**《Java技术》课程上机报告**

**综合大作业**

**学 号**：­­­ 14130130231

**姓 名**： 吴天成

**提交日期：** 2017.5.26

## 一、任务简要描述

用Java实现的客户端/服务器聊天室

## 二、系统需求分析

软件主要功能如下：

1. 用户注册
2. 用户登陆
3. 一个聊天室
4. 多个客户同时聊天
5. 不能重复登陆
6. 每个人能够看到当前聊天室的情况，有多少人，都有哪些人在聊天
7. 多个聊天室
8. 能够私聊

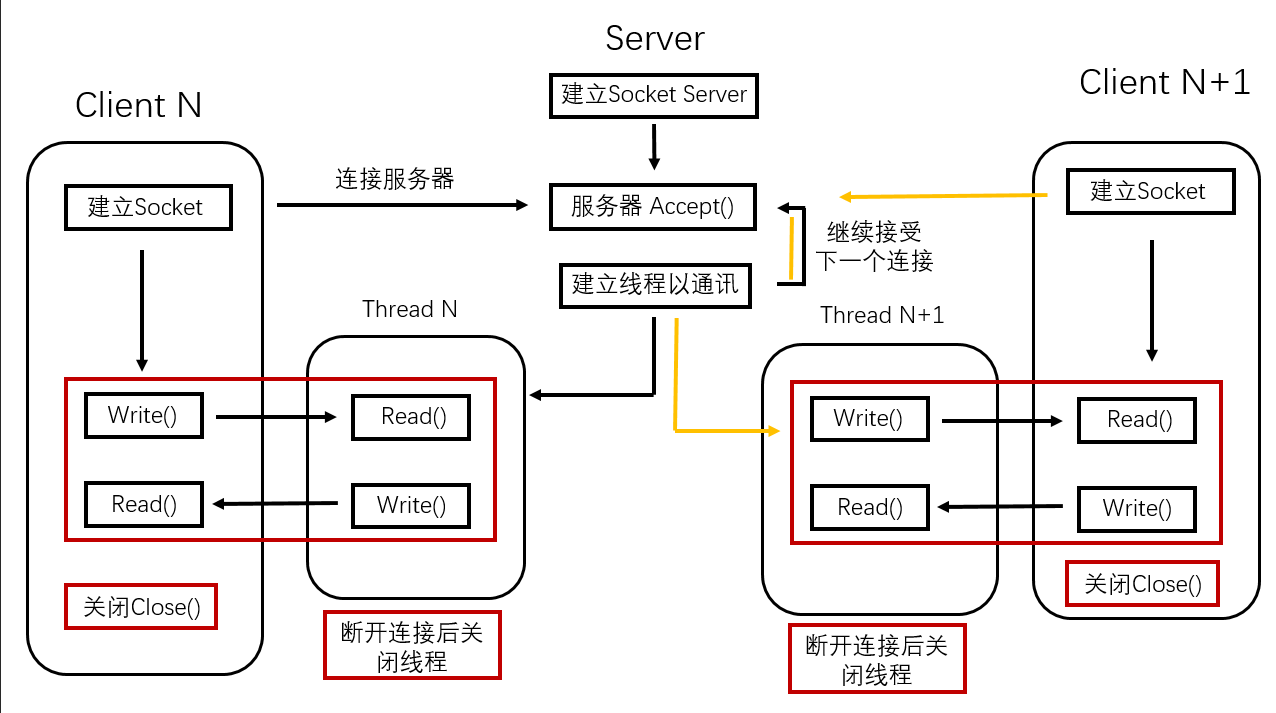
## 三、系统设计

1. **基本技术**
2. Socket 与 ServerSocket

作为一个即时通讯工具，客户端（Client）和服务器（Server）是两个必不可少的部分。首先我们就来解决服务器和客户端连接的问题。

在Internet上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个Socket，并绑定到一个端口上，不同的端口对应于不同的服务。

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个Socket。建立了一个Socket后，从这个Socket中读取I/O流，就可以实现两个程序的通讯。

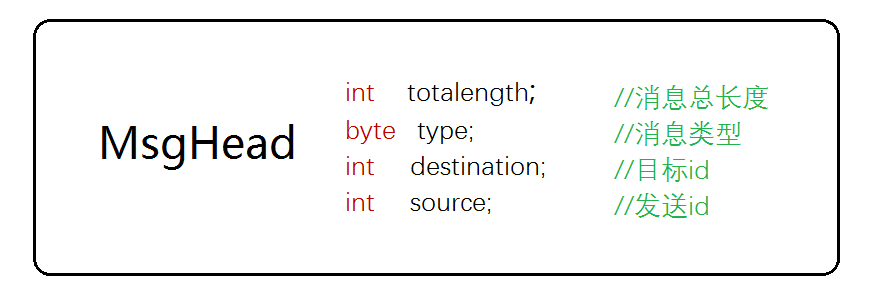
而服务器需要与多个客户端进行双向的通讯连接，并非单一的连接。于是可以通过每接收到一个Client后就开一个单独的进程，来完成与该Client的通讯步骤。这样就可以同时接受多个Client，从而实现服务器的功能。

1. 客户端建立：

Java提供了Socket类，用来建立客户端

1. 服务器建立：

Java提供了ServerSocket类，用来建立服务器。

1. **通讯协议设计**

MsgHead类是所用类的抽象父类，其中在私聊时destination为私聊目标id，在群聊时destination为群聊id号。

下表为MsgHead的具体变量的字节长度

| **MsgHead** | **消息头（13）** | **消息体** |
| --- | --- | --- |
| int totalLen | 消息总长度（4） |  |
| byte type | 消息类型（1） |  |
| int destination | 目标ID号（4） | 服务器的ID号：2000000000 |
| int source | 发送用户ID号（4） | 登录发送ID号：2000000001 |

MsgHead总长度为13字节

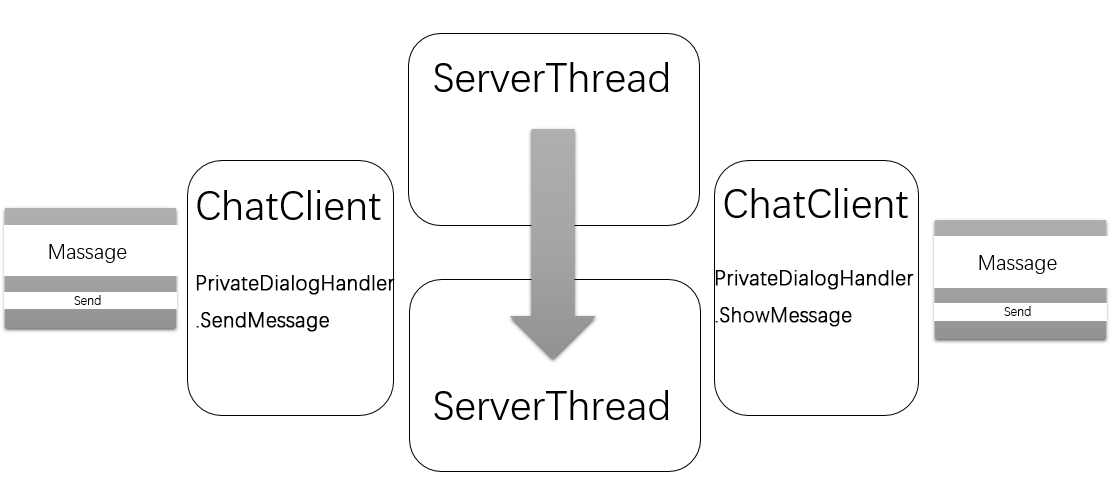
所有继承与MsgHead的子类如下所示

其中MsgParser和MsgHeadWritter并不是继承于MsgHead。

MsgParser用于解析不同消息体，需传入一个MsgHead类。

MsgHeadWritter根据不同需要封装不同的消息体并返回这个消息体。

发送一次信息的完整流程如下图所示

当点击发送按钮后会触发事件调用PrivateHandler.SendMessage方法，将消息封装成MsgChatText类并将该类写入outputStream使服务器接收。

1. **框架设计**
2. 界面层

ChatRoom的主要界面有：

登陆界面

注册界面

好友列表界面

添加好友界面

私聊界面

添加群聊界面

群聊界面

1. 业务逻辑层
2. 服务器业务逻辑层

**服务器主线程：**

| **功能** | **创建服务器** |
| --- | --- |
| 描述 | 创建服务器。 |
| 动作 | 根据指定的port创建服务器。 |
| 输入 | port(端口) |

服务器对应每一个连接的单独线程ServerThread   
这些ServerThread利用HashMap保存到线程库，key为ID号

| **功能** | **接受Client** |
| --- | --- |
| 描述 | 接受客户端，创建单独线程，并通过输入输出流通信。 |
| 动作 | 循环监听是否有client接入，若有，则创建单独线程对其进行操作。 |
| 相关 | 服务器服务线程，ServerThread |

| **功能** | **响应登陆请求&发送好友及群列表** |
| --- | --- |
| 描述 | 响应客户端发送的登陆请求。 |
| 输入 | 根据输入流中读取的数据以及通讯协议，读取userid和pwd。 |
| 行动 | 利用数据库查询输入的userid和pwd是否匹配 |
| 输出 | 根据通讯协议，向输出流输出回执信息，包含最终匹配结果。 |
| 行动 | 若登陆信息匹配，更新用户在线表。 |
| 输出 | 若登陆信息匹配，则再根据通讯协议，向输出流发送好友列表。 |
| 相关 | 数据库，通讯协议 |

| **功能** | **一对一聊天** |
| --- | --- |
| 描述 | 响应客户端的聊天请求，并将聊天信息发到指定客户端。 |
| 输入 | 根据输入流中读取的数据以及通讯协议，获取目标userid以及发送内容。 |
| 行动 | 根据目标userid以及发送内容，给对应客户端发送信息。 |
| 输出 | 通过寻找对应客户端所接入的thread，给对应客户端发送内容。 |
| 相关 | 通讯协议，serverthread |

| **功能** | **注册用户** |
| --- | --- |
| 描述 | 响应客户端的注册请求，并利用数据访问层提供接口写入数据库 |
| 输入 | 根据输入流中读取的数据以及通讯协议，获取目注册用户名以及登陆密码 |
| 行动 | 利用数据访问层提供接口写入数据库 |
| 输出 | 若注册成功，返回客户端注册好的ID号 |
| 相关 | 通讯协议，serverthread |

| **功能** | **添加好友** |
| --- | --- |
| 描述 | 响应客户端的添加好友，利用数据访问层提供接口。 |
| 输入 | 客户端发送的添加好友的ID号以及好友列表名称 |
| 行动 | 利用数据访问层提供的接口判断输入的正确性，若正确在数据库中好友列表添加内容 |
| 输出 | 添加好友的状态。 |
| 相关 | 通讯协议，serverthread |

| **功能** | **群聊** |
| --- | --- |
| 描述 | 响应客户端的聊天请求，并将聊天信息发到指定客户端。 |
| 输入 | 根据输入流中读取的数据以及通讯协议，获取目标userid以及发送内容。 |
| 行动 | 根据目标userid以及发送内容，给对应客户端发送信息。 |
| 输出 | 通过寻找对应客户端所接入的thread，给对应客户端发送内容。 |
| 相关 | 通讯协议，serverthread |

| **功能** | **添加群** |
| --- | --- |
| 描述 | 响应客户端的添加群，利用数据访问层提供接口。 |
| 输入 | 客户端发送的添加群ID号 |
| 行动 | 利用数据访问层提供的接口判断输入的正确性，若正确在数据库中群列表添加内容 |
| 输出 | 添加群后的状态。 |
| 相关 | 通讯协议，serverthread |

| **功能** | **新建群** |
| --- | --- |
| 描述 | 响应客户端的新建群，利用数据访问层提供接口。 |
| 输入 | 客户端发送的添加群ID号 |
| 行动 | 利用数据访问层提供的接口判断输入的正确性，若正确在数据库中群列表添加内容 |
| 输出 | 新建群后的状态。 |
| 相关 | 通讯协议，serverthread |

服务器若处理的是群聊的消息，则会将消息群发给此群中的所有成员，此时目标id存储的是群聊的id

1. **客户端业务逻辑层**

| **功能** | **连接服务器** |
| --- | --- |
| 描述 | 客户端连接服务器，确认服务器可以连接。 |
| 输入 | 读取类中的ServerIP和port。 |
| 动作 | 根据ServerIP和port，开启Socket，获取输入输出流。 |
| 输出 | 能否连接到服务器 |

| **功能** | **注册** |
| --- | --- |
| 描述 | 输入用户名和密码 |
| 输入 | NickName和Password |
| 动作 | 将注册请求，打包成消息，发送给服务器。 |
| 输出 | 根据服务器返回结果显示ID号或者显示注册失败。 |

| **功能** | **登陆服务器** |
| --- | --- |
| 描述 | 客户端申请登陆，确认用户密码是否正确。 |
| 输入 | 用户输入的userid和password |
| 动作 | 将userid和password根据通讯协议传输到服务器，并接收服务器回信。 |
| 输出 | 用户名密码是否正确 |
| 相关 | 通讯协议 |

| **功能** | **获取好友列表** |
| --- | --- |
| 描述 | 完成登陆后，从服务器获取好友列表，并显示出好友界面 |
| 动作 | 从服务器获取好友列表，并通过给好友列表界面对象。 |
| 相关 | 通讯协议，好友列表界面对象 |

| **功能** | **获取群列表** |
| --- | --- |
| 描述 | 完成登陆后，从服务器获取群列表，并显示出群界面 |
| 动作 | 从服务器获取群列表，并通过给群列表界面对象。 |
| 相关 | 通讯协议，群列表界面对象 |

| **功能** | **一对一聊天** |
| --- | --- |
| 描述 | 在好友列表中点击一个好友，开始一对一聊天，可以发送/接收信息 |
| 动作 | 根据通讯协议，向指定用户发送信息 |
| 输入 | 发送的目标userid以及发送内容（聊天界面传入） |
| 动作 | 根据通讯协议，读取信息来源用户，通过聊天界面显示内容。 |
| 输出 | 聊天内容传给聊天界面对象 |
| 相关 | 通讯协议，聊天界面对象 |

| **功能** | **添加好友** |
| --- | --- |
| 描述 | 添加好友到列表 |
| 输入 | 好友的ID号和列表名 |
| 动作 | 将请求打包成消息，发送给服务器，等待服务器返回结果 |
| 输出 | 根据结果弹出结果消息框，更新好友列表。 |

| **功能** | **群聊** |
| --- | --- |
| 描述 | 在好友列表中点击一个群，开始群，可以发送/接收信息 |
| 动作 | 根据通讯协议，向指定群发送信息 |
| 输入 | 发送群的id以及发送内容（聊天界面传入） |
| 动作 | 根据通讯协议，读取信息来源用户，通过聊天界面显示内容。 |
| 输出 | 聊天内容传给聊天界面对象 |
| 相关 | 通讯协议，聊天界面对象 |

| **功能** | **添加群/新建群** |
| --- | --- |
| 描述 | 添加群到列表 |
| 输入 | 群的ID号或新建的群名 |
| 动作 | 将请求打包成消息，发送给服务器，等待服务器返回结果 |
| 输出 | 根据结果弹出结果消息框，更新群列表。 |

1. 数据访问层

本次数据库选用SQLite数据库，数据库结构如下。

数据访问层提供UserModel，其中包括

根据ID号获取用户内容

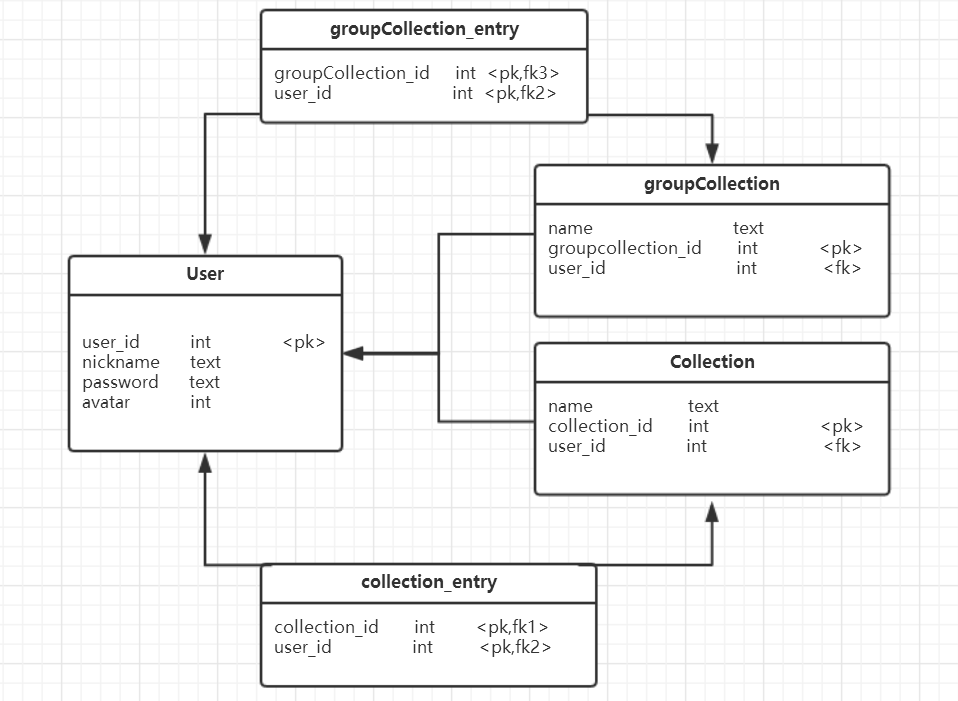
验证用户信息

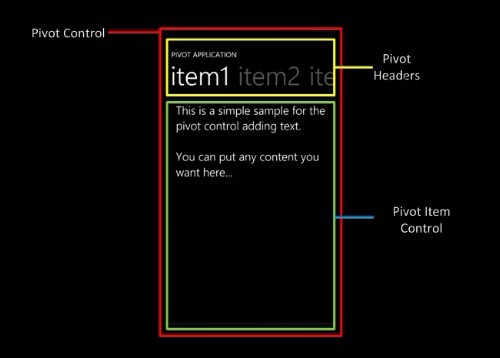
获取用户好友列表

增加好友

增加群组

获取群组列表



1. **UI设计**

聊天室所有窗口采用Metro UI设计  
Metro UI的特点：强调信息本身   
参考资料：[http://www.csdn.net/article/2012-02-01/310896/1](http://www.csdn.net/article/2012-02-01/310896/1" \t "_blank)

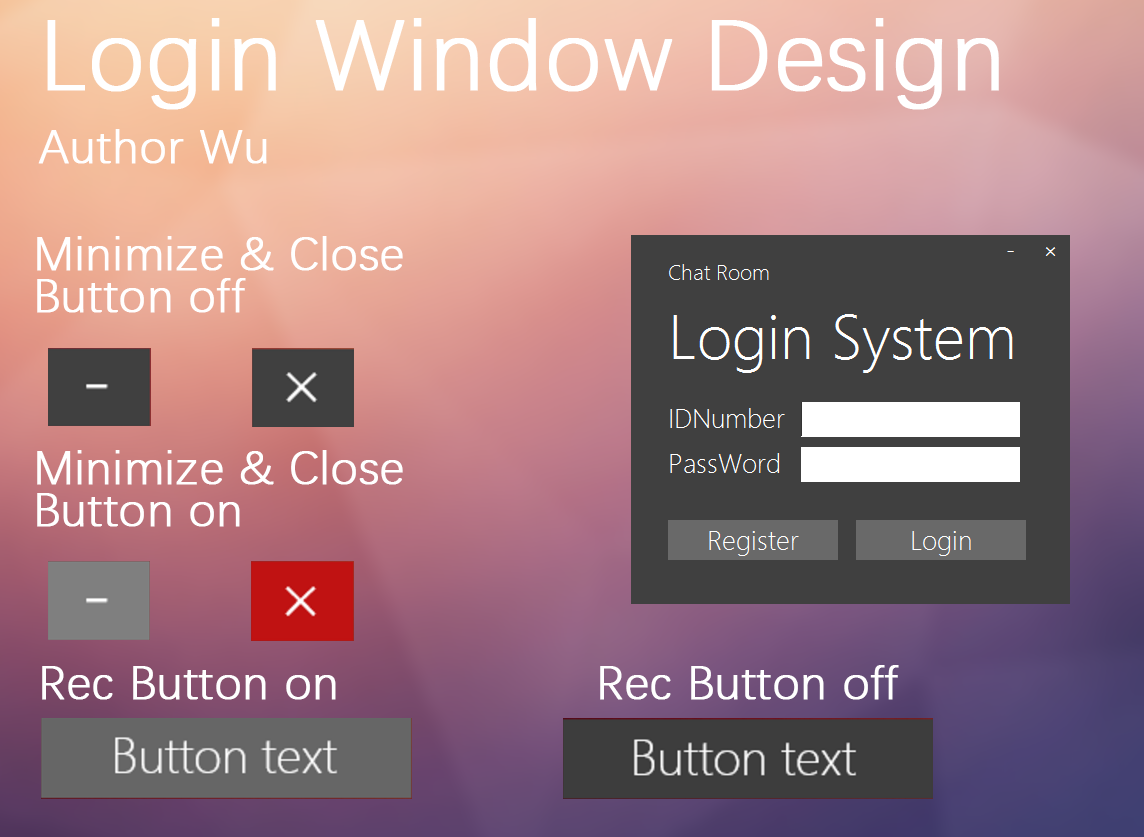
1. 窗体设置无边框以及拖动选项

为了彻底改变Swing的风格，实现Metro UI的界面，将原有的边框以及按钮重写是必不可少的。

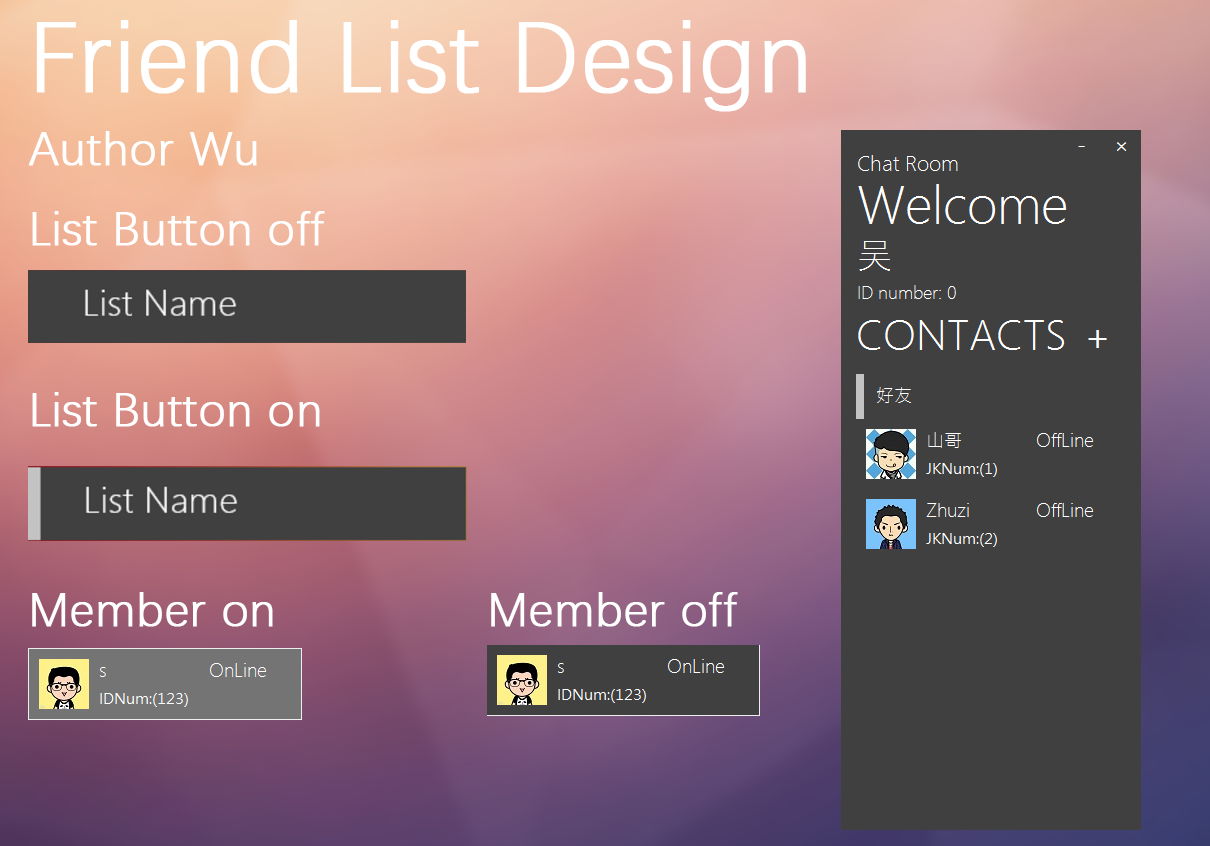
1. 关闭、缩小、选择按钮的重写

因为选择了无边框，则原来的关闭以及缩小按钮都已经不能使用了，于是可以继承JButton，自己重写一个关闭，缩小按钮。同时为了配合Metro UI，选择按钮也需要重写注意实现鼠标移进、移出、按下的颜色改变（可以添加其他效果）。

最终实现效果如下



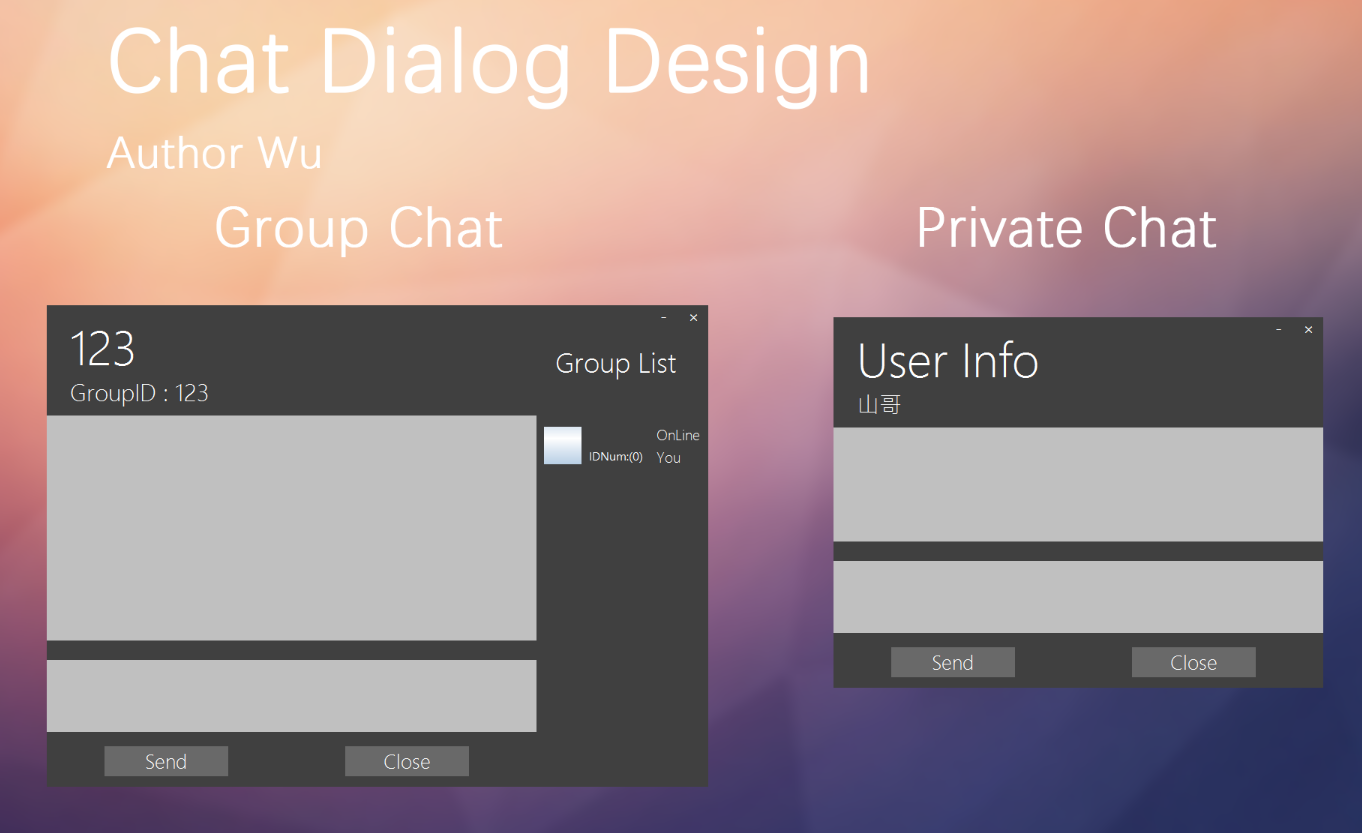
登录界面



登录后界面



注册和添加好友界面



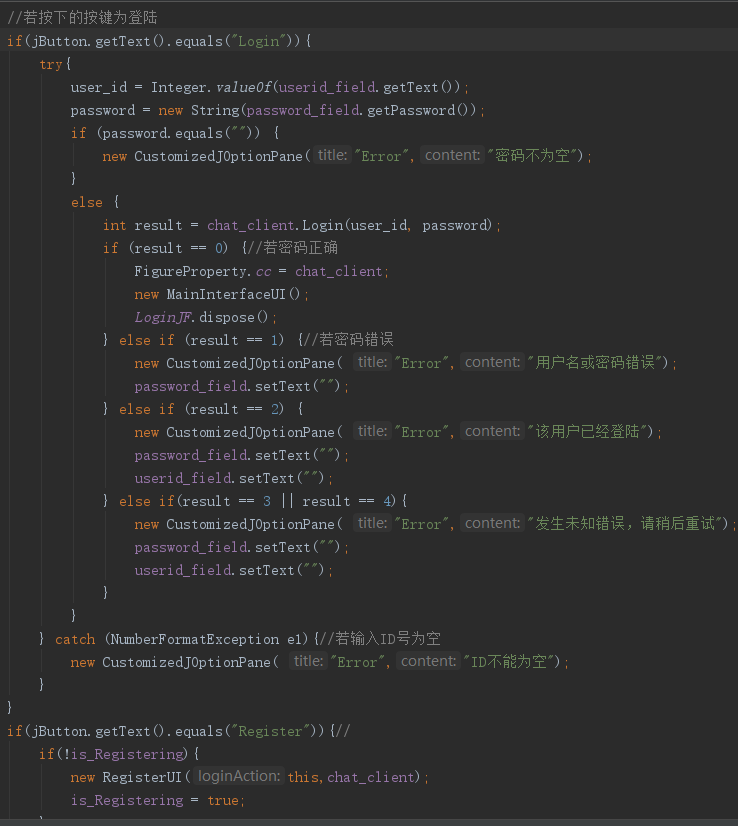
聊天界面

## 四、系统实现

1. **客户端**
2. **界面跳转**

客户端启动时首先跟服务器建立socket。然后弹出登录界面LoginUI并监听登录界面的按钮，然后程序会根据用户不同的操作来执行不同的响应操作。

示例代码如下

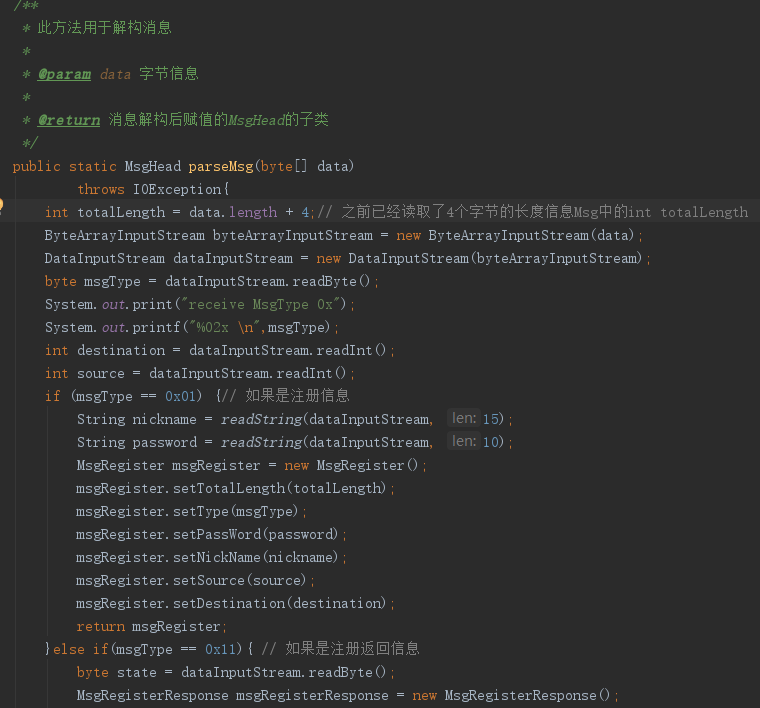


1. **数据传输协议**

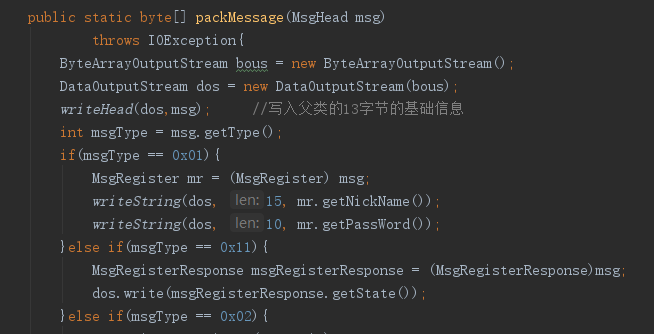
客户端与服务器进行数据传输时，每次会调用MsgParser中的方法从InputStream中读取一个4个字节的int值。该值表示此消息体的总长度，再依次读取类型（1字节）、目标ID（4字节）、源ID（4字节）。

就可以做到区别不同消息体，之后再根据不同消息体的结构进行解构

部分代码如下

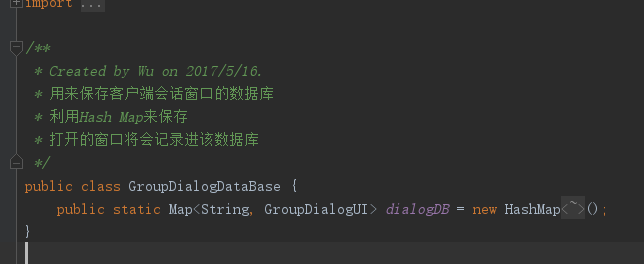
其中MsgHeadWritter类的思路与MsgParser思路大致相同

根据不同需求将消息打包

部分代码如下，其中writeString方法是将传入的String类型的值以byte类型写入DataOutputStream中。

1. **本地存储**

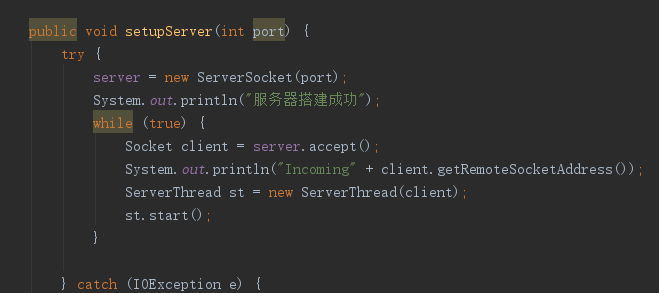
客户端本地保存了一个Map来存储聊天窗口是否存在

每次有新消息来的时候先检查是否打开了窗体，没有再弹窗。

客户端还建立了一个UserInfo类来保存自身的信息

1. **服务器**
2. **与客户端交互**

服务器主线程启动后会不断检测是否有新的socket连接，若有则开一个新的服务器服务线程来与这个socket交互。

客户端上线后服务器会首先判断该用户是否在线，若不在线会给客户端发送该用户的好友列表。

下图为首先判断是否在线，processLogin为处理客户端发来的登录或注册信息

下图为登录处理，broadcastState用于发送好友列表

登录后就会继续不断调用processChat方法来处理加好友、聊天等消息。

1. 与数据库交互

数据库位于服务器端，服务器每次需要连接数据库时新建一个DBconnection的实例用于连接数据库。此类实现了数据库的查询、更新、删除等必要操作。

部分代码如下：

## 五、问题及解决

1. **好友列表的实现**

要实现好友列表，第一个想到的就是树状结构。但是要实现相对漂亮的UI，利用树状结果不太好实现。

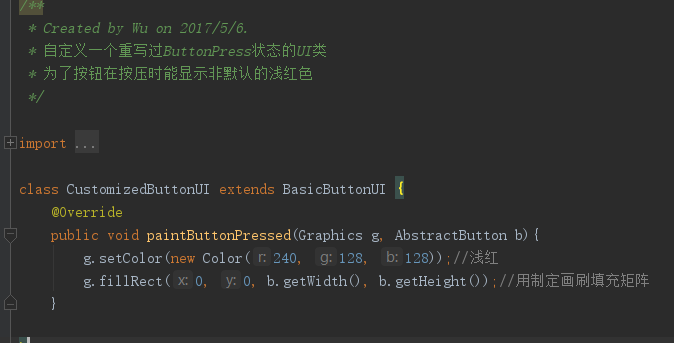
经过多次尝试，最终确定用JScrollPane来完成这个可能需要拖动的好友列表。但是利用JScrollPane后，不宜布局，若利用JPanel来制作好友，其大小不好固定。最终重写滚动条，具体实现见代码。好友列表的表示最终使用了二维数组

1. **UI的重写**

因为选择了无边框所以要自己写窗口拖动和关闭、缩小、选择按钮。

但是在重写关闭按钮时遇到了问题。Java中默认继承JButton的按钮在按压（Pressed）状态下的背景色都是灰白的，但是关闭窗体的按钮在按压状态下应该是浅红，所以刚开始在按钮的事件监听器中添加了在按钮被按压时触发的事件。该事件使按钮背景色变为浅红。但是这个方法并不奏效。通过查阅资料后，发现该方法需要重写ButtonUI中的paintButtonPressed 方法，再使关闭按钮使用这个自定义的ButtonUI类。最终可以达到想要的效果。

重写代码如下

1. **聊天窗口重复打开问题**

最开始不知道如何判断聊天窗口是否重复打开，最后通过在本地建立一个Map来存储打开的聊天窗口。

1. **其他逻辑及异常问题**

在编写程序问题时还出现过许多逻辑及异常错误。其中排错最久的一条为在客户端登录后每次接收好友列表时客户端异常崩溃的错误，仔细看了很多遍代码感觉逻辑没有错。最终发现是在发送预先存在数据库里的好友列表的名称时，发送的中文名称的字节长超过了所限定的长度导致抛出异常使程序终止，但我在写这个异常的时候只是用throws抛出而没有用try/catch，所以使我在排错耗费的很多时间。

## 六、系统特色与创新

1.重写了BasicButtonUI 和 BasicScrollBarUI以获得更好的UI效果

2.服务器端采用多线程以实现与多个客户端交互

3.使用了传输协议，协议都继承于MsgHead类

4.实现了单聊群聊及好友系统

5.界面相对于Java 的基本UI相对清爽

## 七、个人心得与体会

1. 在重写ButtonUI时来改变Button默认按压状态下的颜色时，查阅了很多问答资料也没有找到自己想要的结果。最后只好去查阅Java的API，并发现应该重写paintButtonPressed的方法。让我意识到了阅读Java API的重要性。
2. 在之前学习异常处理时，老师总是一带而过，这也让我有些不理解如try/catch在一个程序中的作用。通过遇到在上文提到的最后一个问题即抛出异常使程序终止，我有些理解了try/catch对于程序健全性的重要性以及在排错时的便利性。
3. 通过自己亲身去重写UI以及功能，明白了自己与平常使用的软件的用户友好度以及功能上的差距。