

**Java核心编程技术课程设计**



**题 目： 多人在线聊天室系统**

**学生姓名： 蒋骏 张瑞凯 姜鹏**

**学 号： 170312149 170312210 170312209**

**指导教师： 俞文昌**

**完成时间：** **2019 年 12 月 29 日**

**大数据与人工智能学院**

目录

[一、 需求分析与总体设计 1](#_Toc4100)

[1.1需求分析 1](#_Toc5340)

[1.2功能图 1](#_Toc17321)

[1.2.1客户端登陆流程图 1](#_Toc25009)

[1.2.2客户端注册流程图 2](#_Toc8448)

[二、 软件设计 2](#_Toc3284)

[2.1设计思想 2](#_Toc10817)

[2.2设计内容 2](#_Toc21675)

[三、 软件重要功能块代码说明 3](#_Toc7436)

[3.1客户端 3](#_Toc15866)

[3.1.1 chatView类 3](#_Toc14893)

[3.1.2 Client类 4](#_Toc274)

[3.1.3 ClienFileThread类 9](#_Toc12442)

[3.1.4 Login类 11](#_Toc12623)

[3.1.5 RegisterView类 12](#_Toc28023)

[3.2服务端 13](#_Toc31127)

[3.2.1 Backstage类 13](#_Toc24865)

[3.2.2 Serve类 14](#_Toc23991)

[3.2.3 ServeFileOutPut类 15](#_Toc2438)

[3.2.ServeFileThread类 16](#_Toc12247)

[3.2.5 ServeOutPut类 17](#_Toc21594)

[3.3UserBD类 18](#_Toc8794)

[四、 软件使用说明 20](#_Toc16367)

[五、 总结 26](#_Toc25056)

[5.1实验问题 26](#_Toc6063)

[5.2实验体会 26](#_Toc16459)

[六、 参考文献 27](#_Toc20878)

# 需求分析与总体设计

## 1.1需求分析

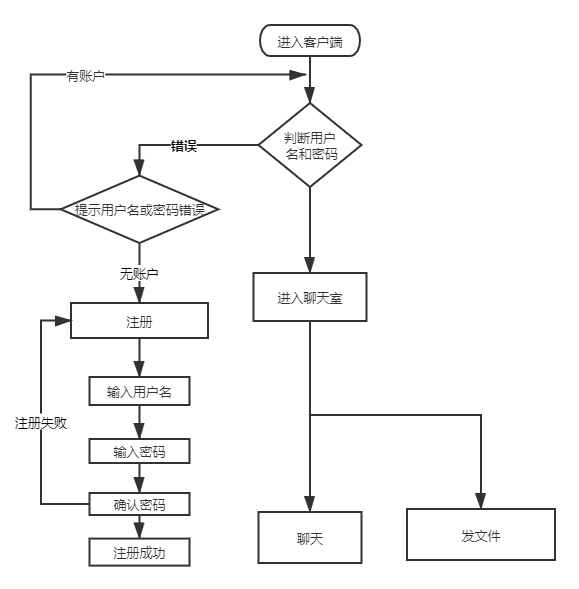
随着互联网的迅速普及，以及网络技术的不断发展，人们通过网络的交流方式不断发展。网络聊天室就是其中的一种。聊天室的系统即使交流的方式满足了网络中多人同时交流的需要。

本设计所完成的网络聊天室具有常用聊天室的所有功能。包括多人在线的同时聊天，两人间的私密聊天。在聊天的时候可以发送文件，还能显示在线用户列表，满足了不同人群的交流方式。

## 1.2功能图

### 1.2.1客户端登陆流程图

客户端登陆流程如图1-1

图1-1

### 1.2.2客户端注册流程图

客户端注册流程如图1-2

图1-2

# 软件设计

## 2.1设计思想

本程序设计的思想是基于cs模式的聊天室。

依据需求分析的结果，该系统应设计成一个可以实现多人同时在线聊天功能的聊天软件，下面从软件的整体结构设计、界面设计、变量设计等方面阐述一下系统的设计内容。依据需求分析结果，设计应实现多人同时在线聊天，就应该分为服务端和客户端两个端口。用户端把信息发送到服务端，而后服务端再把信息分发到各个客户端。

## 2.2设计内容

服务端：设计服务端界面，开启文件传输线程，等待客户端连接请求，对每个客户端开启文本传输线程。

客户端：设计登陆界面、注册界面、聊天室界面，开启文本传输线程、开启文件传输连接请求。

数据库类：建立数据库连接，提供增查操作。

# 软件重要功能块代码说明

## 3.1客户端

3.1.1 chatView类

聊天室界面类，画出一个聊天室界面，“发送”按钮监听发送的消息，“打开文件”按钮监听，会开启一个默认的文件选择器。

/\*\*

\* 聊天室界面类

\*/

public class chatView {

String userName; //用户名

Client.ChatViewListen listener; //聊天界面监听

// 构造方法：得到用户名

public chatView(String userName) {

this.userName = userName ;

init();

}

// 初始化

void init() {

JFrame frame = new JFrame("客户端");

frame.setBounds(500,200,400,330); //设置坐标和大小

frame.setResizable(false); // 缩放为不能缩放

JPanel panel1 = new JPanel();

JLabel lable = new JLabel("用户：" + userName);

//聊天文本域

JTextArea textArea = new JTextArea("\*\*\*\*\*\*\*\*\*\*\*\*用户："+userName+" 登录成功，这里你可以畅所欲言！\*\*\*\*\*\*\*\*\*\*\*\n",12, 35);

textArea.setEditable(false); // 设置为不可修改

JScrollPane scroll = new JScrollPane(textArea); // 设置滚动面板（装入textArea） scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS); // 显示垂直条

panel1.add(lable);

panel1.add(scroll);

JTextField text = new JTextField(20);//发送文本区

JButton but = new JButton("发送");

JButton openFileBut = new JButton("传输文件");

panel1.add(text);

panel1.add(but);

panel1.add(openFileBut);

// 设置“打开文件”按钮监听

openFileBut.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// showFileOpenDialog(frame);

// 创建一个默认的文件选择器

JFileChooser fileChooser = new JFileChooser();

// 设置默认显示的文件夹

fileChooser.setCurrentDirectory(new File("C:/Users/admin/Desktop"));

// 设置默认使用的文件过滤器（FileNameExtensionFilter 的第一个参数是描述, 后面是需要过滤的文件扩展名 可变参数）

fileChooser.setFileFilter(new FileNameExtensionFilter("(txt)", "txt"));

int result = fileChooser.showOpenDialog(frame); // 对话框将会尽量显示在靠近 parent 的中心

// 点击确定

if(result == JFileChooser.APPROVE\_OPTION) {

// 获取路径

File file = fileChooser.getSelectedFile(); //获取文件对象

String path = file.getAbsolutePath(); //获取文件绝对路径

ClientFileThread.outFileToServer(path); //传输文件线程

}

}});

// 设置“发送”按钮监听

listener = new Client().new ChatViewListen(); //new一个文本发送监听类

listener.setJTextField(text); // 获取消息文本域

listener.setJTextArea(textArea); //获取公共聊天域

listener.setChatViewJf(frame); //获取框架

text.addActionListener(listener); // 文本框添加监听：可以直接回车发送消息

but.addActionListener(listener); // 按钮添加监听

frame.add(panel1);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 设置右上角关闭图标的作用

frame.setVisible(true); // 设置可见

}

}

3.1.2 Client类

含有main方法，new一个登陆界面，并包含注册监听、登陆监听、聊天界面监听四个内部类。

public class Client extends Thread {

static Socket mySocket = null; // 一定要加上static，否则新建线程时会清空

static JTextField textInput; //用于输入文本框

static JTextArea textShow; //用于输出的文本域

static JFrame chatViewJFrame; //聊天室面板

private static BufferedReader in = null; //读

private static PrintWriter out = null; //写

private static String userName; //用户名

public static void main(String[] args) { //开启登陆界面

new Login();

}

// 接收从服务端发送来的消息

public void run() {

try {

in = new BufferedReader(new InputStreamReader(mySocket.getInputStream())); // 输入字符流

while (true) {

String str = in.readLine(); // 获取服务端发送的信息

textShow.append(str + '\n'); // 添加进聊天客户端的文本区域

textShow.setCaretPosition(textShow.getDocument().getLength()); // 设置滚动条始终在最下面

}

} catch (Exception e) {}

}

/\*\*

\* 完成注册监听类

\*/

class RegisteredListen implements ActionListener{

JTextField textField; //用户名域

JPasswordField pwdField1; //密码域

JPasswordField pwdField2; //确认密码域

JFrame RegisterFrame; // 注册窗口

Statement statement; //SQL语句操作对象

public void setJTextField(JTextField textField) { //得到用户名

this.textField = textField;

}

public void setJPasswordField1(JPasswordField pwdField1) { //密码

this.pwdField1 = pwdField1;

}

public void setJPasswordField2(JPasswordField pwdField2) { //确认密码

this.pwdField2 = pwdField2;

}

public void setJFrame(JFrame jFrame) { //得到框架

this.RegisterFrame = jFrame;

}

@Override

public void actionPerformed(ActionEvent e) {

userName = textField.getText(); //得到用户名

String userPwd1 = String.valueOf(pwdField1.getPassword()); // getPassword方法获得char数组，再转换成String

String userPwd2 = String.valueOf(pwdField2.getPassword()); // getPassword方法获得char数组，再转换成String

try {

statement = new UserDB().getStatement(); //得到statement对象

}

catch (Exception exception){

JOptionPane.showMessageDialog(RegisterFrame, exception.toString(), "提示", JOptionPane.WARNING\_MESSAGE);

exception.printStackTrace();

}

if(UserDB.check(statement,userName)){ //如果用户名已经注册了

JOptionPane.showMessageDialog(RegisterFrame, "该用户名已经被注册，请重新输入！", "提示", JOptionPane.WARNING\_MESSAGE);

}

else if (!userPwd1.equals(userPwd2)){ //如果两次密码输入不一致

JOptionPane.showMessageDialog(RegisterFrame, "两次输入密码不一致，请重新输入！",

"提示", JOptionPane.WARNING\_MESSAGE);

}

else {

UserDB.update(statement,userName,userPwd1); //添加该用户

JOptionPane.showMessageDialog(RegisterFrame, "恭喜你！注册成功", "注册成功", JOptionPane.WARNING\_MESSAGE);

RegisterFrame.setVisible(false);

}

}

}

/\*\*

\* 登陆监听类

\*/

class LoginListen implements ActionListener {

JTextField textField;

JPasswordField pwdField;

JFrame loginJFrame; // 登录窗口本身

chatView chatView = null;

Statement statement; //SQL语句操作对象

public void setJTextField(JTextField textField) { //获得发送文本域

this.textField = textField;

}

public void setJPasswordField(JPasswordField pwdField) { //获得公共文本域

this.pwdField = pwdField;

}

public void setJFrame(JFrame jFrame) { //获得框架

this.loginJFrame = jFrame;

}

public void actionPerformed(ActionEvent event) {

userName = textField.getText(); //得到用户名

String userPwd = String.valueOf(pwdField.getPassword()); // getPassword方法获得char数组，再转换成String

try {

statement = new UserDB().getStatement(); //得到statement对象

} catch (Exception e) {

e.printStackTrace();

}

if(UserDB.verify(statement,userName,userPwd)) { // 如果用户名密码在user表中

chatView = new chatView(userName); // 新建聊天窗口,设置聊天窗口的用户名（静态）

// 建立和服务器的联系

try {

InetAddress addr = InetAddress.getByName(null); // 获取主机地址

mySocket = new Socket(addr,9700); // 客户端套接字

loginJFrame.setVisible(false); // 隐藏登录窗口

out = new PrintWriter(mySocket.getOutputStream()); // 输出流

out.println("用户【" + userName + "】进入聊天室！"); // 发送用户名给服务器

out.flush(); // 清空缓冲区out中的数据

} catch (IOException e) {

e.printStackTrace();

}

// 新建普通读写线程并启动

start();

// 新建文件读写线程并启动

ClientFileThread fileThread = new ClientFileThread(userName, chatViewJFrame, out);

fileThread.start();

}

else if(!UserDB.check(statement,userName)){ //如果没有找到用户名

JOptionPane.showMessageDialog(loginJFrame, "用户名:"+userName+"未被注册!", "提示", JOptionPane.WARNING\_MESSAGE);

}

else {

JOptionPane.showMessageDialog(loginJFrame, "账号或密码错误，请重新输入！", "提示", JOptionPane.WARNING\_MESSAGE);

}

}

}

/\*\*

\* 聊天界面监听类

\*/

class ChatViewListen implements ActionListener{

public void setJTextField(JTextField text) {

textInput = text; // 得到发送域

}

public void setJTextArea(JTextArea textArea) {

textShow = textArea; // 放在外部类，因为其它地方也要用到

}

public void setChatViewJf(JFrame jFrame) {

chatViewJFrame = jFrame; // 放在外部类，因为其它地方也要用到

// 设置关闭聊天界面的监听

chatViewJFrame.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

out.println("用户【" + userName + "】离开聊天室！");

out.flush();

System.exit(0);

}

});

}

// 监听执行函数

public void actionPerformed(ActionEvent event) {

try {

String str = textInput.getText();

// 文本框内容为空

if("".equals(str)) {

textInput.grabFocus(); // 设置焦点

// 弹出消息对话框（警告消息）

JOptionPane.showMessageDialog(chatViewJFrame, "输入为空，请重新输入！", "提示", JOptionPane.WARNING\_MESSAGE);

return;

}

out.println(userName + "说：" + str); // 输出给服务端

out.flush(); // 清空缓冲区out中的数据

textInput.setText(""); // 清空文本框

textInput.grabFocus(); // 设置焦点：一开始光标就在发送域

} catch (Exception e) {}

}

}

/\*\*

\* 注册监听类

\*/

class RegisterListen implements ActionListener{

RegisterView registerView = null;

@Override

public void actionPerformed(ActionEvent e) {

registerView = new RegisterView();

}

}

}

3.1.3 ClienFileThread类

客户端文件传输线程类，接收服务器传输过来的文件名，文件大小，再根据文件信息接收文件数据，同时，利用弹窗界面可以选择是否接收文件。

public class ClientFileThread extends Thread{

private Socket socket = null;

private JFrame chatViewJFrame = null; //聊天界面

static String userName = null;

static PrintWriter out = null; // 普通消息的发送（Server.java传来的值）

static DataInputStream fileIn = null; //接收文件名，文件大小

static DataOutputStream fileOut = null; //发送文件名，文件大小

static DataInputStream fileReader = null; //接收文件数据

static DataOutputStream fileWriter = null; //发送文件数据

public ClientFileThread(String userName, JFrame chatViewJFrame, PrintWriter out) {

ClientFileThread.userName = userName;

this.chatViewJFrame = chatViewJFrame;

ClientFileThread.out = out;

}

// 客户端接收文件

public void run() {

try {

InetAddress addr = InetAddress.getByName(null); // 获取主机地址

socket = new Socket(addr, 8848); // 客户端套接字

fileIn = new DataInputStream(socket.getInputStream()); // 输入流

fileOut = new DataOutputStream(socket.getOutputStream()); // 输出流

// 接收文件

while(true) {

String textName = fileIn.readUTF(); //得到文件名

long totleLength = fileIn.readLong(); //得到文件长度

int result = JOptionPane.showConfirmDialog(chatViewJFrame, "是否接受？", "提示",

JOptionPane.YES\_NO\_OPTION);

int length = -1;

byte[] buff = new byte[1024];

long curLength = 0;

// 提示框选择结果，0为确定，1位取消

if(result == 0){

out.println("【" + userName + "选择了接收文件！】");

out.flush();

File userFile = new File("./" + userName); //文件放入当前目录

if(!userFile.exists()) { // 新建当前用户的文件夹

userFile.mkdir();

}

File file = new File("./" + userName + "\\"+ textName);

fileWriter = new DataOutputStream(new FileOutputStream(file)); //接收文件

while((length = fileIn.read(buff)) > 0) { // 把文件写进本地

fileWriter.write(buff, 0, length);

fileWriter.flush();

curLength += length;

if(curLength == totleLength) { // 接收完毕后结束

break;

}

}

out.println("【" + userName + "接收了文件！】");

out.flush();

// 提示文件存放地址

JOptionPane.showMessageDialog(chatViewJFrame, "文件存放地址：\n" +

file.getAbsolutePath() , "提示", JOptionPane.INFORMATION\_MESSAGE);

}

else { // 不接受文件的话，还是接收数据，但是不写入本地

while((length = fileIn.read(buff)) > 0) {

curLength += length;

if(curLength == totleLength) { // 强制结束

break;

}

}

}

// fileWriter.close();

}

} catch (Exception e) {}

}

// 客户端发送文件

static void outFileToServer(String path) {

try {

File file = new File(path); //得到文件路径

fileReader = new DataInputStream(new FileInputStream(file));

fileOut.writeUTF(file.getName()); // 发送文件名字

fileOut.flush();

fileOut.writeLong(file.length()); // 发送文件长度

fileOut.flush();

int length = -1;

byte[] buff = new byte[1024];

while ((length = fileReader.read(buff)) > 0) { // 发送内容

fileOut.write(buff, 0, length);

fileOut.flush();

}

out.println("【" + userName + "已成功发送文件！】");

out.flush();

} catch (Exception e) {}

}

}

### 3.1.4 Login类

登陆界面类，画出一个登陆界面，监听“登陆”按钮，用户名和密码合法则开启聊天室界面，并隐藏登陆界面，监听“注册按钮”，画出一个注册界面。

public class Login {

private JTextField textField = null; //用户名文本域

private JPasswordField pwdField = null; //密码文本域

Client.LoginListen loginListener=null; //登录按钮监听

Client.RegisterListen registerListen = null; //注册按钮监听

// 构造函数

public Login() {

init();

}

void init() {

JFrame frame = new JFrame("登录"); //框架

frame.setBounds(600, 300, 310, 210); //设置位置大小

frame.setResizable(false); // 设置不可改变大小

JPanel mainPanel = new JPanel(new BorderLayout()); // BorderLayout边框布局

JPanel panel1 = new JPanel(); //面板1

JLabel headJLabel = new JLabel("登录界面"); //登陆标签

headJLabel.setFont(new Font(null, 0, 35)); // 设置文本的字体类型、样式 和 大小

panel1.add(headJLabel);

JPanel panel2 = new JPanel(); //面板2

JLabel nameJLabel = new JLabel("用户名："); //用户名标签

textField = new JTextField(20);

JLabel pwdJLabel = new JLabel("密码： "); //密码标签

pwdField = new JPasswordField(20);

JButton loginButton = new JButton("登录"); //登录按钮

JButton registerButton = new JButton("注册"); // 注册按钮

panel2.add(nameJLabel);

panel2.add(textField);

panel2.add(pwdJLabel);

panel2.add(pwdField);

panel2.add(loginButton);

panel2.add(registerButton);

mainPanel.add(panel1, BorderLayout.NORTH);

mainPanel.add(panel2, BorderLayout.CENTER);

// 设置监听

loginListener = new Client().new LoginListen(); // 新建登陆按钮监听类

loginListener.setJTextField(textField); // 将用户名传入

loginListener.setJPasswordField(pwdField); //将密码传入

loginListener.setJFrame(frame); //将整个框架传入

pwdField.addActionListener(loginListener); //文本域监听，敲击回车触发事件

loginButton.addActionListener(loginListener); // 登陆按钮添加监听

registerListen = new Client().new RegisterListen(); // 新建注册按钮监听类

registerButton.addActionListener(registerListen); //注册按钮添加监听

frame.add(mainPanel);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 设置关闭图标作用

frame.setVisible(true); // 设置可见

}

}

### 3.1.5 RegisterView类

注册界面类，画出一个注册界面，监听“完成注册”按钮，用户名，密码和确认密码合法，在数据库chat的user表中插入该用户信息，然后提示用户注册成功，并隐藏该界面。

public class RegisterView {

private JTextField textField = null; //用户名文本域

private JPasswordField pwdField1 = null; //密码文本域

private JPasswordField pwdField2 = null; //重复密码文本域

Client.RegisteredListen listener =null; //完成注册按钮监听

// 构造函数

public RegisterView() {

init();

}

void init() {

JFrame frame = new JFrame("注册界面"); //框架

frame.setBounds(750, 350, 310, 310); //设置位置大小

JPanel mainPanel = new JPanel(new BorderLayout()); // BorderLayout边框布局

frame.setResizable(false); // 设置不可改变大小

JPanel panel1 = new JPanel(); //面板1

JLabel headJLabel = new JLabel("请完成注册"); //注册界面标签

headJLabel.setFont(new Font(null, 0, 35)); // 设置文本的字体类型、样式 和 大小

panel1.add(headJLabel);

JPanel panel2 = new JPanel(); //面板2

JLabel nameJLabel = new JLabel(" 请输入用户名："); //用户名标签

textField = new JTextField(20);

panel2.add(nameJLabel);

panel2.add(textField);

JLabel pwdJLabel1 = new JLabel(" 请输入密码： "); //密码标签

pwdField1 = new JPasswordField(20);

panel2.add(pwdJLabel1);

panel2.add(pwdField1);

pwdField2 = new JPasswordField(20);

JLabel pwdJLabel2 = new JLabel("请再次输入密码： "); //密码标签

panel2.add(pwdJLabel2);

panel2.add(pwdField2);

JButton registerButton = new JButton("完成注册"); // 完成注册按钮

panel2.add(registerButton);

mainPanel.add(panel1, BorderLayout.NORTH);

mainPanel.add(panel2, BorderLayout.CENTER);

// 设置监听

listener = new Client().new RegisteredListen(); // 新建注册监听类

listener.setJTextField(textField); // 将用户名传入

listener.setJPasswordField1(pwdField1); //将密码传入

listener.setJPasswordField2(pwdField2); //将二次确认密码传入

listener.setJFrame(frame); //将整个框架传入

pwdField2.addActionListener(listener); //敲击回车出发事件

registerButton.addActionListener(listener); // 登陆按钮添加监听

frame.add(mainPanel);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 设置关闭图标作用

frame.setVisible(true); // 设置可见

}

public static void main(String[] args) {

new RegisterView();

}

}

## 3.2服务端

### 3.2.1 Backstage类

服务器后台界面类，画一个服务器后台，提供setTextArea()方法，将发来的文本信息分发到所有建立连接的客户端公共文本域中，并添加时间。

public class Backstage {

private JTextArea textArea; //显示聊天室信息

// 用于向文本区域添加信息

void setTextArea(String str) {

String nowDate = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(new Date());//得到当前时间

textArea.append(nowDate+'\n'); //加时间

textArea.append(str+'\n'); // 加信息

textArea.setCaretPosition(textArea.getDocument().getLength()); // 设置滚动条在最下面

}

// 构造函数

public Backstage() {

init();

}

void init() {

JFrame frame = new JFrame("服务器端");

frame.setBounds(500,100,550,580); // 设置窗口坐标和大小

frame.setResizable(false); // 设置为不可缩放

JPanel mainPanel = new JPanel(); // 新建容器(边框布局)

JLabel lable = new JLabel("欢迎来到服务器端",SwingConstants.CENTER);

lable.setFont(new Font(null, 0, 24)); // 设置文本的字体类型、样式 和 大小

mainPanel.add(lable); //北

JPanel chatPanel = new JPanel(); //聊天监管面板

textArea = new JTextArea(25,35); // 新建文本区域并设置长宽

textArea.setEditable(false); // 设置为不可修改

JScrollPane textScroll = new JScrollPane(textArea); // 设置滚动面板（装入textArea） textScroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL\_SCROLLBAR\_ALWAYS); // 显示垂直条

chatPanel.add(textScroll); //添加

mainPanel.add(chatPanel);

frame.add(mainPanel);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 设置关闭图标作用

frame.setVisible(true); // 设置可见

}

}

### 3.2.2 Serve类

New一个Backstage对象，开启文件传输线程，对每个连接上的客户端开启文本传输线程。

public class Server{

static ServerSocket server = null; //服务器连接

static Socket client = null; //客户端连接

static List<Socket> list = new ArrayList<Socket>(); // 存储客户端对象

public static void main(String[] args) {

Backstage backstage = new Backstage(); // 创建聊天室后台系统界面

try {

// 在服务器端开启文件传输的线程

ServerFileThread sft = new ServerFileThread();

sft.start();

server = new ServerSocket(9700); // 服务器端套接字

// 等待连接并开启相应线程

while (true) {

client = server.accept(); // 等待连接

list.add(client); // 添加当前客户端到列表

// 在服务器端对客户端开启文本读取的线程

ServerOutput print = new ServerOutput(client,backstage);

print.start();

}

} catch (IOException e1) {

e1.printStackTrace(); // 出现异常则打印出异常的位置

}

}

}

### 3.2.3 ServeFileOutPut类

服务端文件传输类，收到文件后，分发给队列中的客户端。

public class ServerFileOutput extends Thread {

private Socket nowSocket = null; //当前套接字

private DataInputStream input = null; //输入流

private DataOutputStream output = null; //输出流

//构造函数

public ServerFileOutput(Socket socket) {

this.nowSocket = socket;

}

public void run() {

try {

input = new DataInputStream(nowSocket.getInputStream()); // 得到当前套接字的输入流

while (true) {

// 获取文件名字和文件长度

String textName = input.readUTF();

long textLength = input.readLong();

// 发送文件名字和文件长度给所有客户端

for(Socket socket: ServerFileThread.list) { //发送给阻塞队列中的所有用户

output = new DataOutputStream(socket.getOutputStream()); // 输出流

if(socket != nowSocket) { // 发送除了自己之外的其它客户端文件名和文件长度

output.writeUTF(textName);

output.flush();

output.writeLong(textLength);

output.flush();

}

}

// 发送文件内容

int length = -1;

long curLength = 0; //当前已经发送了多少

byte[] buff = new byte[1024]; //一次读入最多1024字节

while ((length = input.read(buff)) > 0) { //如果还没有传输完

curLength += length;

for(Socket socket: ServerFileThread.list) {

output = new DataOutputStream(socket.getOutputStream()); // 输出流

if(socket != nowSocket) { // 发送给其它客户端

output.write(buff, 0, length);

output.flush();

}

}

if(curLength == textLength) { // 如果已经发送完毕则退出

break;

}

}

}

} catch (Exception e) {

ServerFileThread.list.remove(nowSocket); // 线程关闭，移除相应套接字

}

}

}

### 3.2.ServeFileThread类

服务端文件传输线程类，监听端口8848，如果有文件传输任务，则开启文件传输线程。

public class ServerFileThread extends Thread{

ServerSocket server = null;

Socket socket = null;

static List<Socket> list = new ArrayList<Socket>(); // 存储客户端

public void run() {

try {

server = new ServerSocket(8848);

while(true) {

socket = server.accept(); //监听8848端口

list.add(socket); //存储客户端

// 开启文件传输线程

ServerFileOutput ft = new ServerFileOutput(socket); //开启一个服务器文件传输线程

ft.start();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

### 3.2.5 ServeOutPut类

服务端发送线程类，用于接收客户端的消息，并分发给阻塞队列中的其他客户端。

public class ServerOutput extends Thread{

Socket nowSocket = null; //得到的当前客户端进程

Backstage backstage = null; //得到后台界面

BufferedReader in =null; //输入流

PrintWriter out = null; //输出流

// 构造函数

public ServerOutput(Socket s, Backstage backstage) {

this.backstage = backstage; // 获取多人聊天系统界面

this.nowSocket = s; // 获取当前客户端

}

public void run() {

try {

in = new BufferedReader(new InputStreamReader(nowSocket.getInputStream())); // 接收每个发送过信息的客户端的数据

// 获取客户端信息并把信息发送给所有客户端

while (true) {

String str = in.readLine(); //按行读取

String nowDate = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(new Date());//得到当前时间

// 发送给所有客户端

for(Socket socket: Server.list) {

out = new PrintWriter(socket.getOutputStream()); // 对每个客户端新建相应的socket套接字

out.println(nowDate); //发送时间

if(socket == nowSocket) { // 如果是当前客户端

out.println("(你)" + str); //前缀加上(你)

}

else { // 发送给其它客户端不用加这个前缀

out.println(str);

}

out.flush(); // 刷新输出流

}

// 调用setTextArea方法将信息显示到图形界面

backstage.setTextArea(str);

}

} catch (IOException e) {

Server.list.remove(nowSocket); // 线程关闭，移除相应套接字

}

}

}

## 3.3UserBD类

建立数据库连接，提供增查操作。

public class UserDB {

Statement statement; //语句对象

public UserDB() throws SQLException{

Connection conn = getConnection(); //与数据库建立连接

this.statement = conn.createStatement(); //可以向数据库发送要执行的SQL语句

}

//得到statement对象

public Statement getStatement() {

return statement;

}

/\*\*

\* 与数据库建立连接

\*/

public static Connection getConnection() throws SQLException{

String drivers = "com.mysql.jdbc.Driver"; //驱动名称

String url = "jdbc:mysql://localhost:3306/chat?useUnicode=true&useSSL=false"; //关闭ssl连接

String username = "root";

String password = "147258369";

return DriverManager.getConnection(url, username, password);

}

/\*\*

\* 验证用户名和密码:如果正确返回true，否则false

\*/

public static boolean verify(Statement statement ,String name,String password){

String sql ="select \* from chat.user"; //SQL语句：查询user中所有项

boolean flag = false;

try {

ResultSet result = statement.executeQuery(sql); // 执行语句，返回一个结果集合

while(result.next()){

//遍历所有entry如果用户名和密码都一致返回true

if(result.getString(1).equals(name) && result.getString(2).equals(password)){

flag = true;

break;

}

}

}

catch (SQLException e){}

finally {

return flag;

}

}

/\*\*

\* 查询用户是否存在:如果查询到了就返回true,否则返回false

\*/

public static boolean check(Statement statement, String name){

boolean flag = false;

try{

String sql = "SELECT NAME FROM CHAT.USER WHERE NAME = '"+name+"'";

ResultSet resultSet = statement.executeQuery (sql);

while (resultSet.next()){

if (resultSet.getString(1).equals(name)){

flag = true;

break;

}

}

}

catch (Exception e){}

finally {

return flag;

}

}

/\*\*

\* 插入数据

\*/

public static void update(Statement statement,String name ,String password)

{

try{

String sql = "INSERT INTO CHAT.USER VALUES ('"+name+"','"+password+"')";

//执行更新操作

statement.executeUpdate(sql);

// System.out.println("插入成功");

}

catch (Exception e){}

}

}

# 软件使用说明

本程序需要保持服务端在线，在服务端被打开以后，用户端即可打开客户端登陆并进行多人聊天互动。当用户打开客户端后，会自动弹出一个登陆界面，登陆界面可以输入用户名

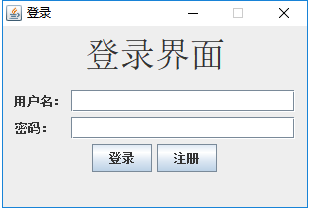


图4-1用户端登陆界面

和密码进行登陆，如果已有用户名，则可直接登陆，如果没有用户名，会提示该用户名未注册，此时需要进行注册，注册成功后登陆。

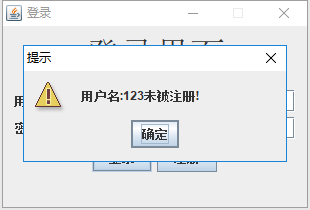


图4-2未注册提示

当用户打开注册界面时，会提示输入用户名，输入密码和确认密码，用户输入完后若提示用户名已被注册，则需更换用户名，完成注册；若提示密码输入错误，则需重新输入并确认密码，完成注册。

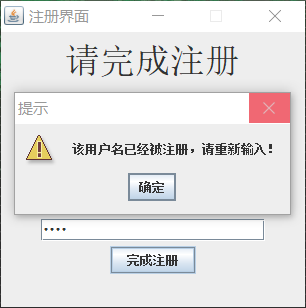


图4-3用户名已被注册

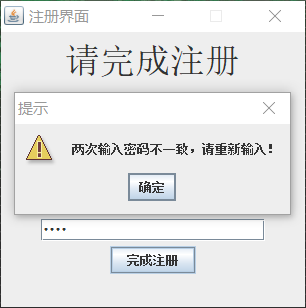


图4-4注册密码输入错误

当用户注册成功时，会提示注册成功，此时即可使用注册成功的用户名和密码进行登陆，若登陆失败，则会提示账号或密码输入不正确，用户需重新确认用户名及密码输入正确完成登陆。

![5~T7V)[T]($9[J{ZK~5Z`YN](data:image/png;base64,)

图4-5新用户注册成功

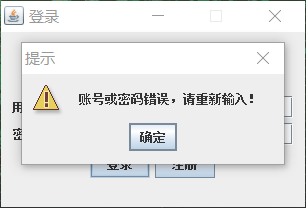


图4-6账号或密码输入错误

用户登陆成功后即可进入聊天室界面，聊天室界面会显示当前用户名与登陆时间，若其它用户登陆，则界面会显示用户名与登陆时间。

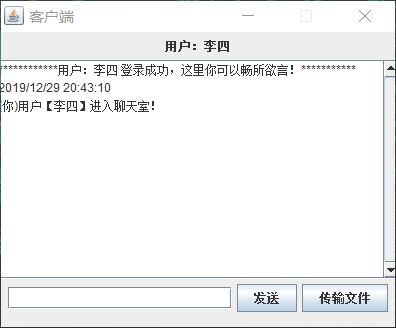


图4-7聊天室界面

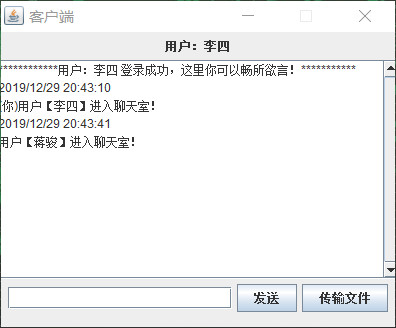


图4-8其它用户加入聊天室

聊天室主要分为两种功能，发送信息以及传输文件，用户可以在对话框内输入文本信息通过发送按钮与其它用户聊天，也可以通过传输文件按钮，将txt格式类型或任意类型文件发送给其它所有用户，其它用户可以选择接收或者拒绝，若接收则会将文件保存到程序当前

![`]]SVDHDHC}YS8](PUD@NXD](data:image/png;base64,)

图4-9用户聊天

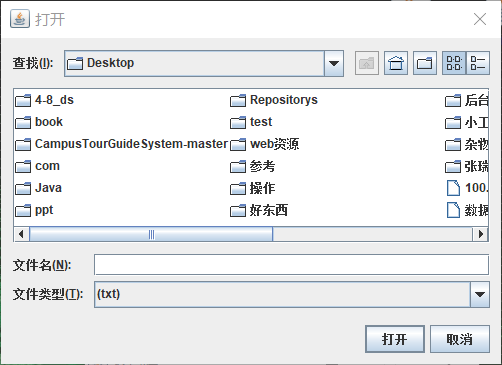


图4-10用户传输文件

目录并自动生成文件名为该用户名的文件，其它用户即可进行查看，当其它用户接收文件后可以在聊天室看到接收文件的用户名提示，若拒绝，则其它用户不会接收也不会提示。

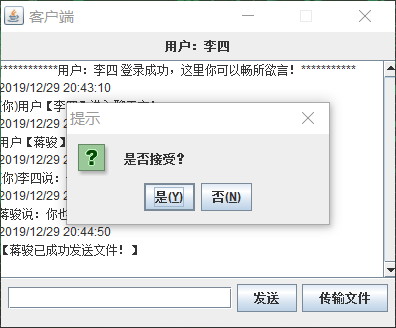


图4-11接收文件

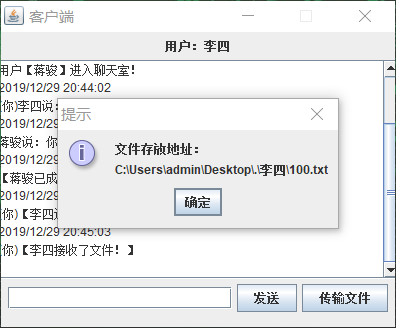


图4-12确认接收文件

当有用户退出聊天室时，其它所以用户可以看到退出的用户名及退出时间。

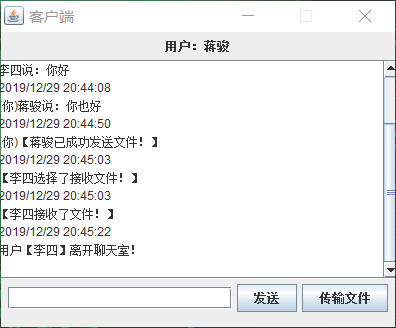


图4-13用户退出提示

# 总结

## 5.1实验问题

小组在设计时，对多线程考虑的不是很周全，发现问题后及时做了修改。总体的思路是基于c/s的模式，client借助socket完成向server的发送和接受两个工作，而发送和接受是需要开两个线程的，一个用户需要自己管理两个线程。server则相对比较复杂，因为这里面涉及到了调度，server需要有发送消息给在线client的线程，以及接受client发来的消息的线程，为了使得所有的用户消息是同步的，server需要管理一个所有用户线程的列表，从而实现用户的行为的控制，这就要求只要有用户请求连接服务器，服务器就要为用户新建一个线程，那么client和server之间需要建立socket连接。

## 5.2实验体会

本程序提供了聊天室最基本的通信功能，通过本次作业和小组的共同学习，小组成员受益匪浅，对Java编程中用到的IO、多线程、网络编程、图形用户界面与事件处理都有了更深的理解并在实际中得到应用。对于目前成熟的聊天软件来说，我们的聊天室未免有些简陋，后续我还会尝试更加完善系统功能，使系统能像QQ等即时通信软件一样发送表情和调节字体等，服务器端可以保存所有用户的通信信息，而客户端可以保存自己的通信信息。

# 参考文献

1. 耿祥义.Java课程设计.清华大学出版社第二版，2008年11月.
2. 马皓.Java语言程序设计.机械工业出版社，2007年3月
3. 孙卫琴.java 面向对象编程.电子工业出版社.2007

[4]刘海军. Java程序设计. 中国铁道出版社，2006

[5]郑阿奇，姜乃松，殷红先. Java实用教程. 电子工业出版社，2005

[6]张广彬、王小宁、高静.Java课程设计案例精编（第二版）清华大学出版社，2011年