Java-Socket聊天室

1. 实验目的

- 巩固复习Java的Swing知识，能够对JTextArea，JTextField等组件有进一步理解
- 学习Java的多线程编程，能够熟练写出多线程的程序，并且了解异常抛出的语法
- 学习Java的Socket编程的相关接口

2. 实验环境

Java版本：1.8.0_351

实验环境：Windows

3. 实验原理

3.1 实验设计

实现一个公共聊天室，聊天室内，每个用户可以发送消息，并接收其他用户的消息。每个用户还可以查看当前在线的用户。

3.2 界面设计

实现说明

- 一个Title，作为聊天室标题
- 一个主页面用来接收显示各个用户传来的消息
- 一个South布局界面用来发送消息
- 一个East布局界面用来显示用户名单

核心代码

其中仍然沿用了Layerout的布局模式，在south栏布置了发送，在east栏布置了名单，在center布置了聊天页面

代码中核心组件为JTextArea类型组件，这是用来作为文本显示的；JScrollPane组件，是用来让聊天页面的文本可以滚动显示的。

```
public client(){
        //初始化布局设置
        setSize(600,450);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout(10, 10));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //Title设置
        setTitle("The chatting room");//Client项目
        //SOUTH栏设置
        JPanel chatPanel = new JPanel();
        textField = new JTextField(45);
        chatPanel.setBackground(Color.BLUE);
```

```
        button = new JButton("Send");
        chatPanel.add(textField);
        chatPanel.add(button);
        add(chatPanel,BorderLayout.SOUTH);
        //EAST栏设置
        JPanel ClinetPanel = new JPanel();//用来显示聊天室成员名单的
        ClinetPanel.setBackground(Color.GRAY);
        clientarea = new JTextArea("This is all the users",20,12);
        ClinetPanel.add(clientarea);
        add(ClinetPanel,BorderLayout.EAST);
        //主页面设置
        Display = new JTextArea();
        Display.setLineWrap(true);//设置自动换行
        JScrollPane scroller= new JScrollPane(Display);
        Display.setFocusable(false);

 scroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWA
YS);

 scroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_
NEVER);
        add(scroller,BorderLayout.CENTER);
    }
```

3.3 Client设计

实现说明

- 用户可以发送消息，并且屏幕上都会显示出来
- 用户可以在East栏中查看所有当前在线的用户
- 其他用户断开或者加入会有所提示

核心代码

发送代码

在处理发送的模块中，主要监听的是发送的button。这里我预设了**三种报文发送的格式**:

- 常规发送，将会打包一个“Client id:message”的报文发送，id代表了自己是客户编号，message代表自己的消息
- 特殊发送1：发送一条"Client id:!!!"的报文，代表本客户想退出公共聊天室。
- 特殊发送2：发送一条”Client id:BACK“，代表已经退出了公共聊天室的客户申请重新回到公共聊天室

```
public void actionPerformed(ActionEvent e){//如果按下了按钮
    String message = textField.getText();//得到了一条要发送的消息
    System.out.println("Send:"+message);
    if (message!=null && !message.equals("") && id!=-1) {
        try {
            out.println("Client "+id+":"+message);
            //这里进行强行设置，如果发送的内容为CANCEL,将会断开连接
            out.flush();//把缓冲区的内容全部先发出去，message内可能太长导致上一次残留
        } catch (Exception e1) {//应out要求，设置异常检测
            e1.printStackTrace();
        }
    }
    if (message.equals("!!!")==true){
```

```
            System.out.println("Closed");
        } else if (message.equals("BACK")==true){
            System.out.println("you have back");
        }
        textField.setText("");//重新设置回原来的空白文件
    }
```

接收代码

接收代码首先读取从客户端发来的消息，这里有**两种接收报文的格式**：

- 收到的消息为"CANCEL"，说明客户端发送了一条要求断开退出聊天室的命令，那么会清空你目前的聊天页面，显示你已经暂时离开了公共聊天室。
- 收到的消息例如"1 2 3 4 5 6 7 8 9 ###Client 1：We love ZJU"，这里的###之前的为当前聊天室内的用户列表（格式为数字+空格）；后面发送用户+信息类型。特别地，如果你的id=-1，提取当前id作为自身id。
- 特别的，通过记录之前的用户数对比这条信息发来时的用户数，用户将得到提示是否有其他用户加入或者离开。

```
try {
    while ((message=in.readLine()) != null) {
        System.out.println("Reciveve:"+message);
        if (message.equals("CANCEL")==true){

            clientarea.setText("");//清空
            Display.setText("");
            Display.append("You have left"+"\n");
            continue;
        }
        /*
         * 这里约定：客户端传回来的数据包为以下格式：
         * 1 2 3 4 5 6 7 8 9 ###信息
         * 也就是，前面是按照一个用户id+一个空格，然后###，最后是message
         * message格式为:Client id:xxxxx
         */
        String []tmp=message.split("###");
        clientarea.setText("");//清空
        String []Tmpclient=tmp[0].split(" ");//分出信息
        clientarea.append("Client now"+"\n");
        for (int i=0; i<Tmpclient.length; i++)
            clientarea.append("Client:"+Tmpclient[i]+"\n");//给出当前的用户数
        if (Tmpclient.length>num){
            Display.append("Client enter"+"\n");
        }
        else if (Tmpclient.length<num){
            Display.append("Client leave"+"\n");
        }
        if (id==-1){
            id=Integer.parseInt(Tmpclient[Tmpclient.length-1]);
            Display.append("Client "+id+" has connected"+"\n");
        }
        num=Tmpclient.length;
        if (tmp[1].equals("?")==false)
            //如果有真的消息传过来
            Display.append(tmp[1] + "\n");//约定格式，用户号+发送信息号
    }
} catch (Exception e2) {
```

```
    e2.printStackTrace();
}
```

3.4 Server设计

实现说明

- 维护客户端信息，记录在线用户
- 进行消息封装后，将转发消息到所有客户端的公屏上

核心代码

接收连接

对于客户端发来的请求，将会有一个rec记录这是第几个用户，并确定它的连接状态，然后组装消息为”1 2 3 4 5 6 7 8 9 ###？“传给用户，前面是用户列表，默认这是最新连接进来的，因此它的分配编号是最后一个，？代表这是一个单纯的回复连接的请求，并不是其余用户在聊天室内发送了消息

```
try{
    while(true){
        Socket socket=Sersocket.accept();//接收连接
        rec[num]=1;//记录这个用户已经进来了
        PrintWriter writer = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())),true);
        //要发送给客户端的初始化
        String mes="";
        for (int i=0; i<sumnum; i++){
            if (rec[i]!=0){
                mes+=String.valueOf(i);
                mes+=" ";
            }
        }
        mes+="###?";
        num++;//连接的客户数量增加1
        writer.println(mes);//完成组装
        writer.flush();// 传id给连接进来的客户端
        Writers.add(writer);//每一个用户需要对应一个给它们写的writer
        ServerHandler reader = new ServerHandler(socket);
        Thread handler = new Thread(reader);
        handler.start();
    }
} finally{
    Sersocket.close();//关闭
        }
```

群发消息

这里分为**三种情况**

- 若客户发送的消息为"!!!"，代表暂时退出该聊天室。标记，群发消息：”1 2 3 ###Client 4 leaves”的消息格式
- 若客户发送的消息为“BACK”，代表返回聊天室。标记，群发消息“1 2 3 4 ###Client 4 reback”的消息格式
- 若客户发送的不为以上两种，则正常转发，“1 2 3 4 ###Client 1：We love ZJU”的消息格式

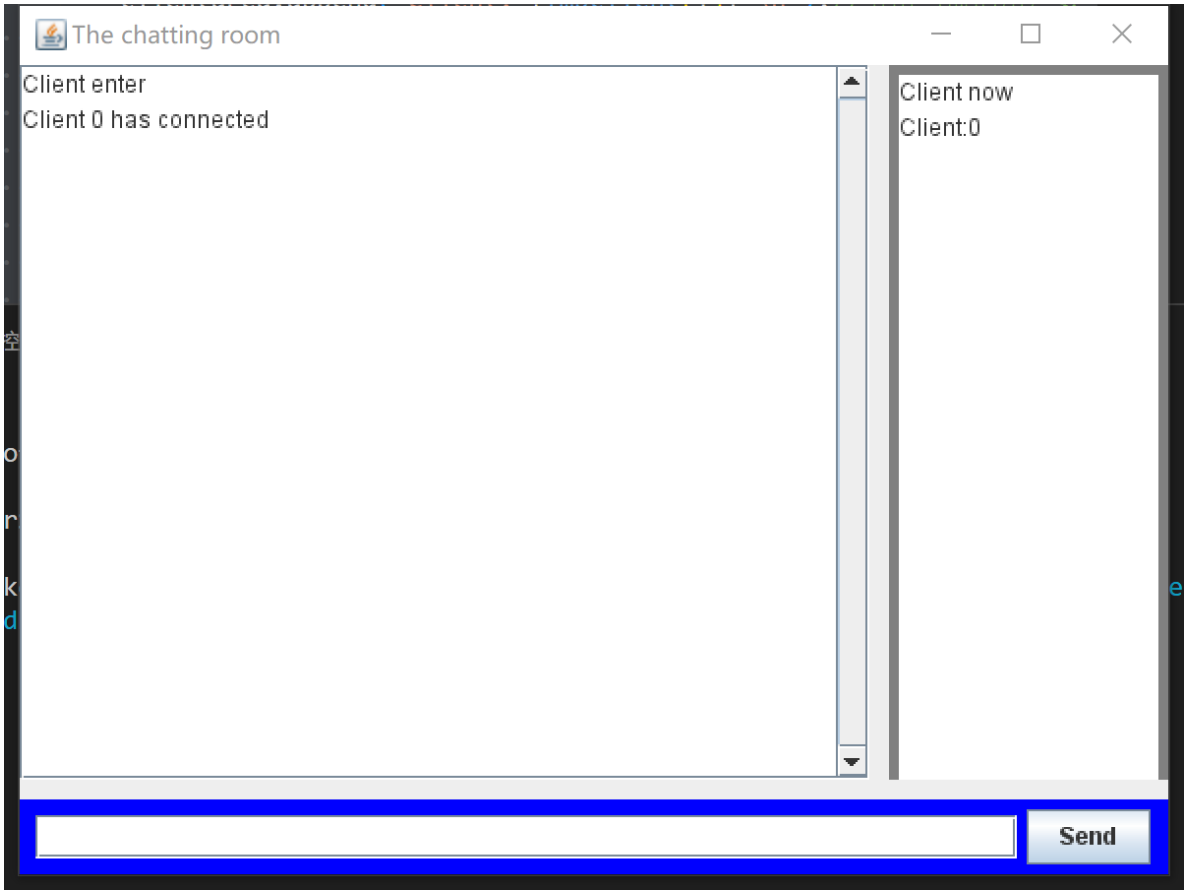
```
try {
```

```
    int index=-1;
    while ((message=in.readLine())!=null){
        System.out.println("Receive "+message);
        String []otemp=message.split(":");
        otemp[0]=otemp[0].substring(7);
        int cc=Integer.parseInt(otemp[0]);//得到是哪个
        if (rec[cc]==0 && message.indexOf("BACK")<0) continue;//如果这个用户已经暂
时离开了，不接受它的消息
        if (message.indexOf("!!!")>=0)
        {//如果收到某一个取消请求的处理，这个时候对于取消的客户发送一个特殊的句子
            //Client 1:CANCEL
            System.out.println("CANCEL YES");
            String []o=message.split(":");
            o[0]=o[0].substring(7);
            index=Integer.parseInt(o[0]);//得到了是哪个Writer在请求断开连接
            rec[index]=0;//记录这个已经不在客户列表里了
            System.out.println("Remove "+index+" ok");//成功移除
            int cnt=0;
            for (PrintWriter writer:Writers) {//对于所有的暂时离开用户,不能维护
                if (rec[cnt]==0) {
                    System.out.println(cnt+" client has out");
                    writer.println("CANCEL");//完成组装
                    writer.flush();
                    cnt++;
                    continue;
                }
                else {
                    cnt++;
                    String mes="";
                    for (int k=0; k<sumnum; k++){
                        if (rec[k]==1){
                            mes+=String.valueOf(k);
                            mes+=" ";
                        }
                    }
                    mes+="###";
                    mes+="Client "+Integer.toString(index)+" leaves";
                    System.out.println("Send"+message);
                    writer.println(mes);//完成组装
                    writer.flush();
                }
            }
        }
        else if (message.indexOf("BACK")>=0){//允许原来离开的人重新回来
            System.out.println("Handle BACK");
            System.out.println("REBACK YES");
            String []o=message.split(":");
            o[0]=o[0].substring(7);
            index=Integer.parseInt(o[0]);//得到了是哪个Writer在请求恢复连接
            rec[index]=1;//记录这个已经人重新回到了这个地方
            System.out.println("Reback "+index+" ok");//成功
            int cnt=0;
            for (PrintWriter writer:Writers) {//对于之后的
                if (rec[cnt]==0) {
                    System.out.println(cnt+" client has out");
                    writer.println("CANCEL");//完成组装
                    writer.flush();
                    cnt++;
```

```
                    continue;
                } else {
                    cnt++;
                    String mes="";
                    for (int k=0; k<sumnum; k++){
                        if (rec[k]==1){
                            mes+=String.valueOf(k);
                            mes+=" ";
                        }
                    }
                    mes+="###";
                    mes+="Client "+Integer.toString(index)+" reback";
                    System.out.println("Send"+message);
                    writer.println(mes);//完成组装
                    writer.flush();
                }
            }
        }
        else {
            int cnt=0;
            for (PrintWriter writer:Writers) {
                if (rec[cnt]==0){
                    writer.println("CANCEL");
                    writer.flush();
                    continue;
                }
                cnt++;
                String mes="";
                for (int i=0; i<sumnum; i++){
                    if (rec[i]!=0){
                        mes+=String.valueOf(i);
                        mes+=" ";
                    }
                }
                mes+="###";
                mes+=message;
                System.out.println("Send "+message);
                writer.println(mes);//完成组装
                writer.flush();
            }
        }
    }
} catch (Exception e2) {
    e2.printStackTrace();
}
```

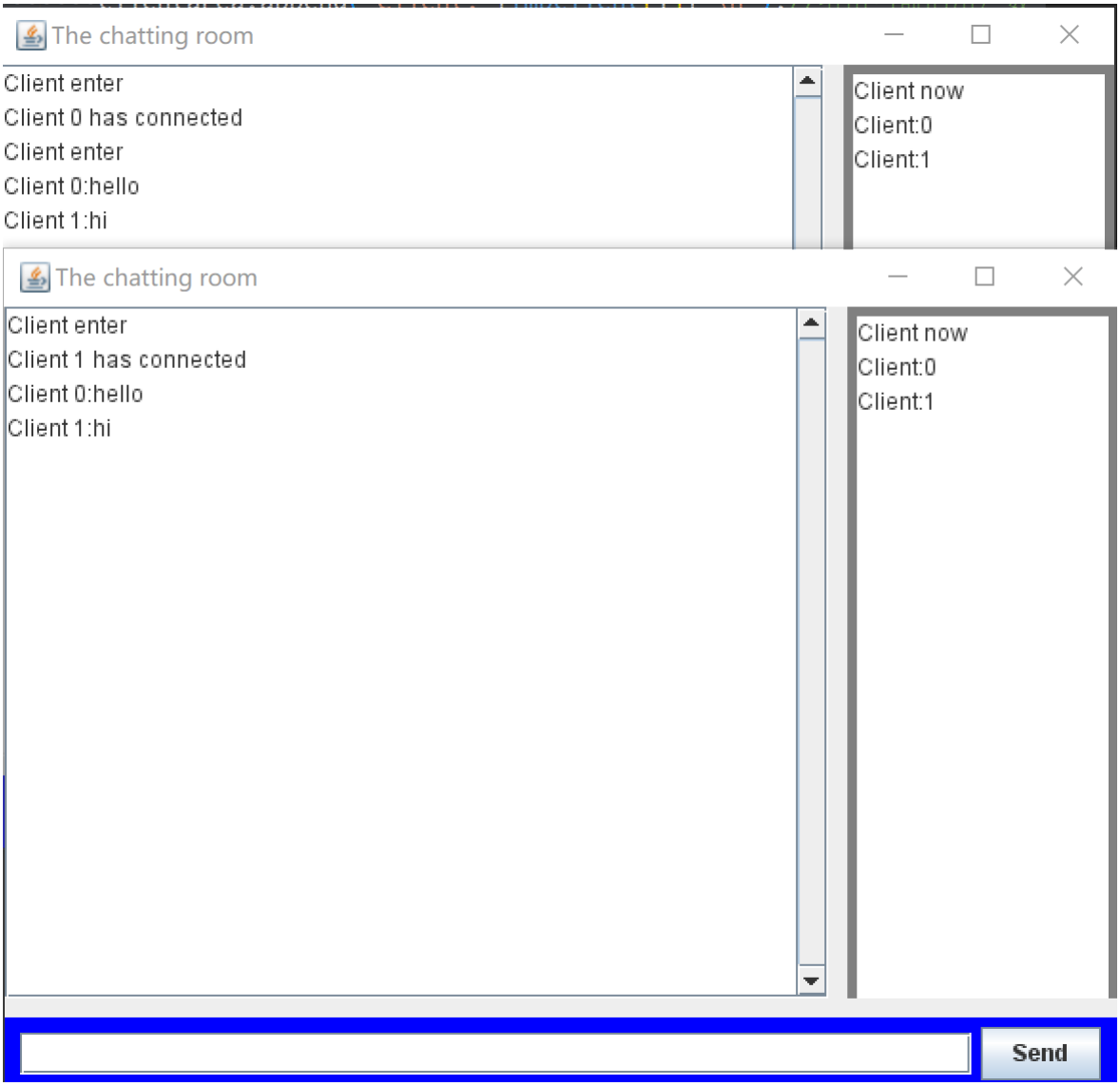
4. 实验结果

4.1 聊天界面显示



4.2 聊天室功能

可以实现聊天，并且新用户进来时有所显示



4.3 退出和重回聊天室

首先，发送一条!!!的消息。

接着，将会看到，短暂退出的用户聊天界面将会有所显示：

具体而言，发送退出的会显示你已经退出，另外一边显示退出的用户

并且，发送了BACK后，可以回来。

5. 实验心得

本次实验中，我主要实现的是一个带有可视化图形界面的公共聊天室的程序。在设计中，首先我是承接了之前MiniCAD中的Swing和AWT知识，设计了一个聊天界面框，其次就开始使用socket知识进行编程。

在Swing的设计中，这里其实对编排有了更好的了解。在Socket的设计中，首先是学习了读写流，然后是学习了java中对于socket的一套API。由于本次实验其实在计算机网络中写过有关C语言的交流，所以这次设计我更注重可视化的体现，不仅要有一个侧边栏可以显示所有的用户；对于用户进出，我都设计了提示；而且，我还允许每一个客户通过要求短暂进入和最终退出聊天室。这些都是我认为更加方便的地方。

最后，我希望下次可以使用类似线程池的技术等，来实现这个项目，最终能够更加的稳定和成熟。而且，我的实验中碰到了一个异常java.net.SocketException: Connection reset，这个我尝试了网上所有的方法都没有能解决，希望能在日后的学习中理解。