课程设计报告二：聊天室系统

一、 课程设计要求与目的

目的：编写一个小型 Java 聊天室系统，掌握 Java 网络通信、多线程、IO 文件操作等高级应用编程技能。

要求：以课本第 15 章 Java 网络通信例 15.3、15.4 的源代码为基础，编写一个小型 Java 聊天室系统。

完成如下功能：

1、多客户端模式下，实现客户与客户的单独通信，要求信息通过服务器中转

2、端到端的通信，实现并行通信模式（不再是你说一句，我说一句，一端的信息发送不受另一端的影响）

3、实现端到端的文件传输

4、添加图形界面（选做）

二、 系统设计

设计思想

使用 Socket 进行通信，服务器端建立 ServerSocket 监听，不断接收来自客户端的请求，每当有客户端连接就启动一个 ServerThread 线程，通过 Socket 获取其对应的输入流和输出流。使用 Map 保存登陆用户的状态信息，可以保存所有用户名和对应的输出流之间的映射关系，当服务器端接收到客户端发送的各种消息时能够对消息进行区分，因此可以在发送不同信息之前对信息进行处理，在内容的前后添加一些特殊字符。对于客户端，每当启动时需要进行登录，输入一个用户名，用户名不能重复不能为空以及长度限制，登录成功服务器端界面显示用户上线信息以及在线的用户，客户端可以进行私聊群聊以及文件发送功能，私聊时可以输入对方的用户名，用户不存在进行提示，默认为群聊，文件发送时选择文件进行群发或者私发，首先发送文件名，服务器端和发送对象用户建立对应的文件输入流，然后使用字符流一行一行向服务器发送，服务器可以接收到一切信息，各种类型的信息用特殊字符来区分。

整体设计（类与类的关系）

Client 类为客户端，Server 类为服务器端，客户端和服务器端各需要两个线程，用来阻止 BufferedReader 的 readLine()方法读取数据时线程被阻塞的问题，一个为主线程，一个用于读取数据的线程，因此建立 ClientThread 和 ServerThread 两个线程类，继承类 Thread，进行各种信息处理，可以使用内部类或者外部类两种形式，该程序当做内部类，可以方便访问一些私有数据成员。类 Client 中有内部类 MyListener 和 MyFileListener，均为事件监听类，new WindowAdapter()为 Client 的匿名内部类。定义的协议字符放在接口 Mystring 里，Client 和 Server 都需要使用这些特殊字符。类 MyMap 通过组合 HashMap 对象来实现，自定义一些功能，方便 Server 的使用，对一些信息进行操作。

局部设计（单个类的设计）

Server 类包含静态数据成员 MyMap<String, PrintStream>的对象 clients 保存所有用户名和对应的输出流之间的映射关系，方法 init()对一些数据成员进行初始建立简易的图形界面，创建一个 ServerSocket 在端口 4700 监听客户请求，死循环，一旦监听到客户请求便根据得到的 Socket 对象创建服务线程 ServerThread 并启动。

Client 类方法 initFrame()对一些数据成员进行初始建立简易的图形界面，init()初始化 socket = new Socket("127.0.0.1", 4700) 连接到服务器，然后获取该 Socket 对应的输入流和输出流，不断地弹出对话框要求输入用户名，成功登陆便打开用户聊天窗口，最后启动 ClientThread 线程。

MyMap 类通过组合 HashMap 对象来实现，建立几个方法实现了能够通过 value 的值来删除对应项和查找对应项，以及实现 put()方法要求值不能重复。

ServerThread 继承 Thread，构造函数接收一个 Socket 创建线程，run()方法获取对应的输入流和输出流，循环 while ((line = br.readLine()) != null) 时对字符串 line 的开头和结尾字符进行区分处理，信息有用户登录的用户名、私聊信息、群聊信息、文件信息、成功接收文件时提示已经接收完毕信息、用户退出信息等，getMessage()用来去掉前后字符得到真正的信息。

ClientThread 继承 Tread，构造函数得到输入流信息， run()方法循环 while ((line = br.readLine()) != null) 时对字符串 line 的开头和结尾字符进行区分处理，信息有普通聊天信息、文件信息等，getMessage()用来去掉前后字符得到真正的信息。

MyListener 类实现 ActionListener 接口，对鼠标点击的按钮进行处理，按钮有私聊、群聊、发送、清屏、退出。MyFileListener 类实现 ActionListener 接口，当点击文件发送按钮时进行发送，匿名内部类 new WindowAdapter()使用户能够进行窗口的关闭，并且加入了发送用户退出信息，服务器端能够对下线用户进行在线用户列表的更新。

三、 系统实现

采用的主要技术

多线程，用来阻止 BufferedReader 的 readLine()方法读取数据时线程被阻塞的问题；**异常处理**，捕捉到异常后时从 Map 中删除用户，关闭各种流，使用 finally 块来关闭线程对应的输入流和输出流等，finally 块用来回收资源；**Swing 编程**，构建简易的图形界面，方便聊天信息的测试，**事件的处理**实现事件监听器类创建对象，调用 addXXListener()方法将对象注册给事件源，当发生指定事件时会触发事件监听器调用相应方法来处理；**Java 的集合**，Map 用于保存具有映射关系的键值对，key 和 value 之间存在单项一对一的关系，本程序通过组合 HashMap 对象来实现自定义的 MyMap 类，实现了能够通过 value 的值来删除对应项和查找对应项，因此可以通过用户名得到对应的输出流，也可以通过输入流找到对应的用户名，方便用于程序的操作；**匿名内部类**，大部分时候事件处理器都没有复用的价值，创建匿名内部类时会立即创建一个该类的实例，这个类的定义立即消失，不能够重复使用。

代码摘要

```
// 获取该 Socket 对应的输入流
br = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                    // 获取该 Socket 对应的输出流
```

```
                    ps = new PrintStream(socket.getOutputStream());
                    String line = null;
                    while ((line = br.readLine()) != null) {
                        // 私聊信息
                        else if (line.startsWith(MyString.PRIVATE_ROUND)
                                && line.endsWith(MyString.PRIVATE_ROUND)) {
                            String userMessage = getMessage(line);
                            if (userMessage.length() == 1) // 没有任何信息时
                                ps.println("没有任何输入，请输入一个用户名");
                            else {
                    String user = userMessage.split(MyString.SPLIT_SIGN)[0];// 分割得到名字
                            if (!clients.map.containsKey(user)) {
                                ps.println("该用户不存在，请重新输入或者选择群聊！");
                                } else {if (userMessage.split(MyString.SPLIT_SIGN).length ==
2)// 是否存在信息
        String msg=userMessage.split(MyString.SPLIT_SIGN)[1];// 分割得到信息
                                // 获取私聊用户对应的输出流发送私聊信息
        clients.map.get(user).println(clients.getKeyByValue(ps)
        + "悄悄地对你说：" + msg); ps.println("你悄悄对" + user + "说:" + msg);
viewArea.append(sdf.format(System.currentTimeMillis())+ "\t"+ clients.getKeyByValue(ps)
        + "悄悄地对" + user + "说：" + msg + "\n");// 服务器打印聊天信息
                                    }}}}
                            // 获取文件名
                else if (line.startsWith(MyString.FILE_NAME)&& line.endsWith(MyString.FILE_NAME)) {
                            if(getMessage(line).startsWith(MyString.PRIVATE_FILE)) {
        Stringmessage=getMessage(line).substring(1,getMessage(line).length());
                            fileUser = message.split(MyString.SPLIT_SIGN)[0];
                            fileName = message.split(MyString.SPLIT_SIGN)[1];
                            clients.map.get(fileUser).println(MyString.FILE_NAME + 0 +
fileName+ MyString.FILE_NAME); // 发送文件名
```

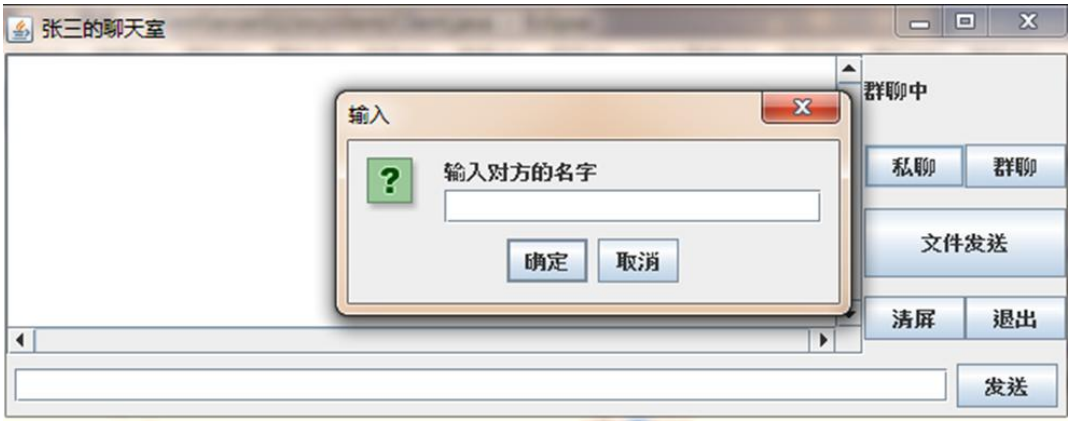
```
                    flag = false; // 设置为正在私聊
                } else {
                    int i = 0;fileName = getMessage(line);
                    for (PrintStream clientPs : clients.valueSet()) {
                if (clientPs != ps) {// 向除自己之外的所有用户发送文件名
                clientPs.println(MyString.FILE_NAME + i+ fileName +
MyString.FILE_NAME); i++; // 区别文件名
                    }
                }flag = true;}
            out = new OutputStreamWriter(new FileOutputStream(fileName),"UTF-8");
                    // 接收文件并转发
                    else if (line.startsWith(MyString.FILE_ROUND)
    && line.endsWith(MyString.FILE_ROUND)) {String userMessage = getMessage(line); if (!flag)
{
                clients.map.get(fileUser).println(MyString.FILE_ROUND + userMessage+
MyString.FILE_ROUND);} else {
                for (PrintStream clientPs : clients.valueSet()) {if (clientPs != ps) {
    // 向除自己之外的所有用户发送文件
        clientPs.println(MyString.FILE_ROUND+ userMessage+
MyString.FILE_ROUND); }}}out.write(userMessage + "\n");// 服务器接收文件
                    }
                    // 成功接收文件时提示已经接收完毕
                    else if (line.startsWith(MyString.FILE_END)) {
    viewArea.append(sdf.format(System.currentTimeMillis())
    + "\t" + "成功接收" + clients.getKeyByValue(ps)+ "发送的文件:" + fileName + "\n");if (!flag)
{clients.map.get(fileUser).println("成功接收" + clients.getKeyByValue(ps)
        + "发送的文件:" + fileName);} else {
            for (PrintStream clientPs : clients.valueSet()) {
                    if (clientPs != ps) {// 向除自己之外的所有用户发送文件
                clientPs.println("成功接收"+ clients.getKeyByValue(ps)+ "发送的文件:" +
```

```
fileName);}}}} else if (line.startsWith(MyString.EXIT)) {// 用户退出
        viewArea.append(sdf.format(System.currentTimeMillis())
                                                + "\t" + clients.getKeyByValue(ps) + "下线\n");
                                        listModel.removeElement(clients.getKeyByValue(ps));
                                    clients.removeByValue(ps);ps.close();// 移除下线的用户
                                }// 群聊
                                    else {String msg = getMessage(line);
                                        // 遍历 clients 中的每个输出流发送信息
                                    for (PrintStream clientPs : clients.valueSet()) {
                                    if (clientPs == ps) {ps.println("你说:" + msg);
                                            } else {
            clientPs.println(clients.getKeyByValue(ps)+ "说： " + msg);}
        viewArea.append(sdf.format(System.currentTimeMillis())+ "\t" + clients.getKeyByValue(ps) +
"说： " + msg+ "\n");}}
        }
        public interface MyString {
            String MSG_ROUND = "¤§"; // 普通信息
            String USER_ROUND = "ΠΣ"; // 用户名
            String LOGIN_SUCCESS = "1";
            String NAME_ERROR = "-1";
            String PRIVATE_ROUND = "『』";
            String SPLIT_SIGN = "※";
            String FILE_ROUND = "ɕɕ";
            String FILE_NAME = "ɣʔɣ";
            String PRIVATE_FILE = "ㄱ ";
            String FILE_END = "ế";
            String EXIT = "Ę";
            String IS_FLAG = "£Þ";
        }
```

四、 系统测试

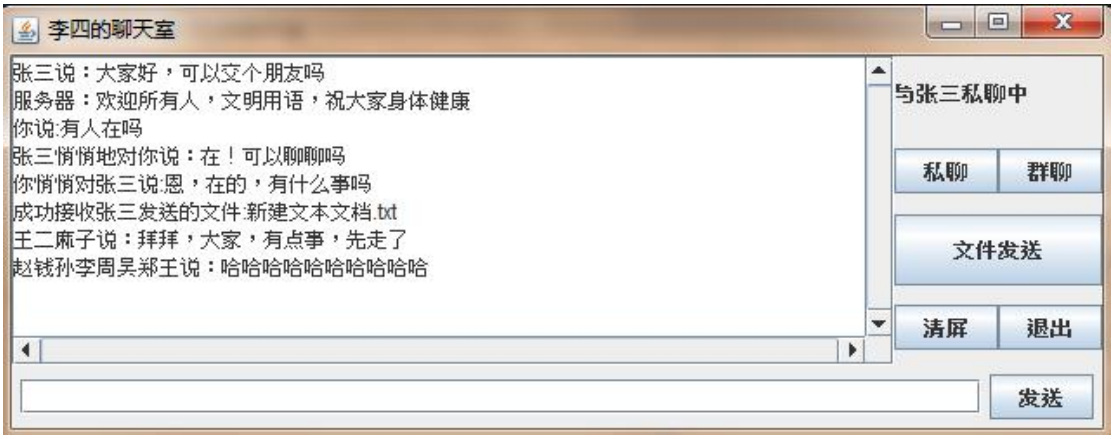
用户登陆上线，如果没有任何输入，字符长度大于 10 以及与在线用户的用户名重复时会不停的弹出

选择私聊时需要输入对方的名字，没有输入或者用户不在线时会进行提示

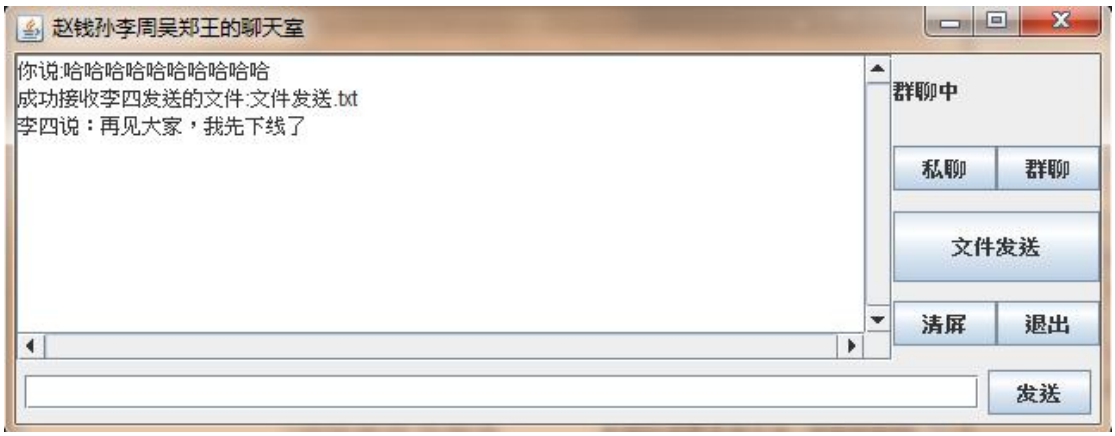


发送文件进行文件选择

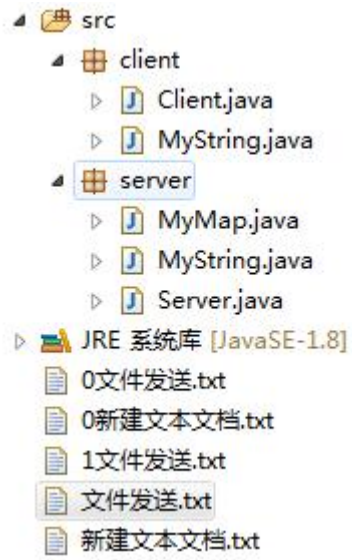
用户聊天信息，默认为群聊，发送的信息所有人都能看到，使用 Alt+Enter 为快捷键发送信息，发送的文件存放在项目目录下，正在与对方私聊时只向对方发送文件，发送成功进行提示



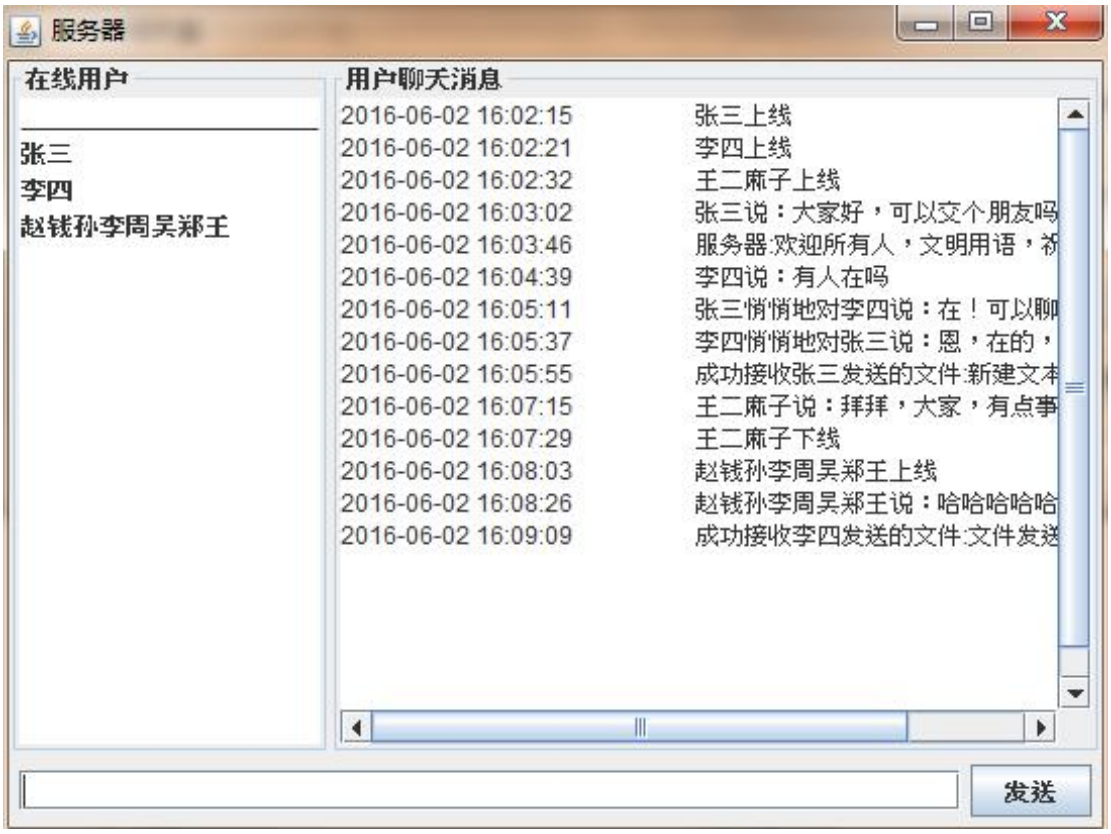
群聊时发送的文件所有在线的用户都能收到



接收到文件，由于放在同一目录下，服务器接收的为原名，其他用户用 0,1,2…表示



服务器端记录在线用户以及用户的聊天信息，记录用户的上线下线，接收用户发送的各种文件，可以进行信息的发送，所有在线用户都能收到



五、 课程设计总结

通过本次课程设计实现了一个简易的聊天系统，具体系统功能有登陆用户可以实现单独的私聊以及与在线用户进行群聊，并行通信一端的信息发送不受另一端的影响，实现了端到端之间的文件发送功能。使用的 Java 编程技术有多线程、异常处理、Swing 编程、java 的集合等。课程设计过程中也遇到了很多问题，比如如何实现文件的发送，解决办法采用了先发送文件名，增加了输出流 OutputStreamWriter out 用来接收文件，接收到文件名便 out = new OutputStreamWriter(new FileOutputStream(fileName), "UTF-8")进行初始化；如何实现在线用户列表以及列表的更新，使用了 JList 和 DefaultListModel，用户登录成功 listModel.addElement(userName)更新在线列表，用户退出需要发送退出信息 listModel.removeElement()移除下线的用户；开始发送文件时发生了中文乱码的问题，程序使用 UTF-8 编码，GBK 每个汉字两个字节，而 UTF-8 每个汉字三个字节，输入输出应使用同一种编码方式，解决办法是让文本文档另存为 UTF-8 编码，建立文件输出流和输入流时之后加上"UTF-8"。但是程序还是存在一些不足之处，比如在网上查了资料打算实现发送 word 等其他格式的文档，需要使用 Apache POI，但是最终没有实现，文件发送功能只能发送.txt 文件；接收的文件没有建立各个用户独有的文件目录，而是全部放在了项目目录下，只好采用服务器接收的为原名，其他用户用 0,1,2…表示，还要测试时如果发送文件个数较多时可能会发生部分内容丢失或者文件空白的问题，使用了一些方法未能解决；另外聊天界面过于简易，实现的功能不多，没有实现客户端在线用户列表，以至于私聊时只能进行输入，对用户名处理表现的有些繁琐，因此程序还有很多地方需要进行改善。通过本次课程设计很大提高使用 Java 语言进行编程和调试程序的能力，学习到很多 Java 编程的知识，比如使用 Swing 编程设计简易的界面等，同时也发现了许多自身的不足之处，对于一些知识掌握的并不是很深刻，查找一些资料动手编写程序对编程能力的提高有很多帮助。