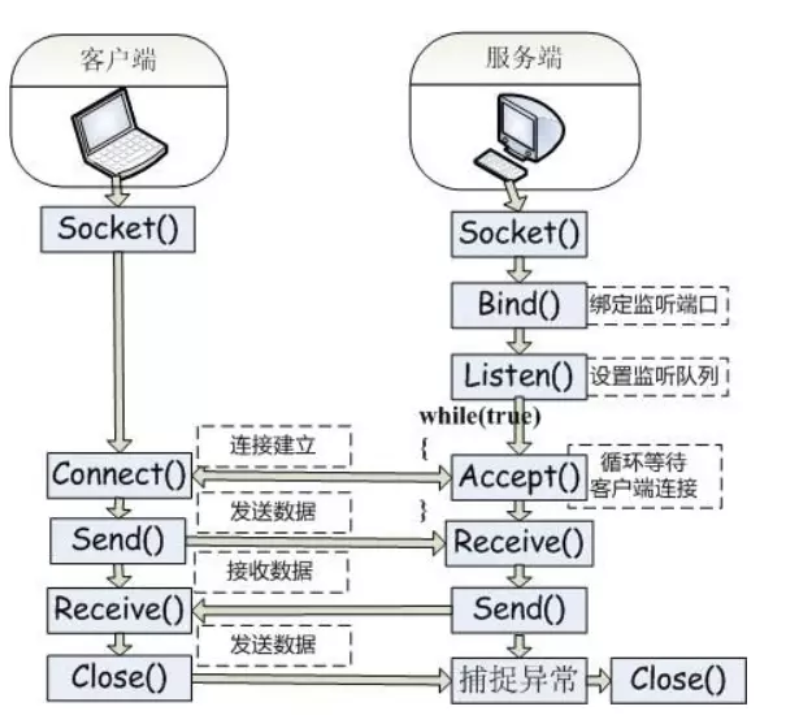
  TCP是（Tranfer Control Protocol）的简称，是一种面向连接的保证可靠传输的协议。通过TCP协议传输，得到的是一个顺序的无差错的数据流。发送方和接收方的成对的两个socket之间必须建立连接，当一个socket（通常都是server socket）等待建立连接时，另一个socket可以要求进行连接，一旦这两个socket连接起来，它们就可以进行双向数据传输，双方都可以进行发送或接收操作。

  Socket本质是编程接口(API)，对TCP/IP的封装，TCP/IP也要提供可供程序员做网络开发所用的接口，这就是Socket编程接口；HTTP是轿车，提供了封装或者显示数据的具体形式；Socket是发动机，提供了网络通信的能力。 Socket实质上提供了进程通信的端点。进程通信之前，双方首先必须各自创建一个端点，否则是没有办法建立联系并相互通信的。正如打电话之前，双方必须各自拥有一台电话机一样。套接字之间的连接过程可以分为三个步骤：服务器监听，客户端请求，连接确认。

 1、服务器监听：是服务器端套接字并不定位具体的客户端套接字，而是处于等待连接的状态，实时监控网络状态。

 2、客户端请求：是指由客户端的套接字提出连接请求，要连接的目标是服务器端的套接字。为此，客户端的套接字必须首先描述它要连接的服务器的套接字，指出服务器端套接字的地址和端口号，然后就向服务器端套接字提出连接请求。

 3、连接确认：是指当服务器端套接字监听到或者说接收到客户端套接字的连接请求，它就响应客户端套接字的请求，建立一个新的线程，把服务器端套接字的描述发给客户端，一旦客户端确认了此描述，连接就建立好了。而服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。

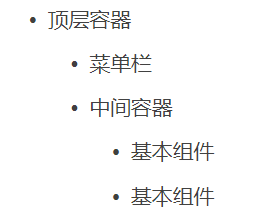


基于java的socket网络编程实现：Server端Listen监听某个端口是否有连接请求，Client端向Server 端发出连接请求，Server端向Client端发回Accept接受消息。这样一个连接就建立起来了。Server端和Client端都可以通过Send，Write等方法与对方通信。对于一个功能齐全的Socket，都要包含以下基本结构，其工作过程包含以下四个基本的步骤：1、创建Socket；2、 打开连接到Socket的输入/出流； 3、按照一定的协议对Socket进行读/写操作； 4、关闭Socket。

### Java Swing简介：

Swing 是 Java 为图形界面应用开发提供的一组工具包，是 Java 基础类的一部分。Swing 包含了构建图形界面（GUI）的各种组件，如: 窗口、标签、按钮、文本框等。Swing 提供了许多比 AWT 更好的屏幕显示元素，使用纯 Java 实现，能够更好的兼容跨平台运行。为了和 AWT 组件区分，Swing 组件在javax.swing.\*包下，类名均以 J 开头，例如: JFrame、JLabel、JButton等。

swing组件：一个 Java 的图形界面，由各种不同类型的“元素”组成，例如: 窗口、菜单栏、对话框、标签、按钮、文本框等等，这些“元素”统一被称为 组件（Component）。组件按照不同的功能，可分为 顶层容器、中间容器、基本组件。一个简单窗口的组成，如下层级结构所示:



组件类型的继承关系:

顶层容器 属于窗口类组件，继承自java.awt.Window；

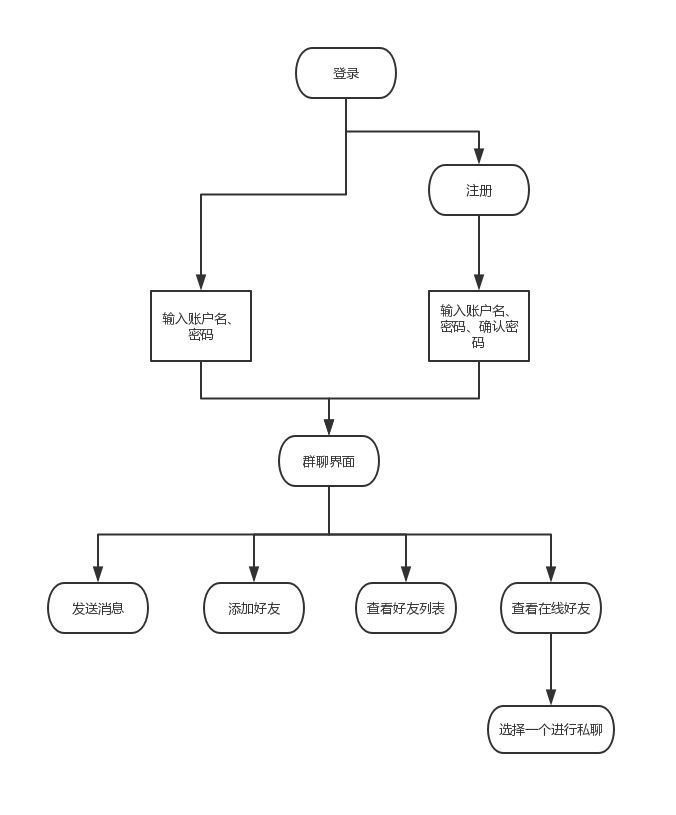
中间容器 和 基本组件 继承自javax.swing.JComponent。

顶层容器属于窗口类组件，可以独立显示，一个图形界面至少需要一个窗口。中间容器充当基本组件的载体，不可独立显示。中间容器可以添加若干基本组件（也可以嵌套添加中间容器），对容器内的组件进行管理，类似于给各种复杂的组件进行分组管理。最顶层的一个中间容器必须依托在顶层容器（窗口）内。基本组件是直接实现人机交互的组件。

布局管理器：把 Swing 的各种组件(JComponent)添加到面板容器中(JPanel)，需要给面板容器指定布局管理器(LayoutManager)，明确容器(Container)内的各个组件之间的排列布局方式。

## 概要设计

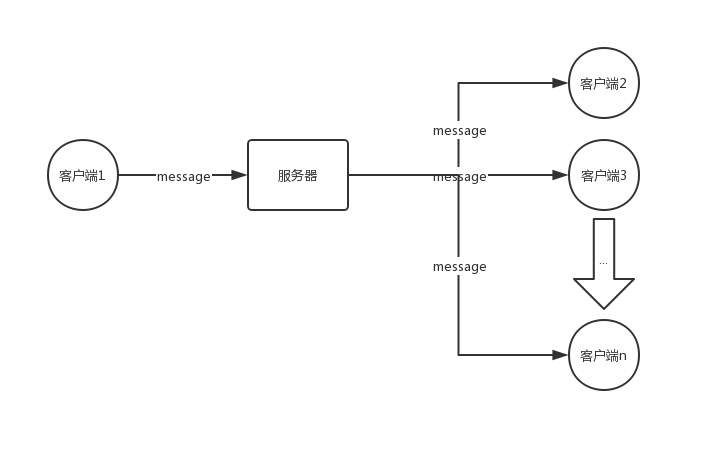
### 客户端功能流程图如下：



### 服务器功能流程图如下：

### qt_temp

### 群发消息流程：



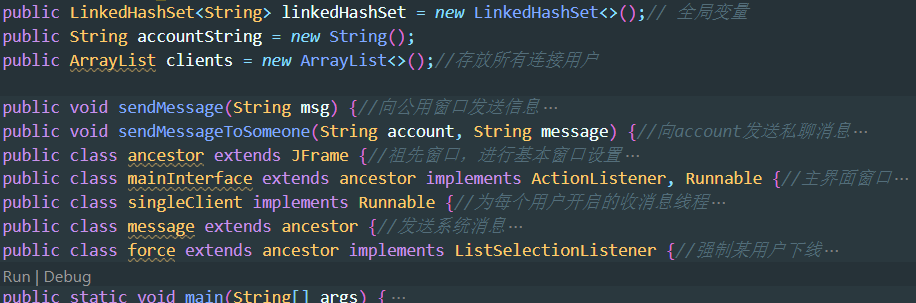
## 实验内容与实现

### chatRoom\_server.server.java

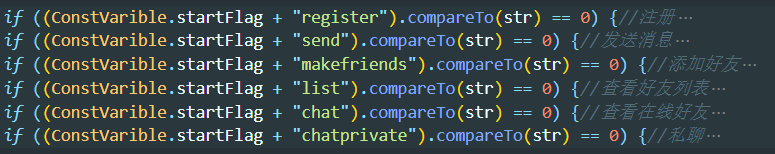
功能：服务器端功能实现

内容：

|  |  |
| --- | --- |
| 发送消息的函数 | sendMessage、sendMessageToSomeone |
| 界面的设置 | ancestor、mainInterface |
| 单个用户的服务线程 | singleClient |
| 系统消息发送类 | message |
| 强制用户下线类 | force |
| 启动服务器 | main |



其中单个用户服务线程通过判断 用户发送的请求消息 来启用相应功能

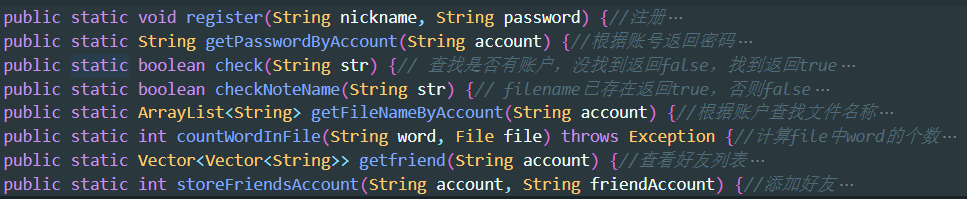


### chatRoom\_server.fileOperation.java

功能：文件操作

内容：

|  |  |
| --- | --- |
| 注册信息存储 | register |
| 查询文件信息 | getPasswordByAccount、check、checkNoteName、getFileNameByAccount |
| 计算文件中词的个数 | countWordInFile |
| 查看好友列表 | getfriend |
| 存储好友信息 | storeFriendsAccount |

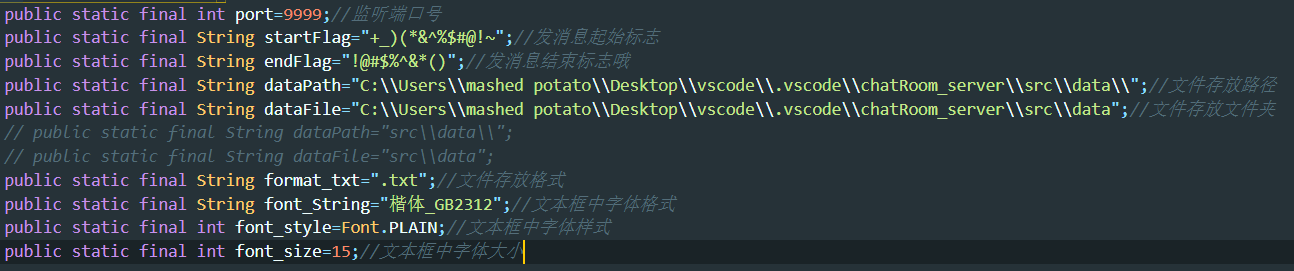


### chatRoom\_server.ConstVarible.java

功能：完成对常量的设置

|  |  |
| --- | --- |
| 端口号 | port |
| 发送消息标志 | startFlag、endFlag |
| 文件位置 | dataPath、dataFile |
| 文件格式 | format\_txt |
| 字体设置 | font\_String、font\_style、font\_size |

内容：

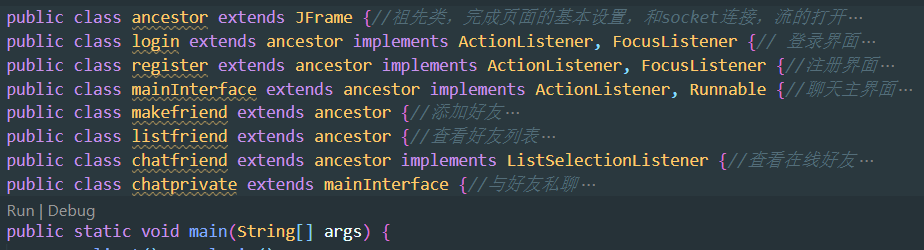


### charRoom\_client.client.java

功能：客户端功能实现

|  |  |
| --- | --- |
| 基础界面设置及socket连接 | ancestor |
| 登陆界面 | login |
| 注册界面 | register |
| 聊天主界面 | mainInterface |
| 添加好友 | makefriend |
| 好友列表 | listfriend |
| 在线好友 | chatfriend |
| 私聊好友 | chatprivate |
| 启动登录界面 | main |

内容：



### charRoom\_client.ConstVarible.java

功能：完成对客户端常量的设置

内容：

|  |  |
| --- | --- |
| 服务器网址 | address\_home |
| 端口号 | port |
| 发送消息标志 | startFlag+endFlag |
| 文本框字体设置 | font\_String+font\_style+font\_size |

### 

## 遇到的问题与解决方案

|  |  |
| --- | --- |
| 问题 | 解决方案 |
| 在试图使用本类实现ActionListener接口时，区分被按下的按钮 | 使用getText() |
| 取出JComboBox中的当前值 | JComboBox有一个 SelectedItem属性，所以使用getSelectedItem()就可以取到其中的选中值 |
| 使用Swing的JList的问题：  A：怎么把JList弄到界面上去  B：怎么往JList上放数据  C：怎么监听JList里的项的双击事件，然后取出数据  D：数据太多，怎么添加滚动条。 | A、B：把一个JList放在JPanel上，只要设置好JList的数据、大小，就可以直接用JPanel的add方法添加上去。  C：Propertiy.getSelectedValue()  Ｄ：这个也很简单，把JPanel换成JScrollPane即可，但是这时候不再调用add方法，而是调用setViewportView方法 |
| null布局（空布局）的使用 | 1)       首先利用setLayout(null)语句将容器的布局设置为null布局（空布局）。  2)       再调用组件的setBounds(int x, int y, int width,int height)方法设置组件在容器中的大小和位置，单位均为像素。  x为控件左边缘离窗体左边缘的距离  y为控件上边缘离窗体上边缘的距离  width为控件宽度  height为控件高度 |
| java中String值为空字符串与null的判断方法 | null表示的是一个对象的值，而不是一个字符串。例如声明一个对象的引用，String a=null。         “”表示的是一个空字符串，也就是说它的长度为0。例如声明一个字符串String s=”“。 |
| JTable行选取的事件触发如何实现 | JTabel是Swing里的表格组件,它的行选取事件, 不是由JTabel直接接受的,而是通过给SelectionModel 添加ListSelectionListener 来对选取事件作出反应. |
| java JTable中如何添加一行数据 | 使用DefaultTableModel的addRow(Vector rowData)方法。 |
| java.io.StreamCorruptedException: invalid stream header: 7371007E | 因为在客服端中建立一个对象ops 的时候默认只设置了一次头所以当我们第一次串对象的时候服务器解析是正确的，但是第二次传的时候ops是不会再加一次头传递过去而是直接传送的数据流了，但是在服务器端仍然是使用创建对象的方式来读取对象流（这是一个全新的对象）所以这就是为什么会抛出异常的原因，因为第二次写入输出流的代码中并不会重新去加入一个头。  因此在使用ObjectInputStream 和ObjectOutputStream时，一定要找准它们的对应关系，免得再犯这种很让人郁闷的错误。 |
| 当java swing界面启动无法正常显示，当调整JFrame大小后才能全部显示子组件 | 可以在JFrame对象上调用validate（）方法 |
| setLocationRelativeTo(null)不起作用 | 对于一个新的JFrame来讲，基本可以认为是0x0大小的，你在填充内容前调用这个函数，等于是把窗口的左上角放在了屏幕中央。然后你后面再添加了别的子组件之后pack或者setSize,JFrame窗口变大，但是左上角还在屏幕中央。 |
| java Swing组件随着窗口拖动等比移动或等比放大 | 1清空布局（使用绝对布局）  2添加监听器（监听窗口是否被拖动）  3在监听器里面动态调整 组件的位置 |
| 怎样从一个JPanel得到它所在的JFrame的引用 | JFrame frame=(JFrame)pane.getTopLevelAncestor(); |
| 利用JOptionPane.showConfirmDialog的返回值 | JOptionPane.YES\_OPTION JOptionPane.CANCEL\_OPTION JOptionPane.CLOSED\_OPTION JOptionPane.NO\_OPTION |
| java socket通信时，中文字符乱码的一个解决方法 | in = new BufferedReader(new InputStreamReader(socket.getInputStream(),"UTF-8")); out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(),"UTF-8")); |

## 实验结果

## 

### 聊天主界面：

### 

### 好友功能菜单：

### 

### 添加好友：

### 

### 查看好友列表：

### 

### 私聊窗口：

### 

### 服务器界面：

## 

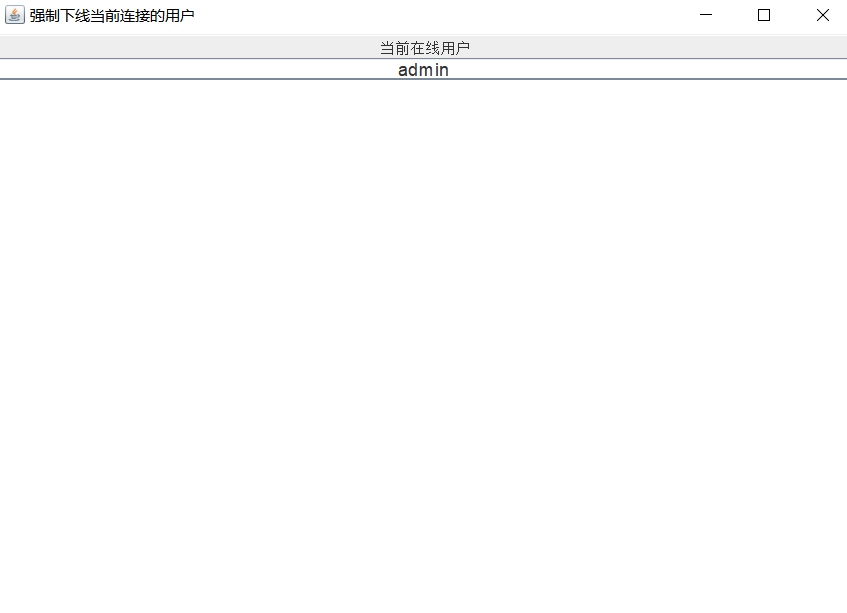
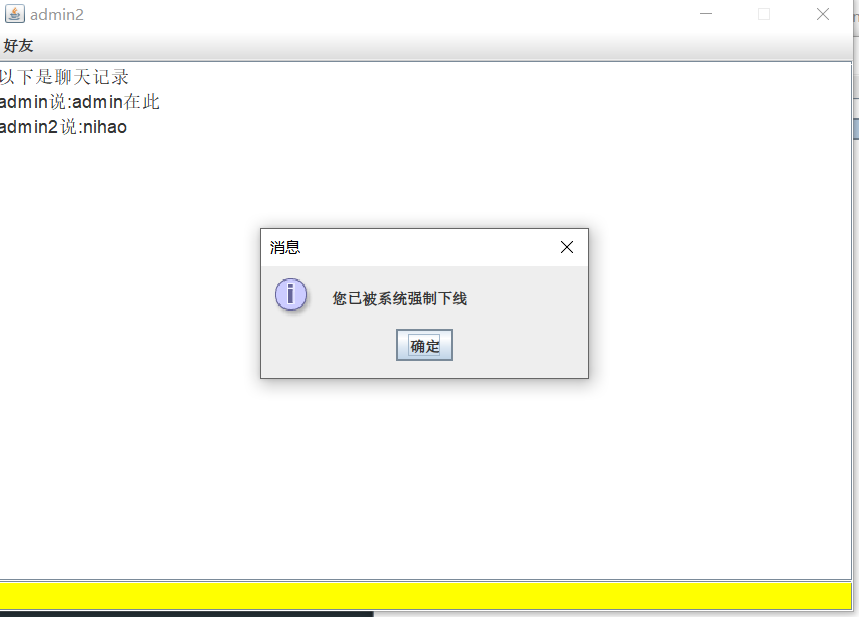
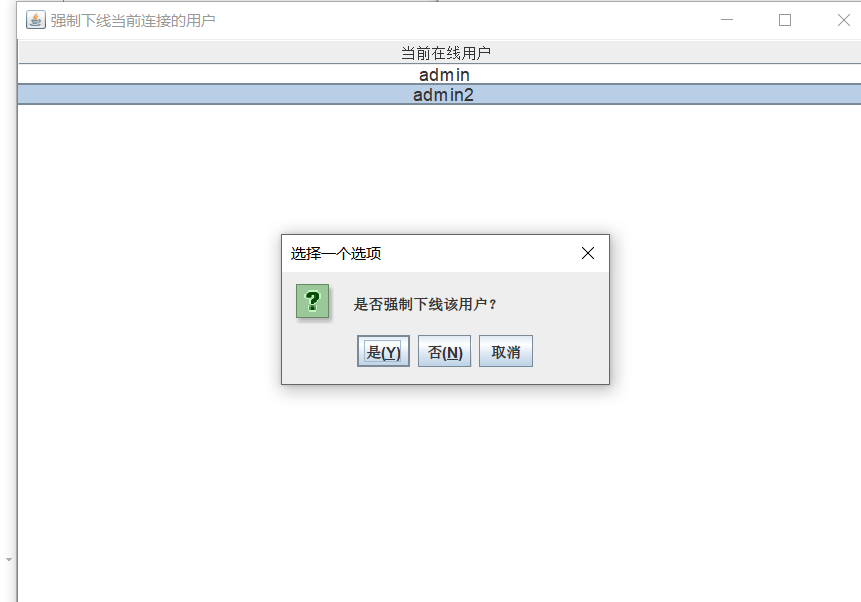
### 系统消息：

### 

### 管理当前连接用户：

### 

### 强制下线某当前连接用户：



## 八、实验源代码

chatRoom\_server.server.java

package server;

import java.awt.event.\*;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.BorderLayout;

import java.awt.Font;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.ObjectOutputStream;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.SocketException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.util.Vector;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.table.DefaultTableCellRenderer;

public class server {

    public LinkedHashSet<String> linkedHashSet = *new* LinkedHashSet<>();*// 全局变量*

    public String accountString = *new* String();

    public ArrayList clients = *new* ArrayList<>();*//存放所有连接用户*

    public void sendMessage(String msg) {*//向公用窗口发送信息*

*for* (int i = 0; i < clients.size(); i++) {

            singleClient sc = (singleClient) clients.get(i);

*if* (sc.ID.compareTo("public") == 0) {

                sc.ps.println(msg);

            }

        }

    }

    public void sendMessageToSomeone(String account, String message) {*//向account发送私聊消息*

*for* (int i = 0; i < clients.size(); i++) {

            singleClient sc = (singleClient) clients.get(i);

*if* (sc.account.compareTo(account) == 0 && sc.ID.compareTo("private") == 0) {

                sc.ps.println(message);

            }

        }

    }

    public class ancestor extends JFrame {*//祖先窗口，进行基本窗口设置*

        protected JPanel jPanel = *new* JPanel();

        protected ServerSocket ss = null;

        protected Socket s = null;

        protected String title = null;

        public ancestor(String title) {

            super(title);

            this.title = title;

            this.setVisible(true);

            this.setLocationRelativeTo(null);

            this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        }

    }

    public class mainInterface extends ancestor implements ActionListener, Runnable {*//主界面窗口*

        private JButton sendButton = *new* JButton("系统消息");

        private JButton force = *new* JButton("管理当前连接的用户");

        public mainInterface() {

            super("服务器");

            jPanel.add(sendButton);

            jPanel.add(force);

            this.add(jPanel);

            this.setSize(400, 250);

            jPanel.setLayout(null);

            sendButton.setBounds(135, 30, 100, 50);

            force.setBounds(90, 120, 200, 60);

            this.setLocationRelativeTo(null);

            this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

            sendButton.addActionListener(this);

            force.addActionListener(this);

*try* {

                ss = *new* ServerSocket(ConstVarible.port);

*new* Thread(this).start();*//多线程开启*

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

        public void run() {

*try* {

*while* (true) {*//多线程不断接受用户连接请求*

                    s = ss.accept();

                    singleClient sc = new singleClient(s);

*new* Thread(sc).start();

                    clients.add(sc);*//将连接用户存入clients*

                }

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

        public void actionPerformed(ActionEvent ae) {

            JButton jButton = (JButton) ae.getSource();

*if* (jButton.getText().compareTo("系统消息") == 0) {

*new* message(s);

            } *else* *if* (jButton.getText().compareTo("管理当前连接的用户") == 0) {

*new* force();

            }

        }

    }

    public class singleClient implements Runnable {*//为每个用户开启的收消息线程*

        private Socket s = null;

        private BufferedReader br = null;

        public PrintStream ps = null;

        public String account = "account";

        protected String ID = "public";

        public singleClient(Socket s) {

            this.s = s;

*try* {

                br = *new* BufferedReader(*new* InputStreamReader(s.getInputStream()));

                ps = *new* PrintStream(s.getOutputStream());

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

        public void run() {

*try* {

                ID = br.readLine();*//先读入ID和account*

                account=br.readLine();

*while* (true) {

                    String str = *new* String();

*try* {

                        str = br.readLine();

                    } *catch* (SocketException ex) {

*if* (ex.getMessage().compareTo("Connection reset") == 0) {

                            linkedHashSet.remove(accountString);*// 断开连接抛出java.net.SocketException:*

*// Connectionreset，移除该用户连接*

                            sendMessage("系统消息：" + accountString + "下线");

*break*;

                        }

                    }

*if* ((ConstVarible.startFlag + "login").compareTo(str) == 0) {*//登录*

*// account = br.readLine();*

                        accountString = account;

                        String password = fileOperation.getPasswordByAccount(account);

*// System.out.println(password);*

                        ps.println(password);

                        linkedHashSet.add(account);

                        sendMessage("系统消息：" + account + "上线");

                    }

*if* ((ConstVarible.startFlag + "register").compareTo(str) == 0) {*//注册*

*// account = br.readLine();*

                        accountString = account;

                        String password = br.readLine();

*if* (fileOperation.check(account) == false) {

                            fileOperation.register(account, password);

                            ps.println("success");

                        } *else* {

                            ps.println("failed");

                        }

                        linkedHashSet.add(account);

                        sendMessage("系统消息：" + account + "上线");

                    }

*if* ((ConstVarible.startFlag + "send").compareTo(str) == 0) {*//发送消息*

*try* {

*while* (true) {

                                String string = br.readLine();

                                sendMessage(string);

                                br.readLine();*// 读掉一个ID*

                                br.readLine();*//读掉account*

                                br.readLine();*// 读掉起始符*

                            }

                        } *catch* (Exception ex) {

                            ex.printStackTrace();

                        }

                    }

*if* ((ConstVarible.startFlag + "makefriends").compareTo(str) == 0) {*//添加好友*

*// account = br.readLine();*

                        String friendAccount = br.readLine();

                        int res = fileOperation.storeFriendsAccount(account, friendAccount);

*// System.out.println("hh");*

*if* (res == 0) {

                            ps.println("该好友账户不存在");

                        } *else* *if* (res == 1) {

                            ps.println("您已经添加过该好友");

                        } *else* *if* (res == 2) {

                            ps.println("添加成功");

                        } *else* *if* (res == 3) {

                            ps.println("添加失败");

                        }

                    }

*if* ((ConstVarible.startFlag + "list").compareTo(str) == 0) {*//查看好友列表*

*// account = br.readLine();*

                        ObjectOutputStream objectOutputStream = *new* ObjectOutputStream(s.getOutputStream());

                        objectOutputStream.writeObject(fileOperation.getfriend(account));

                    }

*if* ((ConstVarible.startFlag + "chat").compareTo(str) == 0) {*//查看在线好友*

*// account = br.readLine();*

                        Vector<Vector<String>> friends = fileOperation.getfriend(account);

*for* (int i = 0; i < friends.size(); i++) {

                            String string = friends.get(i).get(0);

                            Iterator<String> iterator = linkedHashSet.iterator();

*while* (iterator.hasNext()) {

*if* (string.compareTo(iterator.next()) == 0) {

                                    ps.println(string);

*// System.out.println(string);*

                                }

                            }

                        }

                        ps.println(ConstVarible.endFlag);

                    }

*if* ((ConstVarible.startFlag + "chatprivate").compareTo(str) == 0) {*//私聊*

*// account = br.readLine();*

                        String friendAccount = br.readLine();*// 好友账户*

*// for (int i = 0; i < clients.size(); i++) {// 给好友发送通知*

*// if (friendAccount.compareTo(((singleClient) clients.get(i)).account) == 0) {*

*// ((singleClient) clients.get(i)).ps.println(ConstVarible.startFlag +*

*// "chatprivate");*

*// ((singleClient) clients.get(i)).ps.println(account);*

*// // System.out.println(((singleClient) clients.get(i)).account);*

*// System.out.println(i);*

*// }*

*// }*

                        sendMessage("系统消息："+account+"向"+friendAccount+"发起私聊");

*try* {

*while* (true) {

                                String string = br.readLine();

*if* (string == null)

*break*;

                                sendMessageToSomeone(friendAccount, string);

                                sendMessageToSomeone(account, string);

*// br.readLine();// 读掉起始符*

                            }

                        } *catch* (Exception ex) {

                            ex.printStackTrace();

                        }

                    }

                }

            } *catch* (Exception ex) {

                ex.printStackTrace();

            } *finally* {

*try* {

                    br.close();

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

            }

        }

    }

    public class message extends ancestor {*//发送系统消息*

        protected JTextArea taMsg = *new* JTextArea("以下是系统消息\n");

        protected JTextField tfMsg = *new* JTextField();

        protected JScrollPane newJScrollPane = *new* JScrollPane(taMsg);

        protected ActionListener sendListener = null;

        public message(Socket s) {

            super("系统消息");

            BorderLayout borderLayout = *new* BorderLayout();

            jPanel.setLayout(borderLayout);

            jPanel.add(newJScrollPane, BorderLayout.CENTER);

            jPanel.add(tfMsg, BorderLayout.SOUTH);

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            taMsg.setFont(font);

            tfMsg.setFont(font);

            taMsg.setEditable(false);

            tfMsg.setBackground(Color.yellow);

            sendListener = *new* ActionListener() {

                public void actionPerformed(ActionEvent ae) {

*try* {

                        sendMessage("系统消息：" + tfMsg.getText());

                        taMsg.append("系统消息：" + tfMsg.getText() + "\n");

                        tfMsg.setText("");

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

            };

            tfMsg.addActionListener(sendListener);

            this.add(jPanel);

            this.setSize(700, 500);

            this.setLocationRelativeTo(null);

            this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        }

    }

    public class force extends ancestor implements ListSelectionListener {*//强制某用户下线*

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        protected Vector<Vector<String>> data = *new* Vector<>();

        protected Vector<String> columnName = *new* Vector<>();

        public force() {

            super("强制下线当前连接的用户");

            columnName.add("当前在线用户");

            Iterator<String> iterator = linkedHashSet.iterator();

*while* (iterator.hasNext()) {

                Vector<String> row = *new* Vector<>();

                row.add(iterator.next());

                data.add(row);

            }

            BorderLayout borderLayout = *new* BorderLayout();

            jPanel.setLayout(borderLayout);

            jTable = *new* JTable(data, columnName) {

                public boolean isCellEditable(int row, int column) {

*return* false;

                }

            };

            jTable.setPreferredScrollableViewportSize(*new* Dimension(600, 450));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.setFillsViewportHeight(true);

            DefaultTableCellRenderer defaultTableCellRenderer = *new* DefaultTableCellRenderer();

            defaultTableCellRenderer.setHorizontalAlignment(JLabel.CENTER);

            jTable.setDefaultRenderer(Object.class, defaultTableCellRenderer);

            jTable.getSelectionModel().addListSelectionListener(this);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane, BorderLayout.CENTER);

            this.add(jPanel);

            this.setSize(700, 500);

            this.setLocationRelativeTo(null);

        }

        public void valueChanged(ListSelectionEvent le) {

*if* (le.getValueIsAdjusting() == false) {

                int row = jTable.getSelectedRow();

                int option = JOptionPane.showConfirmDialog(this, "是否强制下线该用户？");

*if* (option == JOptionPane.YES\_OPTION) {

*try* {

*for* (int i = 0; i < clients.size(); i++) {

                            singleClient sc = (singleClient) clients.get(i);

*if* (sc.account.compareTo(data.get(row).get(0)) == 0) {

*// System.out.println(i);*

                                sc.ps.println(ConstVarible.startFlag + "exit");

                            }

                        }

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

            }

        }

    }

    public static void main(String[] args) {

*new* server().new mainInterface();

    }

}

chatRoom\_server.fileOperation.java

package server;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintStream;

import java.util.ArrayList;

import java.util.Properties;

import java.util.Set;

import java.util.TreeMap;

import java.util.Vector;

import javax.swing.JOptionPane;

public class fileOperation {

    private static Properties pps;

    static {*//pps初始化*

        pps = *new* Properties();

        FileReader reader = null;

*try* {

            File file = *new* File(ConstVarible.dataPath + "account.inc");

*if* (file.exists() == false) {

                file.createNewFile();

            }

            reader = *new* FileReader(file);

            pps.load(reader);

        } *catch* (Exception ex) {

            JOptionPane.showMessageDialog(null, "文件操作异常1");

            System.exit(0);

        } *finally* {

*try* {

                reader.close();

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

    }

    public static void register(String nickname, String password) {*//注册*

        pps.setProperty(nickname, password);

        PrintStream ps = null;

*try* {

            ps = *new* PrintStream(ConstVarible.dataPath + "account.inc");

            pps.list(ps);

        } *catch* (Exception ex) {

            JOptionPane.showMessageDialog(null, "文件操作异常2");

            System.exit(0);

        } *finally* {

*try* {

                ps.close();

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

    }

    public static String getPasswordByAccount(String account) {*//根据账号返回密码*

        String password = pps.getProperty(account);

*return* password;

    }

    public static boolean check(String str) {*// 查找是否有账户，没找到返回false，找到返回true*

*if* (pps.getProperty(str) == null) {

*return* false;

        } *else* {

*return* true;

        }

    }

    public static boolean checkNoteName(String str) {*// filename已存在返回true，否则false*

        File file = *new* File(ConstVarible.dataPath + str);

*// System.out.println(ConstVarible.dataPath + str);*

*if* (file.exists() == true) {

*return* true;

        } *else* {

*return* false;

        }

    }

    public static ArrayList<String> getFileNameByAccount(String account) {*//根据账户查找文件名称*

        File file = *new* File(ConstVarible.dataFile);

        File[] files = file.listFiles();

        ArrayList<String> arrayList = *new* ArrayList<String>();

*for* (File f *:* files) {

            String filename = f.getAbsolutePath();

*// System.out.println(filename);*

*if* (filename.startsWith(ConstVarible.dataPath + account + "\_")) {

                arrayList.add(filename);

            }

        }

*return* arrayList;

    }

    public static int countWordInFile(String word, File file) throws Exception {*//计算file中word的个数*

        FileInputStream fis = *new* FileInputStream(file);

        byte[] data = *new* byte[(int) file.length()];

        fis.read(data);

        fis.close();

        String msg = *new* String(data);

        String aft = msg.replace(word, "");

*return* (msg.length() - aft.length()) / word.length();

    }

    public static Vector<Vector<String>> getfriend(String account) {*//查看好友列表*

        File file = *new* File(ConstVarible.dataPath+account);

*if*(file.exists()==false)*return* null;

        Vector<Vector<String>> data = *new* Vector<>();

*try*{

            FileReader fileReader=*new* FileReader(file);

            BufferedReader bufferedReader=*new* BufferedReader(fileReader);

*while*(true){

                String string=bufferedReader.readLine();

*// System.out.println(string);*

*if*(string==null)*break*;

                Vector<String> row=*new* Vector<>();

                row.add(string);

                data.add(row);

            }

            bufferedReader.close();

        }*catch*(Exception ex){

            ex.printStackTrace();

        }

*return* data;

    }

    public static int storeFriendsAccount(String account, String friendAccount) {*//添加好友*

*if* (check(friendAccount) == false) {*// 先检查好友账户是否存在*

*return* 0;

        } *else* {

            File file = *new* File(ConstVarible.dataPath + account);

*try* {

*if* (file.exists() == false) {*//检查文件是否存在*

                    file.createNewFile();

                }

*if*(countWordInFile(friendAccount, file)==1){*//检查是否已经添加过该好友*

*return* 1;

                }

                FileWriter fw=*new* FileWriter(file,true);

                fw.append(friendAccount+"\n");

                fw.close();

                storeFriendsAccount(friendAccount, account);*//互为好友*

*return* 2;

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

*return* 3;

        }

    }

}

chatRoom\_server.ConstVarible.java

package server;

import java.awt.Font;

public class ConstVarible {

    public static final int port=9999;*//监听端口号*

    public static final String startFlag="+\_)(\*&^%$#@!~";*//发消息起始标志*

    public static final String endFlag="!@#$%^&\*()";*//发消息结束标志哦*

    public static final String dataPath="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\chatRoom\_server\\src\\data\\";*//文件存放路径*

    public static final String dataFile="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\chatRoom\_server\\src\\data";*//文件存放文件夹*

*// public static final String dataPath="src\\data\\";*

*// public static final String dataFile="src\\data";*

    public static final String format\_txt=".txt";*//文件存放格式*

    public static final String font\_String="楷体\_GB2312";*//文本框中字体格式*

    public static final int font\_style=Font.PLAIN;*//文本框中字体样式*

    public static final int font\_size=15;*//文本框中字体大小*

}

chatRoom\_client.client.java

package client;

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.ObjectInputStream;

import java.io.OutputStream;

import java.io.PrintStream;

import java.net.Socket;

import java.util.Vector;

import java.awt.event.\*;

import java.awt.\*;

import java.awt.Font;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.table.DefaultTableCellRenderer;

public class client {

    public class ancestor extends JFrame {

        protected JPanel jPanel = *new* JPanel();

        protected Socket s = null;

        protected OutputStream os = null;

        protected PrintStream ps = null;

        protected InputStream is = null;

        protected BufferedReader br = null;

        protected String account = null;

        protected String ID="ID";

        public ancestor(String str) {

            super(str);

            account = str;

            this.setTitle(str);

            this.setVisible(true);

            this.setLocationRelativeTo(null);

            this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

            this.setResizable(false);

*try* {

                s = *new* Socket(ConstVarible.address\_home\_wifi, ConstVarible.port);

                os = s.getOutputStream();

                ps=*new* PrintStream(os,true,"UTF-8");

*// ps = new PrintStream(os);*

                is = s.getInputStream();

                br = *new* BufferedReader(*new* InputStreamReader(is,"UTF-8"));

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

    }

    public class login extends ancestor implements ActionListener, FocusListener {*// 登录界面*

        private GridLayout gridLayout = *new* GridLayout(3, 2, 20, 20);

        private JLabel label\_name = *new* JLabel("账号");

        private JLabel label\_password = *new* JLabel("密码");

        private JTextField input\_name = *new* JTextField("输入账号");

        private JPasswordField input\_password = *new* JPasswordField("输入密码");

        private JButton registerButton = *new* JButton("去注册");

        private JButton loginButton = *new* JButton("登录");

        public login() {

            super("登录界面");

            jPanel.setLayout(gridLayout);

            jPanel.add(label\_name);

            jPanel.add(input\_name);

            jPanel.add(label\_password);

            jPanel.add(input\_password);

            jPanel.add(loginButton);

            jPanel.add(registerButton);

            this.add(jPanel);

            this.setSize(400, 300);

            this.setLocationRelativeTo(null);

            registerButton.addActionListener(this);

            loginButton.addActionListener(this);

            input\_name.addFocusListener(this);

            input\_password.addFocusListener(this);

            input\_password.addActionListener(this);

        }

        public void actionPerformed(ActionEvent e) {

            JButton jButton = *new* JButton("jButton");

*try* {

                jButton = (JButton) e.getSource();

            } *catch* (Exception ex) {

                log();

*return*;

            }

*if* (jButton.getText().compareTo("去注册") == 0) {

*new* client().new register();

*// this.dispose();*

                this.setVisible(false);

            } *else* {

                log();

            }

        }

        public void log() {

            String realPassword = null;

*try* {

                ps.println(ID);

                ps.println(input\_name.getText());

                ps.println(ConstVarible.startFlag + "login");

*// ps.println(input\_name.getText());*

                realPassword = br.readLine();

            } *catch* (Exception ex) {

                ex.printStackTrace();

            } *finally* {

*try* {

                    br.close();

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

            }

*if* (realPassword.compareTo(String.valueOf(input\_password.getPassword())) == 0) {

                JOptionPane.showMessageDialog(null, "登录成功");

*new* client().new mainInterface(input\_name.getText());

*// this.dispose();*

                this.setVisible(false);

            } *else* {

                JOptionPane.showMessageDialog(null, "账号不存在或密码错误");

*new* client().new login();

*// this.dispose();*

                this.setVisible(false);

            }

        }

        public void focusGained(FocusEvent fe) {

*if* (input\_name.isFocusOwner() == true) {

                input\_name.setText("");

            }

*if* (input\_password.isFocusOwner() == true) {

                input\_password.setText("");

            }

        }

        public void focusLost(FocusEvent fe) {

*if* (input\_name.isFocusOwner() == false) {

*// input\_name.setText("输入账号");*

            }

*if* (input\_password.isFocusOwner() == false) {

*// input\_password.setText("输入密码");*

            }

        }

    }

    public class register extends ancestor implements ActionListener, FocusListener {

        private GridLayout gridLayout = *new* GridLayout(4, 2, 20, 20);

        private JLabel label\_name = *new* JLabel("账号");

        private JLabel label\_password = *new* JLabel("密码");

        private JLabel label\_password\_again = *new* JLabel("确认密码");

        private JTextField input\_name = *new* JTextField("输入账号");

        private JPasswordField input\_password = *new* JPasswordField("输入密码");

        private JPasswordField input\_password\_again = *new* JPasswordField("重复密码");

        private JButton registerButton = *new* JButton("注册");

        private JButton cancelButton = *new* JButton("取消");

        public register() {

            super("注册界面");

            input\_name.addFocusListener(this);

            input\_password.addFocusListener(this);

            input\_password\_again.addFocusListener(this);

            jPanel.setLayout(gridLayout);

            jPanel.add(label\_name);

            jPanel.add(input\_name);

            jPanel.add(label\_password);

            jPanel.add(input\_password);

            jPanel.add(label\_password\_again);

            jPanel.add(input\_password\_again);

            jPanel.add(registerButton);

            jPanel.add(cancelButton);

            this.add(jPanel);

            this.setSize(500, 400);

            this.setLocationRelativeTo(null);

            registerButton.addActionListener(this);

            cancelButton.addActionListener(this);

        }

        public void actionPerformed(ActionEvent e) {

            JButton jButton = (JButton) e.getSource();

*if* (jButton.getText().compareTo("注册") == 0) {

*if* (String.valueOf(input\_password.getPassword())

                        .compareTo(String.valueOf(input\_password\_again.getPassword())) == 0) {

*try* {

                        os = s.getOutputStream();

                        ps = *new* PrintStream(os,true,"UTF-8");

                        ps.println(ID);

                        ps.println(input\_name.getText());

                        ps.println(ConstVarible.startFlag + "register");

*// ps.println(input\_name.getText());*

                        ps.println(String.valueOf(input\_password.getPassword()));

                        is = s.getInputStream();

                        br = *new* BufferedReader(*new* InputStreamReader(is,"UTF-8"));

*if* (br.readLine().compareTo("success") == 0) {

                            JOptionPane.showMessageDialog(this, "注册成功");

*new* client().new mainInterface(input\_name.getText());

                            this.dispose();

                        } *else* {

                            JOptionPane.showMessageDialog(this, "此账号已存在");

*new* client().new register();

                            this.dispose();

                        }

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    } *finally* {

*try* {

                            br.close();

                        } *catch* (Exception ex) {

                            ex.printStackTrace();

                        }

                    }

                } *else* {

                    JOptionPane.showMessageDialog(null, "密码与确认密码不一致，请重新输入");

                }

            } *else* *if* (jButton.getText().compareTo("取消") == 0) {

*new* client().new login();

                this.dispose();

            }

        }

        public void focusGained(FocusEvent fe) {

*if* (input\_name.isFocusOwner() == true) {

                input\_name.setText("");

            }

*if* (input\_password.isFocusOwner() == true) {

                input\_password.setText("");

            }

*if* (input\_password\_again.isFocusOwner() == true) {

                input\_password\_again.setText("");

            }

        }

        public void focusLost(FocusEvent fe) {

        }

    }

    public class mainInterface extends ancestor implements ActionListener, Runnable {

        protected JTextArea taMsg = *new* JTextArea("以下是聊天记录\n");

        protected JTextField tfMsg = *new* JTextField();

        protected JScrollPane newJScrollPane = *new* JScrollPane(taMsg);

        protected JMenuBar jMenuBar = *new* JMenuBar();

        protected JMenu friends = *new* JMenu("好友");

        protected JMenuItem makefriends = *new* JMenuItem("添加好友");

        protected JMenuItem listfriends = *new* JMenuItem("好友列表");

        protected JMenuItem chatfriends = *new* JMenuItem("私聊好友");

        protected ActionListener sendListener = null;

        public mainInterface(String str) {

            super(str);

            account = str;

            ID="public";

            BorderLayout borderLayout = *new* BorderLayout();

            jPanel.setLayout(borderLayout);

            jPanel.add(newJScrollPane, BorderLayout.CENTER);

            jPanel.add(tfMsg, BorderLayout.SOUTH);

            jPanel.add(jMenuBar, BorderLayout.NORTH);

            jMenuBar.add(friends);

            friends.add(makefriends);

            friends.add(listfriends);

            friends.add(chatfriends);

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            taMsg.setFont(font);

            tfMsg.setFont(font);

            taMsg.setEditable(false);

            tfMsg.setBackground(Color.yellow);

            sendListener = *new* ActionListener() {

                public void actionPerformed(ActionEvent ae) {

                    ps.println(ID);

                    ps.println(account);

                    ps.println(ConstVarible.startFlag + "send");

*try* {

                        ps.println(account + "说:" + tfMsg.getText());

                        tfMsg.setText("");

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

            };

            tfMsg.addActionListener(sendListener);

            makefriends.addActionListener(this);

            listfriends.addActionListener(this);

            chatfriends.addActionListener(this);

            this.add(jPanel);

            this.setSize(700, 500);

            this.setLocationRelativeTo(null);

            this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

*new* Thread(this).start();

        }

        public void run() {

*try* {

*while* (true) {

                    String str = br.readLine();

*if*(str.compareTo(ConstVarible.startFlag+"exit")==0){

                        JOptionPane.showMessageDialog(this, "您已被系统强制下线");

                        System.exit(-1);

                    }

                    taMsg.append(str + "\n");

                }

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

        public void actionPerformed(ActionEvent ae) {

            JMenuItem jMenuItem = (JMenuItem) ae.getSource();

*if* (jMenuItem.getText().compareTo("添加好友") == 0) {

*new* makefriend(account);

            } *else* *if* (jMenuItem.getText().compareTo("私聊好友") == 0) {

*new* chatfriend(account);

            } *else* *if* (jMenuItem.getText().compareTo("好友列表") == 0) {

*new* listfriend(account);

            }

        }

    }

    public class makefriend extends ancestor {

        public makefriend(String account) {

            super("交友");

            this.setVisible(false);

            String friendAccount = JOptionPane.showInputDialog(this, "请输入好友账号");

*if* (friendAccount == null || friendAccount.compareTo("") == 0) {

*return*;

            } *else* {

                ps.println(ID);

                ps.println(account);

                ps.println(ConstVarible.startFlag + "makefriends");

*// ps.println(account);*

                ps.println(friendAccount);

*// JOptionPane.showMessageDialog(this, "hh");*

*try* {

*// System.out.println("hh");*

                    JOptionPane.showMessageDialog(this, br.readLine());

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

            }

        }

    }

    public class listfriend extends ancestor {

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        protected Vector<Vector<String>> data = *new* Vector<>();

        protected Vector<String> columnName = *new* Vector<>();

        public listfriend(String account) {

            super(account);

            this.setTitle("好友列表");

            ps.println(ID);

            ps.println(account);

            ps.println(ConstVarible.startFlag + "list");

            ObjectInputStream objectInputStream = null;

*try* {

                objectInputStream = *new* ObjectInputStream(is);

                data = (Vector<Vector<String>>) objectInputStream.readObject();

            } *catch* (Exception ex) {

                System.out.println("objecetInputStream");

                ex.printStackTrace();

            }

            FlowLayout flowLayout = *new* FlowLayout();

            jPanel.setLayout(flowLayout);

            columnName.add("好友列表");

            jTable = *new* JTable(data, columnName) {

                public boolean isCellEditable(int row, int column) {

*return* false;

                }

            };

            jTable.setPreferredScrollableViewportSize(*new* Dimension(600, 450));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.setFillsViewportHeight(true);

            DefaultTableCellRenderer defaultTableCellRenderer = *new* DefaultTableCellRenderer();

            defaultTableCellRenderer.setHorizontalAlignment(JLabel.CENTER);

            jTable.setDefaultRenderer(Object.class, defaultTableCellRenderer);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane);

            this.add(jPanel);

            this.setSize(700, 500);

            this.setLocationRelativeTo(null);

        }

    }

    public class chatfriend extends ancestor implements ListSelectionListener {

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        protected Vector<Vector<String>> data = *new* Vector<>();

        protected Vector<String> columnName = *new* Vector<>();

        protected JMenuBar jMenuBar = *new* JMenuBar();

        protected JMenu jMenu = *new* JMenu("刷新");

        protected JMenuItem refreshfriends = *new* JMenuItem("刷新列表");

        protected ActionListener refresh = null;

        public chatfriend(String account) {

            super(account);

            this.setTitle("选择私聊好友");

            ps.println(ID);

            ps.println(account);

            ps.println(ConstVarible.startFlag + "chat");

*// 获取表格数据*

            columnName.add("在线好友");

*try* {

*while* (true) {

                    String string = br.readLine();

*// System.out.println(string);*

*if* (string.compareTo(ConstVarible.endFlag) == 0) {

*break*;

                    }

                    Vector<String> row = *new* Vector<>();

                    row.add(string);

                    data.add(row);

                }

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

*// 界面*

            BorderLayout borderLayout = *new* BorderLayout();

            jPanel.setLayout(borderLayout);

            jPanel.add(jMenuBar, BorderLayout.NORTH);

            jMenuBar.add(jMenu);

            jMenu.add(refreshfriends);

*// 监听器*

            refresh = *new* ActionListener() {

                public void actionPerformed(ActionEvent ae) {

*new* chatfriend(account);

                    jPanel.getTopLevelAncestor().setVisible(false);

                }

            };

            refreshfriends.addActionListener(refresh);

            jTable = *new* JTable(data, columnName) {

                public boolean isCellEditable(int row, int column) {

*return* false;

                }

            };

            jTable.setPreferredScrollableViewportSize(*new* Dimension(600, 450));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.setFillsViewportHeight(true);

            DefaultTableCellRenderer defaultTableCellRenderer = *new* DefaultTableCellRenderer();

            defaultTableCellRenderer.setHorizontalAlignment(JLabel.CENTER);

            jTable.setDefaultRenderer(Object.class, defaultTableCellRenderer);

            jTable.getSelectionModel().addListSelectionListener(this);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane, BorderLayout.CENTER);

            this.add(jPanel);

            this.setSize(700, 500);

            this.setLocationRelativeTo(null);

        }

        public void valueChanged(ListSelectionEvent le) {

            int row = jTable.getSelectedRow();

*if* (le.getValueIsAdjusting() == false) {

*new* chatprivate(account, data.get(row).get(0));

            }

        }

    }

    public class chatprivate extends mainInterface {

        public chatprivate(String account, String friendAccount) {

            super(account);

            ID="private";

            ps.println(ID);

            ps.println(account);

            ps.println(ConstVarible.startFlag + "chatprivate");

            ps.println(friendAccount);*// 好友账户*

            this.setTitle(account + "与" + friendAccount + "的私聊");

            tfMsg.removeActionListener(sendListener);

            tfMsg.addActionListener(*new* ActionListener() {

                public void actionPerformed(ActionEvent ae) {

*try* {

                        ps.println(account + "说:" + tfMsg.getText());

                        tfMsg.setText("");

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

            });

        }

    }

    public static void main(String[] args) {

*new* client().new login();

    }

}

chatRoom\_client.ConstVarible.java

package client;

import java.awt.Font;

public class ConstVarible {

*// public static final String address\_home\_wifi="192.168.0.100";*

    public static final String address\_home\_wifi="192.168.1.2";*//云服务器*

    public static final int port=9999;

    public static final String startFlag="+\_)(\*&^%$#@!~";

    public static final String endFlag="!@#$%^&\*()";

    public static final String font\_String=null;

    public static final int font\_style=Font.PLAIN;

    public static final int font\_size=15;

}

## 九、实验总结

我学到了有关JAVA等方面的知识，了解了关于软件开发的流程。了解了自己的不足，知道了自己努力的方向。 我很珍惜这次的程序设计实践，因为它能够让我了解自己的不足以 及以后自己努力的方向，同时也能让我了解软件开发的流程，增加一点软件开发经验和社会经验，增加就业机会，

在做这次程序设计之前，我有许多不足：不能全面系统的、深入的掌握某种技术，也就是会的挺多，但都是皮毛，不能满足就业的需要。 没有任何实际的开发经验，完全是想象中学习，考试还行，一到实际开发和应用就歇菜了。

通过这次的实训，我学到了很多：首先，对JAVA识比以前有了更深的了解。其次，在这次的实训中我的动手操作能力有了一点的提高。总的说来，自己还是有一定的收获的。因为自己的动手操纵能力得到了提高。最后是团队协作。在整个项目完成过程中团队协作有着不可替代的作用。从在刚拿到项目时对项目的分析到最后的项目完结的都有一定的体现。刚拿到项目时，我们团队进行了分析，并分配了各自的任务。当我们其中一人遇到问题的时候，另一人会去帮忙，效率提升了很多。  
 在这次的实训当中我收获颇丰。但仅仅靠这点学习还是远远不够的。所以在以后的学习中我会更加努力，提高自己的能力。

## 十、参考资料

《用Java语言描述面向对象》 郭克华

<https://www.cnblogs.com/tongxupeng/p/10259547.html>

<https://bbs.csdn.net/topics/350035312/>

<https://blog.csdn.net/vv___/article/details/78447027>

<https://blog.csdn.net/weixin_36571185/article/details/71616952>

<https://blog.csdn.net/qq_36761831/article/details/81413415>

<https://blog.csdn.net/qq_36761831/article/details/81460476>

<https://blog.csdn.net/qq_33860226/article/details/51489786>

<https://blog.csdn.net/liujun13579/article/details/7774267>

<https://blog.csdn.net/bingwen0210/article/details/6529160?utm_source=blogxgwz4>

<https://blog.csdn.net/zlb_lover/article/details/79006840>

<https://zhidao.baidu.com/question/686974790273912452.html>

<https://blog.csdn.net/ukeychen/article/details/1726779>

<https://zhidao.baidu.com/question/263232359.html>

<https://blog.csdn.net/qq_38341596/article/details/78800646>

<https://www.cnblogs.com/iucforever/p/4828603.html>

<http://blog.sina.com.cn/s/blog_7ea6850b0100qlpx.html#commonComment>

<https://blog.csdn.net/gwqlzu/article/details/10426517>

<https://blog.csdn.net/qq_19533277/article/details/83039659>

<https://blog.csdn.net/caiandyong/article/details/53791784>

<http://www.myexception.cn/j2se/390939.html>

<https://www.cnblogs.com/easyidea/p/11026032.html>

<https://bbs.csdn.net/topics/40009106>

<https://blog.csdn.net/qq_40948862/article/details/90085185>

<https://blog.csdn.net/mecho/article/details/7342654>