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

2. 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。

3. 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。

4. 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。

5. 发送的消息实现

发送消息可以通过类继承Serializable接口序列化发送，然后再用FileInputStream接收之后反序列化。

并非所有的对象都可以序列化，,至于为什么不可以，有很多原因了,比如：

1.安全方面的原因，比如一个对象拥有private，public等field，对于一个要传输的对象，比如写到文件，或者进行rmi传输 等等，在序列化进行传输的过程中，这个对象的private等域是不受保护的。

2. 资源分配方面的原因，比如socket，thread类，如果可以序列化，进行传输或者保存，也无法对他们进行重新的资源分 配，而且，也是没有必要这样实现。

这个确实是个问题.当年在学习Java基础的时候,也问过类似的问题.对呀.为何要进行序列化呢.不进行序列化,我的程序不跑的好好的吗?你想要什么结果,我也能给解决不是.我想说确实是这样,如果你的程序与网络无关,那很好你已经可以摒弃它了.

那下面我来简单分析下为何java需要进行序列化呢.

首先我们要明白,序列化是做什么作用的.java序列化: 以特定的方式对类实例的瞬时状态进行编码保存的一种操作.(可能不是很精确,咱不是搞学术的,看懂即可).从此定义可以看出,序列化作用的对象是类的实例.对实例进行序列化,就是保存实例当前在内存中的状态.包括实例的每一个属性的值和引用等.

既然后序列化,便会有反序列化.反序列化的作用便是将序列化后的编码解码成类实例的瞬时状态.申请等同的内存保存该实例.

从上述定义可以发现,序列化就是为了保存java的类对象的状态的.保存这个状态的作用主要用于不同jvm之间进行类实例间的共享.在ORMaping中的缓存机制,进行缓存同步时,便是常见的java序列化的应用之一.在进行远程方法调用,远程过程调用时,采用序列化对象的传输也是一种应用...当你想从一个jvm中调用另一个jvm的对象时,你就可以考虑使用序列化了.

简而言之:序列化的作用就是为了不同jvm之间共享实例对象的一种解决方案.由java提供此机制,效率之高,是其他解决方案无法比拟的.自家的东西嘛.

<http://blog.csdn.net/dreamtdp/article/details/15378329>

最后并没有采用序列化的方法，而是自己写协议，客户端和服务器之间约定好不同协议的读取顺序，按不同顺序readByte，readInt得到最后的消息体。