## 1.课程设计目标

借此次java课程设计，来巩固所学Java语言基本知识，进而增进Java语言编辑基本功，掌握JDK、Editplus、Eclipse、JCreator等开发工具的运用，拓宽常用类库的应用。把所学课程及相关知识加以融会贯通，全面掌握Java语言的编程思想及面向对象程序设计的方法，为今后从事实际应用中打下坚实的基础。

本课程设计聊天系统，分为聊天客户端和聊天服务器。

## 2.系统需求分析

### 2.1 系统目标

使用所学知识制作一个基于java的聊天系统。

### 2.2 主体功能

客户端与服务端的链接，服务端仅作为转发消息的中转，依次实现客户端一对一，一对多的通信功能。

### 2.3 开发环境

Eclipse

## 3.系统设计

### 3.1设计思路

本系统客户端包括13个java源文件和客户端所保存的聊天消息文件。服务端包括7个java源文件和用户的数据文件。客户端与服务端建立连接，并利用socket编程实现收发消息。服务端主线程用于监听想要链接的客户端，收到客户端请求时，则创建一系列子线程来处理各个客户端的请求。

### 3.2类与接口的设计

#### 服务端（图1和表1）

|  |  |  |  |
| --- | --- | --- | --- |
| 表1 | | | |
| **序号** | **文件名** | **主要功能** | **备注** |
| 1 | Client\_Thread.java | 实现服务端功能的类，通过线程单独处理每一个客户端消息 |  |
| 2 | Server\_Network.java | 服务器主线程，等待客户端连接并建立新线程 |  |
| 3 | Server\_Interface.java | 服务器可视化管理界面，可以管理用户和群组 |  |
| 4 | Working.java | 仅用于执行程序 |  |
| 5 | Data\_Reading.java | 读取文件建立文件树 | 提供给两个管理面板用 |
| 6 | Group\_Panel.java | 管理界面的群组分类，实现管理群组的操作 |  |
| 7 | User\_Panel.java | 管理界面的用户分类，实现管理用户的操作 |  |

图 1

#### 客户端（图2和表2）

表 2

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **文件名** | **主要功能** | **备注** |
| 1 | Client\_Begin.java | 登录窗口的实现和组件监听 |  |
| 2 | Login\_Interface.java | 登录窗口的分类，构建登录界面 |  |
| 3 | Register\_Interface.java | 登录窗口的分类，构建祖册界面 |  |
| 4 | Build\_Tree.java | 建立好友树和群组树组件 |  |
| 5 | Center\_Panel.java | 聊天消息显示和聊天消息输入的界面 |  |
| 6 | Left\_Panel.java | 整合好友、群组列表及个人信息的界面 | 卡片布局 |
| 7 | Friend\_List.java | 好友列表（树）的监听管理 |  |
| 8 | Group\_List.java | 群组列表（树）的监听管理 |  |
| 9 | Person\_Panel.java | 个人信息的显示和管理 |  |
| 10 | Client\_Interface.java | 整合Left\_Panel和Center\_Panel的客户端主窗口 |  |
| 11 | Client\_Network.java | 客户端接收消息的线程 |  |
| 12 | Running.java | 仅用于运行程序 |  |
| 13 | TalkEmoji.java | emoji聊天表情快捷窗口 |  |

图 2

## 4.系统具体实现

### 4.0 基础设想

本系统服务端包括7个java文件和用户数据及群组数据，而客户端包括13个java文件和用户本地保存的消息记录。保证服务端的开启，客户端才能使用，如果服务端的ip有变动则需要程序内部修改，这是认为不应该由用户设置ip以此达到便捷效果。

客户端一经启动，便于服务端完成链接，此时服务端对应的线程仅有连接建立的时间作为标识，等到用户完成登录，才会使服务端对应的线程标识是本客户端账号。

### 4.1服务端

#### 4.1.1、Server\_Network主类(包括Runnable接口)

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| socket编程服务端 | ServerSocket | server |
| 建立了连接的服务端列表 | ConcurrentHashMap  <String,Client\_Thread> | clientlist |
| 聊天界面 | Server\_Interface | si |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Server() | 建立Socket服务端，构建服务器窗口 |  |
| run() | 显示服务器窗口，保持运行等待客户端连接请求 | 请看列表下方 |

run注释及代码：

一直运作监听，等待客户端接入，然后为客户端建立线程，归入哈希图

public void run()

{

si.server\_interface\_display();

while(true)

{

try {

Socket ss=server.accept();

if(ss.isConnected()) {System.out.println("一个新链接");}

Client\_Thread ct=new Client\_Thread(ss);

clientlist.put(ct.key, ct);

} catch (IOException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

}

#### 4.1.2、Client\_Thread类(包括Runnable接口)

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 读取收到的消息的输入流 | BufferedReader | ii |
| 发送消息的输出流 | PrintStream | oo |
| 与客户端的连接 | Socket | client |
| 这个的线程对应的  客户端名称 | String | key |
| 操作文件用 | File | file |
| 线程 | Thread | light |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Client\_Thread() | 完成线程初始化 |  |
| run() | 持续接收消息 | 客户端下线后能自我关闭线程，代码如表格下 |
| message\_analysis(String) | 收到的消息解析  并进行需要的操作 |  |
| send(String) | 发送消息 |  |
| vilidate(String) | 用户登录验证其账号密码 |  |
| registration(String) | 用户注册保存其信息 |  |
| addfriend(String,string) | 处理用户添加朋友  传来的消息 |  |
| friendlist() | 整合用户好友列表并返回 | 配合friend\_list()去除已删除用户的更新 |
| grouplist() | 整合群组的列表并返回 | 自身检测群组是否被删除并且更新群组列表 |
| friend\_exist(String) | 判断用户是否存在 |  |
| delFolder(String) | 删除文件，主要用于用户注销删除其数据 |  |
| delAllFile(String) | 删除文件 | 与delFolder配合使用 |

run代码：

public void run()

{

String message=null;

while(true)

{

try {

if(!client.isClosed())

{message=ii.readLine();}

else

{break;}

} catch (IOException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

if(message!=null)

{

message.trim();

System.out.println("从"+key+"接收："+message);

message\_analysis(message);

}

}

System.out.println(key+"断开了链接，其线程已关闭");

light.stop();

}

#### 4.1.3、Server\_Interface 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 服务端窗口 | JFrame | si |
| 用户管理界面 | User\_Panel | up |
| 群组管理界面 | Group\_Panel | gp |
| 用户、群组的布局界面 | JPanel | si\_card |
| 布局界面用的卡片布局 | CardLayout | card |
| 操控卡片布局的按钮 | JButton | sibu,sibg |
| 提示当前是哪个卡片 | JLabel | sil |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Server\_Interface() | 构建服务端窗口 | 使用两个按钮来进行  卡片布局卡片的更替 |
| server\_interface\_display() | 显示服务端窗口 |  |

#### 4.1.4、Data\_Reading 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 树形组件 | JTree | tree |
| 树模型 | DefaultTreeModel | newModel |
| 树模型节点 | DefaultMutableTreeNode | Node |
| 树模型节点 | DefaultMutableTreeNode | temp |
| 读取文件建树的路径 | String | path |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Data\_Reading(String) | 构建树 |  |
| traverseFolder(String) | 读取文件建立树 | 注释如下 |

traverseFolder(String)代码：

public DefaultMutableTreeNode traverseFolder(String path) {

DefaultMutableTreeNode fujiedian = new DefaultMutableTreeNode(new File(path).getName());

File file = new File(path);

if (file.exists()) {

File[] files = file.listFiles();

if (files.length == 0) {

if(file.isDirectory()) {//如果是空文件夹

DefaultMutableTreeNode dn=new DefaultMutableTreeNode(file.getName(), false);

return dn;

}

}else{

for (File file2 : files) {

if (file2.isDirectory()) {

//是目录的话，生成节点，并添加里面的节点

fujiedian.add(traverseFolder(file2.getAbsolutePath()));

}else{

//是文件的话直接生成节点，并把该节点加到对应父节点上

temp=new DefaultMutableTreeNode(file2.getName());

fujiedian.add(temp);

}

}

}

} else {//文件不存在

return null;

}

return fujiedian;

}

#### 4.1.5、Group\_Panel 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 群组管理界面面板 | JPanel | gp |
| 需要用到的按钮 | JButton[4] | gpb[0]~gpb[3] |
| 文本域 | JtextArea | gpta |
| 生成群组树的类 | Data\_Reading | group |
| 提供对文件操作 | File | file |
| 记录当前对树节点的选择 | String | selection |
| 滚动条 | JScrollPane | sp |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Group\_Panel() | 构建群组管理界面面板 | 包含了按钮的监听  和树的选择监听 |
| get\_Panel() | 获取这个面板 |  |
| delete\_group(String) | 删除群组 |  |
| create\_group() | 新建群组并显示在树上 | 内含一个窗口用于  输入新群组名称 |
| saveTextArea() | 保存文本域内容 | 可以作为群组介绍 |
| setTextArea() | 读取文件显示在文本域 |  |
| addgroup(String) | 将新创建的群组  保存在文件中 |  |

#### 4.1.6、User\_Panel 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 用户管理界面面板 | JPanel | up |
| 用户管理界面操作面板 | JPanel | up\_other |
| 显示用户信息的单行文本框 | JTextField[2] | upt[0]~upt[1] |
| 需要用到的按钮 | JButton | upb1,upb2,upb3 |
| 生成树的类 | Data\_Reading | user |
| 记录当前对树节点的选择 | String | selection |
| 滚动条组件 | JScrollPane | sp |
| 提示用标签 | JLabel | upl |
| 提供对文件操作 | File | file |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| User\_Panel() | 构建用户管理界面 | 包含各组件监听 |
| user\_save() | 编辑后保存用户的信息 |  |
| user\_reading() | 读取用户的信息并显示 |  |
| delFolder(String) | 删除用户文件 |  |
| delAllFile(String) | 删除用户文件 |  |
| get\_Panel() | 获取up面板 |  |

### 4.2客户端

#### 4.2.1 Client\_Network 主类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 与服务端的链接 | Socket | client |
| 接收消息的输入流 | BufferedReader | iin |
| 发送消息的输出流 | PrintStream | oout |
| 登录/注册窗口 | Client\_Begin | cb |
| 客户端窗口 | Client\_Interface | ci |
| 客户端窗口左面板 | Left\_Panel | lp |
| 左面板内的个人信息面板 | Person\_Panel | lppp |
| 左面板内的群组列表面板 | Group\_List | lpgl |
| 左面板内的好友列表面板 | Friend\_List | lpfl |
| 提供对文件操作 | File | file |
| 客户端窗口右（中）面板 | Center\_Panel | cp |
| 当前用户的账号 | String | accent |
| 接收到的消息 | String | message |
| 分割后的消息组 | String[] | fort |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Client\_Network() | 建立与服务端链接并构建登录/注册窗口 |  |
| run() | 保持运行接收服务端发来的消息并且分析执行下一步 | 请看列表下方 |
| send(String) | 向服务端发送消息 |  |
| saveword(String,String) | 存储聊天的消息 |  |
| getaccent() | 获得用户的账号 |  |
| closeclient() | 关闭与服务端连接 |  |
| delFolder(String) | 用户注销删除本地文件 | 与delAllFile配合使用 |
| delAllFile(String) | 用户注销删除本地文件 |  |

run()的解释:

run首先显示登录窗口，然后在用户输入信息并点击登录后，客户端与服务器交互了login、friendlist和grouplist三条消息后，才会构建出客户端窗口并显示。

login用于Person\_Panel（即个人信息面板）的构建，friendlist用于Friend\_List（即好友列表面板）的构建，grouplist用于Group\_List（即群组列表面板）的构建。

#### 4.2.2 Client\_Begin 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 登录窗口 | JFrame | cb |
| 登录面板 | Login\_Interface | cli |
| 注册面板 | Register\_Interface | cri |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Client\_Begin() | 构建登录/注册窗口 | 请看列表下方法 |
| login\_display() | 在窗口显示登陆面板  并监听组件 |  |
| register\_display() | 在窗口显示注册面板  并监听组件 |  |
| successs\_register(boolean) | 提示注册是否成功 |  |
| non\_login() | 登录失败消息提示 |  |
| beginclose() | 完成登录关闭窗口 |  |
| illegal\_character(String) | 对账号密码非法字符的判定 | 本系统只允许数字和英文字母作为账号密码 |

注释：

登录窗口构建出来后，优先显示登陆面板，如果用户点击了注册，则移除登陆面板给窗口添加注册面板。由于本程序在启动时就与服务端建立了链接，因此不仅在客户端窗口关闭后，而在登录窗口选择了关闭，也会发送离线消息并断开连接，确保服务器稳定运行。

#### 4.2.3 Login\_Interface 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 登陆面板 | JPanel | li |
| 登陆面板提示用标签 | JLabel[5] | lil[0]~lil[4] |
| 需要用到的按钮 | JButton | lib1,lib2 |
| 账号单行文本框 | JTextField | lit1 |
| 密码单行文本框 | JPasswordField | lit2 |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Login\_Interface() | 构建登陆面板 |  |
| shut\_lil() | 关闭账号密码非法字符提示 |  |

#### 4.2.4 Register\_Interface 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 注册面板 | JPanel | ri |
| 注册面板提示用标签 | JLabel[8] | ril[0]~ril[7] |
| 需要用到的按钮 | JButton | rib1,rib2 |
| 账号单行文本框 | JTextField | rit[0] |
| 密码及确认密码单行文本框 | JPasswordField | rit[1],rit[2] |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Register\_Interface() | 构建注册界面 |  |

#### 4.2.5 Client\_Interface 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 客户端窗口 | JFrame | ci |
| 客户端窗口内的右面板 | Center\_Panel | cp |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Client\_Interface(Left\_Panel) | 构建客户端窗口 | 设置了关闭监听，点击后会向服务端发送下线消息 |
| client\_interface\_display() | 显示客户端窗口 |  |

#### 4.2.6 Left\_Panel 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 左面板 | JPanel | lp |
| 左面板卡片布局面板 | JPanel | lp\_card |
| 用于切换卡片的按钮 | JButton[3] | lpb[0]~lpb[2] |
| 个人信息面板 | Person\_Panel | pp |
| 好友列表面板 | Friend\_List | fl |
| 群组列表面板 | Group\_List | gl |
| 卡片布局 | CardLayout | card |
| 卡片布局当显示的卡片 | int | cardnum |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Left\_Panel(Person\_Panel,  Friend\_List,Group\_List) | 构建左面板，建立卡片布局转换监听 | 3个按钮分别对应三张卡片 |
| get\_lp() | 获得左面板 |  |

#### 4.2.7 Person\_Panel 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 个人信息面板 | JPanel | ppp |
| 提示用标签组件 | JLabel[3] | ppl |
| 显示个人信息的单行文本框 | JTextFiled[2] | ppt |
| 修改信息和确认修改的按钮 | JButton | ppb1,ppb2 |
| 添加好友用提示标签 | JLabel | newl |
| 添加好友用的单行文本框 | JTextField | newt |
| 确认添加文本框内账号好友 | JButton | newb |
| 注销按钮 | JButton | clo |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Person\_Panel(String) | 构建个人信息面板，  监听组件 | 流式布局 |
| add\_friend(boolean) | 添加好友成功与否提示 |  |
| change\_yes() | 修改信息成功提示 |  |
| illegal\_character(String) | 修改信息非法字符判定 |  |

#### 4.2.8 Friend\_List类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 好友列表树 | JTree | fl |
| 所有好友账号字串 | String[] | friend |
| 当前好友树选择 | String | selection |
| 建立好友树组件的类 | Build\_Tree | bt |
| 滚动条 | JScrollPane | sp |
| 在线好友记录 | ConcurrentHashMap<String, DefaultMutableTreeNode> | online |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Friend\_List(String) | 构建好友列表面板 |  |
| online\_friend(String) | 好友上线记录 | 上线记录进哈希图 |
| offline\_friend(String) | 好友下线记录 | 下线从哈希图移除 |
| returnnode(String) | 配合上线记录 | 注释请看表下方 |
| repeat\_friend(String) | 是否有该好友检测 |  |
| remind(String) | 消息提示 |  |
| deletefriend(String) | 从树组件中移除好友 |  |
| getselection() | 获取当前好友树选择 |  |

returnnode(String)，online\_friend(String),offline\_friend(String)注释及代码：

收到某用户上线消息，并判定其是自己好友，通过其账号遍历节点，返回一个节点模型，配合其账号储存在哈希图中，并删去名称中的“[offline]”离线标识。在哈希图中方便于消息提示在名称上添加的“(\*)”消息提示字符。收到下线消息后，则增加离线标识并移出哈希图。

public void online\_friend(String sss)//好友上线标记

{

if(!repeat\_friend(sss))

{

online.put(sss, returnnode(sss));

online.get(sss).setUserObject(sss);

DefaultTreeModel treeModel = (DefaultTreeModel) fl.getModel();

//刷新

treeModel.reload();

}

}

public void offline\_friend(String sss)//好友离线标记

{

if(!repeat\_friend(sss))

{

online.get(sss).setUserObject(sss+"[offline]");

online.remove(sss);

DefaultTreeModel treeModel = (DefaultTreeModel) fl.getModel();

//刷新

treeModel.reload();

}

}

public DefaultMutableTreeNode returnnode(String sss)

{

TreeNode root=(TreeNode) fl.getModel().getRoot();

Enumeration<?> e= root.children();

while(e.hasMoreElements())

{ //遍历枚举对象.

//先定义一个节点变量.

DefaultMutableTreeNode node;

node=(DefaultMutableTreeNode) e.nextElement();

if((sss+"[offline]").equals(node.toString()))

{return node;}

}

return null;

}

#### 4.2.9 Group\_List类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 群组列表树 | JTree | gl |
| 当前群组树组件选择 | String | selection |
| 构建群组树的类 | Build\_Tree | bt |
| 滚动条 | JScrollPane | sp |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Group\_List(String) | 群组列表面板构建 |  |
| delete\_group(String) | 从树组件中删除群组 |  |

#### 4.2.10 BuildTree类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 树组件 | JTree | tree |
| DefaultTreeModel | DefaultTreeModel | newModel |
| DefaultMutableTreeNode | DefaultMutableTreeNode | Node |
| DefaultMutableTreeNode | DefaultMutableTreeNode | temp |
| 构建树用到的信息 | String | path |
| 好友树或群组树构建判断 | int | h |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Build\_Tree(String,int) | 构建树 |  |
| traverseFolder() | 依据信息构建树返回树的根 | 注释如下 |

traverFolder()代码：

public DefaultMutableTreeNode traverseFolder(String path)

{

String element[]=path.split("\\|");

DefaultMutableTreeNode fujiedian = new DefaultMutableTreeNode("此处用来刷新你的窗口");

for(int i=1;i<element.length;i++)

{

if(h==1)

{temp=new DefaultMutableTreeNode(element[i]+"[offline]");}

else {temp=new DefaultMutableTreeNode(element[i]);}

fujiedian.add(temp);

}

return fujiedian;

}

#### 4.2.11 Center\_Panel 类

类的成员

|  |  |  |
| --- | --- | --- |
| **成员变量描述** | **变量类型** | **名称** |
| 右（中）面板 | JPanel | cp |
| 聊天消息文本域 | JTextArea | up |
| 待发送消息文本域 | JTextArea | down |
| 发送消息按钮 | JButton | cpb1 |
| 表情窗口按钮 | JButton | cpb2 |
| 表情窗口 | TalkEmoji | te |

类的方法

|  |  |  |
| --- | --- | --- |
| **方法名** | **功能** | **备注** |
| Center\_Panel() | 构建右（中）面板 |  |
| record\_display(String) | 读取消息记录并显示 | 注释如表下 |
| record\_save(String) | 保存自己发送的消息 |  |

record\_display注释及代码：

当用户点击树，会发生发生选择节点的更新，会从指定的消息记录文件中获得从前的聊天记录并显示在文本域中。

public static void record\_display(String address) throws IOException

{

File file=new File("客户端\\"+Client\_Network.getaccent()+"\\"+address+".txt");

if(!file.exists())

{file.createNewFile();}

FileReader fr = new FileReader(file);

BufferedReader br = new BufferedReader(fr);

try

{

up.setText("当前显示的是与 "+address+" 聊天消息"+"\r\n");

while(br.ready())

{up.append(br.readLine()+"\r\n");}

}

finally

{

br.close();

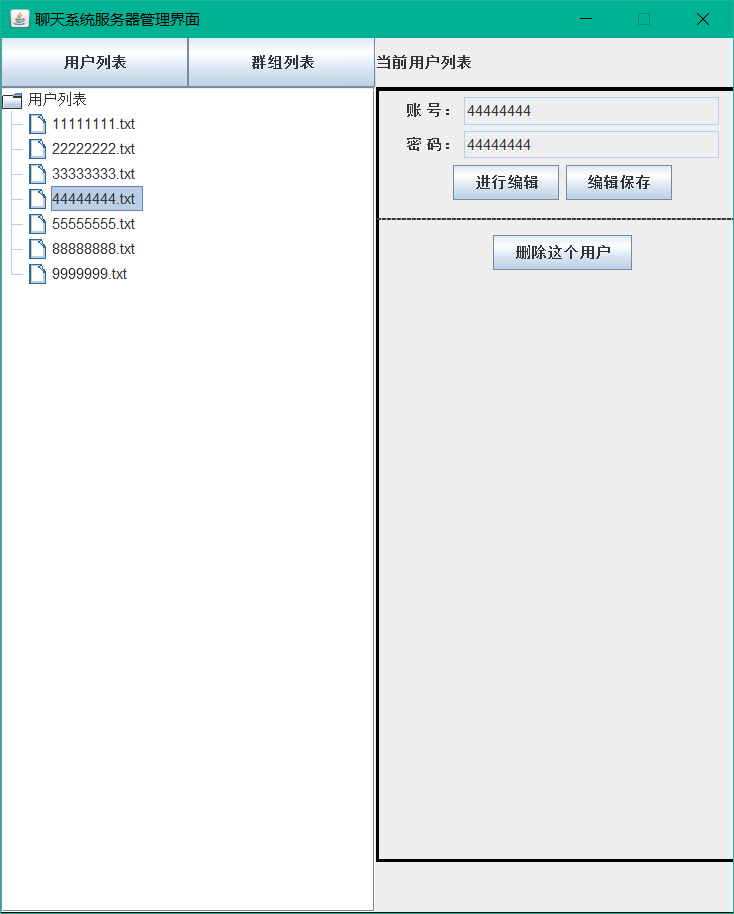
fr.close();

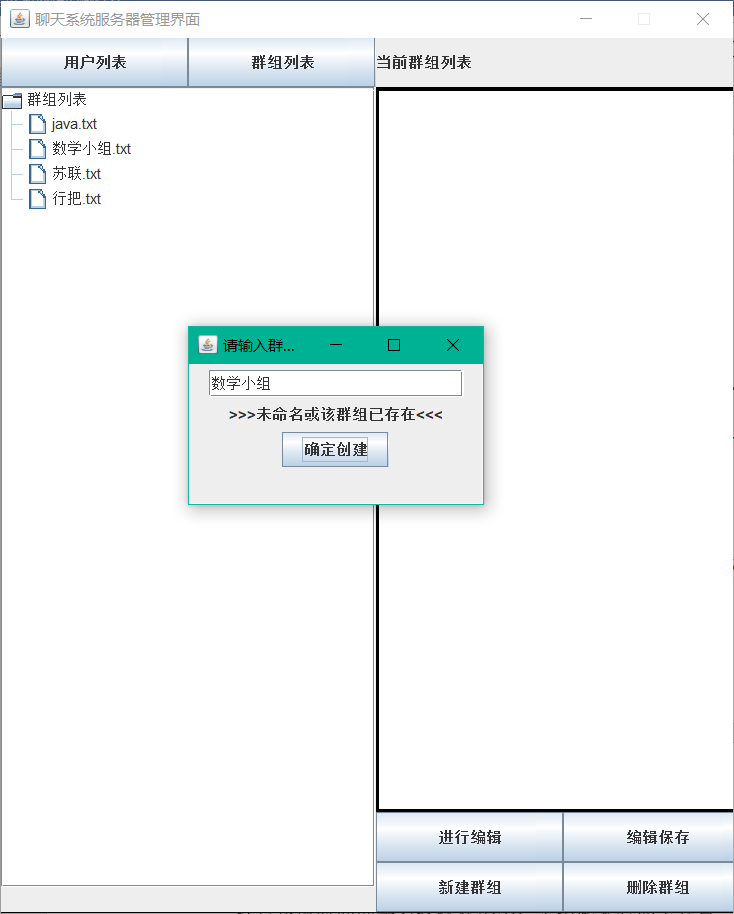
}

}

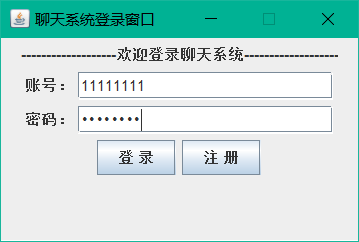
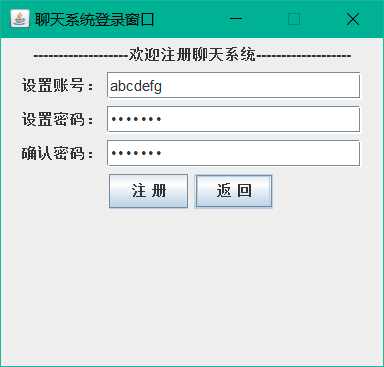
## 5.系统运行效果

运行程序，首先将服务器启动，窗口显示后，用户树组件，群组树组件都完成显示，并能通过右面板进行对用户和群组的管理。

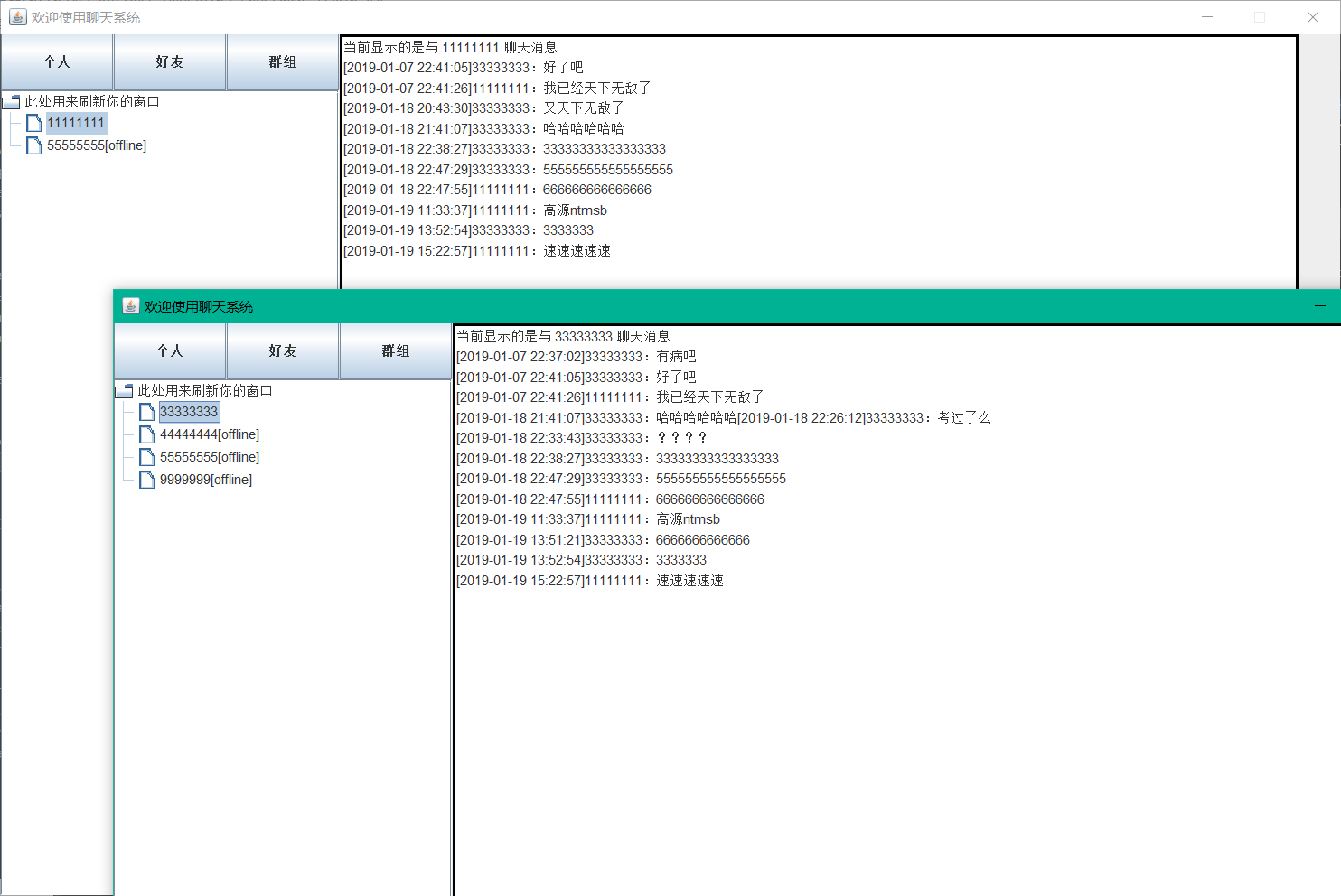


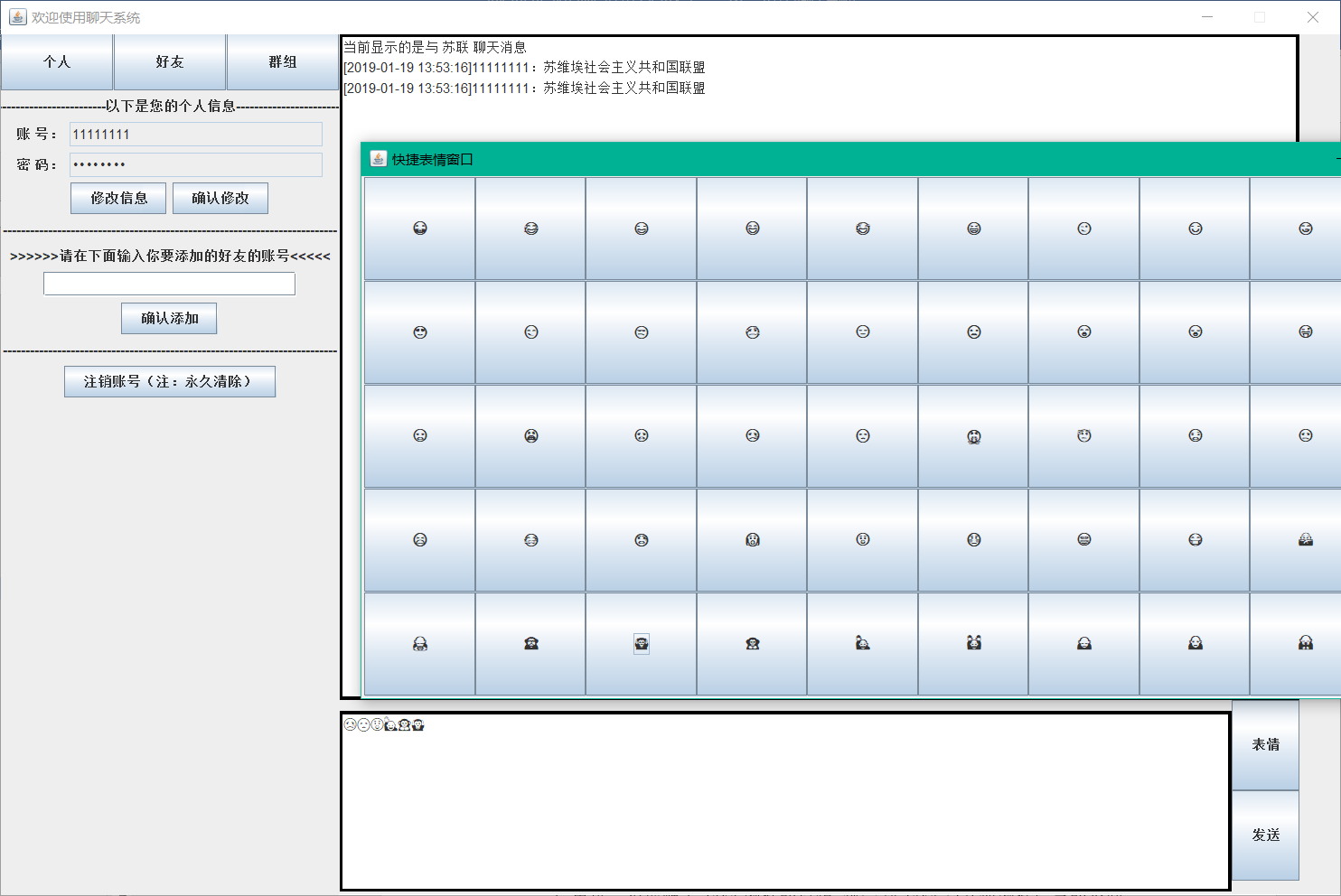


然后启动客户端，登录与注册面板很好的切换

客户端登录完成并实现私聊群聊，个人信息，添加好友的功能，以及聊天表情快捷窗口





程序完成了最开始保证的功能，各功能都能较好的运行，服务端与客户端之间稳定的通信，没发现什么突出的问题。