**Java程序设计实验报告**

计算机类2208班 姓名 张艺 学号2020303210226

实验环境：Windows 10

实验目的：学习Socket编程： 通过编写Java QQ程序，学习如何使用Socket实现客户端和服务端之间的通信。

1、理解多客户端通信： 实现多客户端模式下的通信，通过服务端中转实现客户与客户之间的单独通信，加深对网络编程的理解。

2、实现并行通信模式： 理解并实现端到端的通信，确保信息的发送不受另一端的影响，培养并行编程的思维。

3、GUI设计与实现： 学习如何使用Java Swing或其他GUI库创建图形用户界面，实现登陆界面、登陆失败提示界面、好友管理界面和聊天界面，提高对图形界面设计和实现的能力。

4、文件传输实现： 了解并实现端到端的文件传输功能，加深对网络数据传输的理解，培养数据传输的相关技能。

5、群聊天实现： 探索群聊天的实现，了解在网络编程中如何处理多用户之间的通信，加强对多用户系统的设计和管理的理解。

### 实验内容：

**依据Java-QQ 实验说明以及评分标准**

**源代码：共 3 个 Java 文件，SingleTalkServer.java, SingleTalkClient.java**

**和 MultiTalkServer.java**

**一． 运行源代码成功，10 分**

**二． 在多客户端模式下，实现客户与客户的单独通信，要求信息通**

**过服务端中转，20 分**

**三． 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，端的信息发送不受另一端的影响），20 分**

**四． 实现界面，50 分**

**1. 登陆界面，登陆失败提示界面；**

**2. 简单的好友管理界面，不需要添加、删除好友功能；**

**3. 聊天界面，请参照 QQ；**

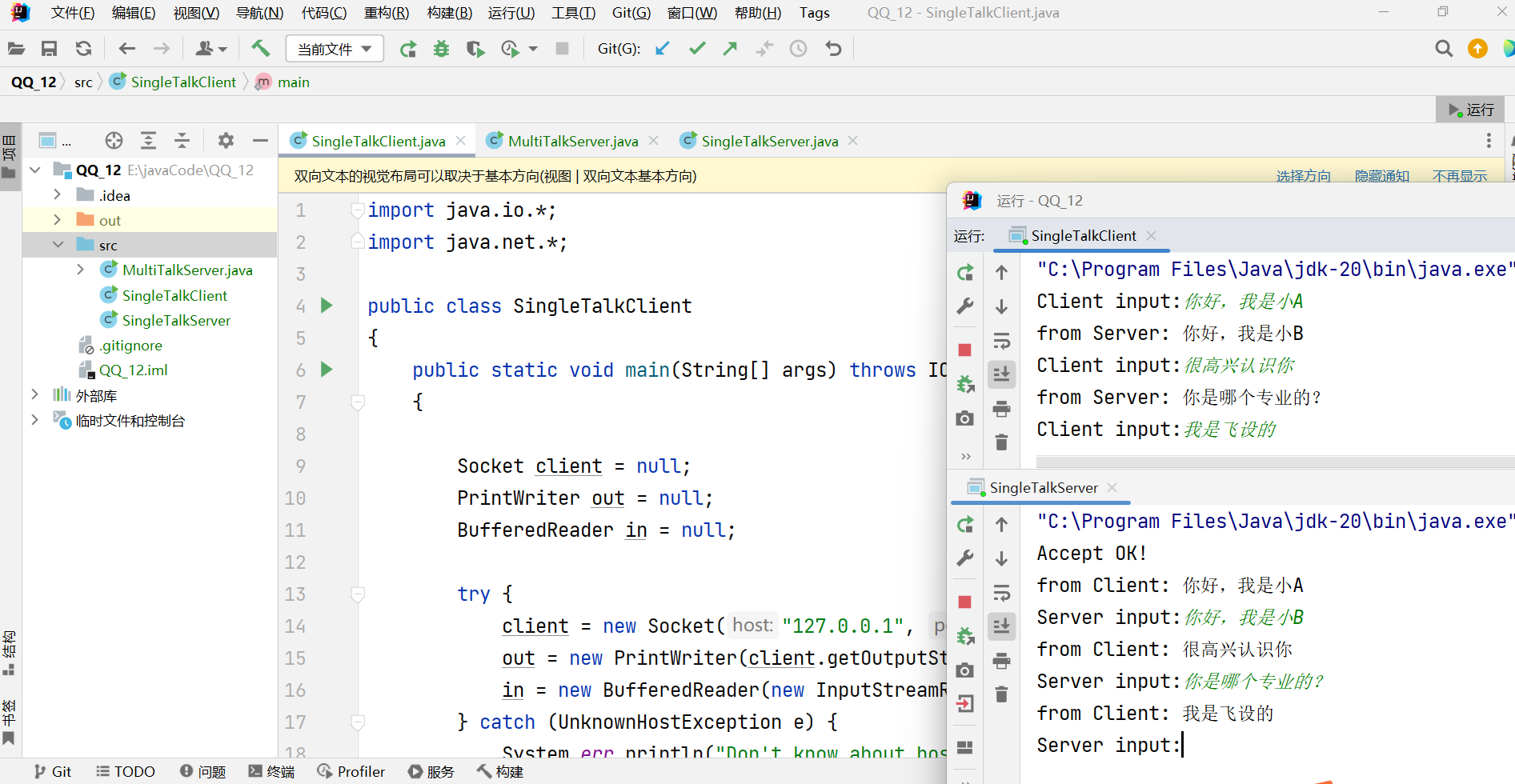
**五． 实现端到端的文件传输，30 分，选做**

**六． 实现群聊天，30 分，选做**

**（备注，考勤分数另行计算）**

### 实验步骤：

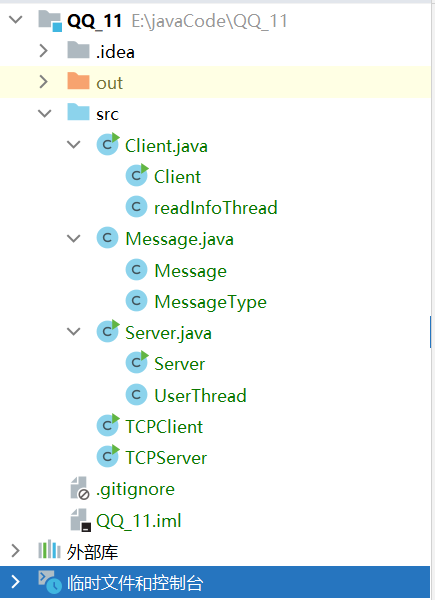
**一． 运行源代码成功，10 分**



**二． 在多客户端模式下，实现客户与客户的单独通信，要求信息通**

**过服务端中转，20 分**

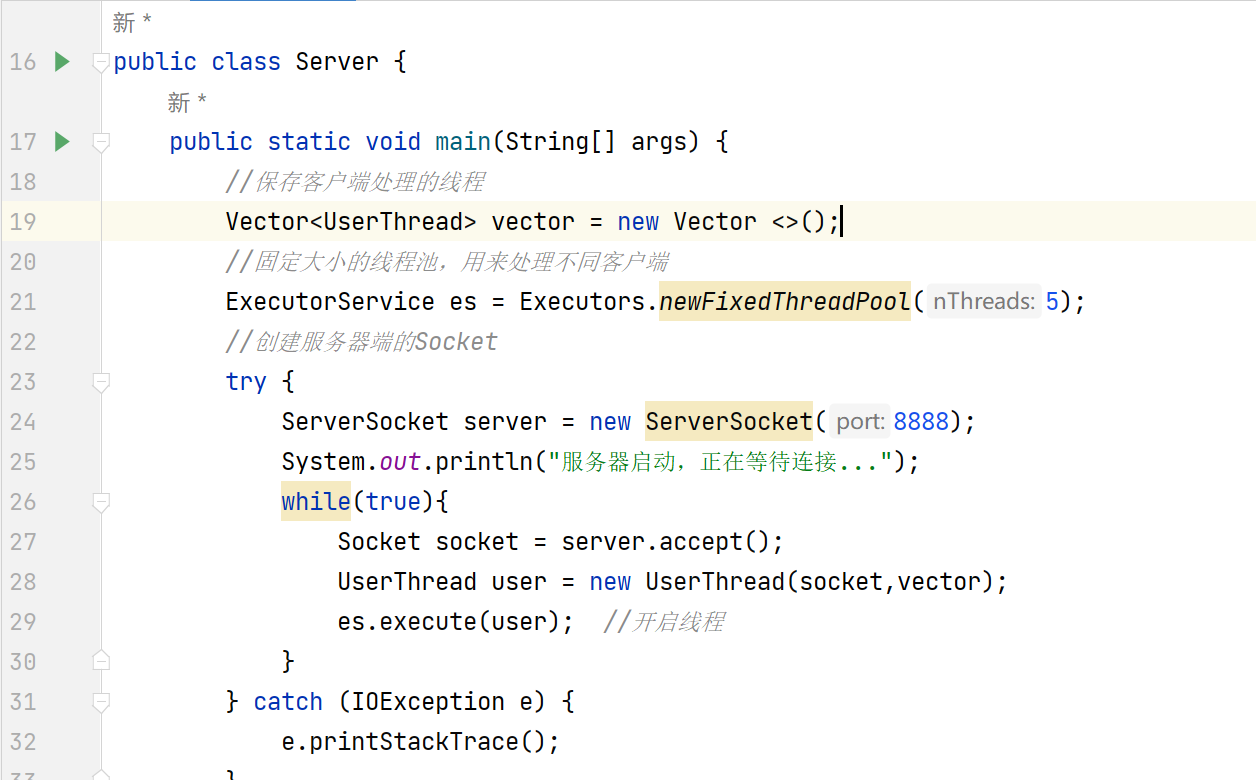
项目结构如图：



整体思路：

①Server服务端：

1. 创建一个Vector类型的集合，用于保存处理每个客户端连接的线程
2. 使用Executors.newFixedThreadPool创建一个固定大小为5的线程池，用于管理并发处理多个客户端连接
3. 持续接收客户端的请求，使用while（true），当有客户端请求连接时，accept（）方法会阻塞并返回一个与客户端建立连接的socket对象
4. 创建一个UserThread对象，负责处理与客户端的通信，将新创建的线程对象和保存线程的Vector对象传递给UserThread的构造函数，重写run方法，进行登录和发送消息的处理并捕获异常

核心代码：

重写run（）方法：

**服务端作为中转站，通过遍历Vector，将接收到的消息发送给指定用户**





②Client客户端：

1. 创建一个单线程池 Executors.newSingleThreadExecutor()
2. 利用Socket创建客户端，匹配本机ip地址和对应设置的端口，构建输出和输入流
3. 进行客户登录处理，客户自己输入自己的名字
4. 发送消息：将要发送的消息封装成一个Message对象，使用主线程发送消息，利用while（flag）不断发送
5. 读入消息：利用while（flag）不断读入，在控制台打印读到的发送者+消息，同时判断没有数据时关闭流

核心代码：

创建客户端线程：



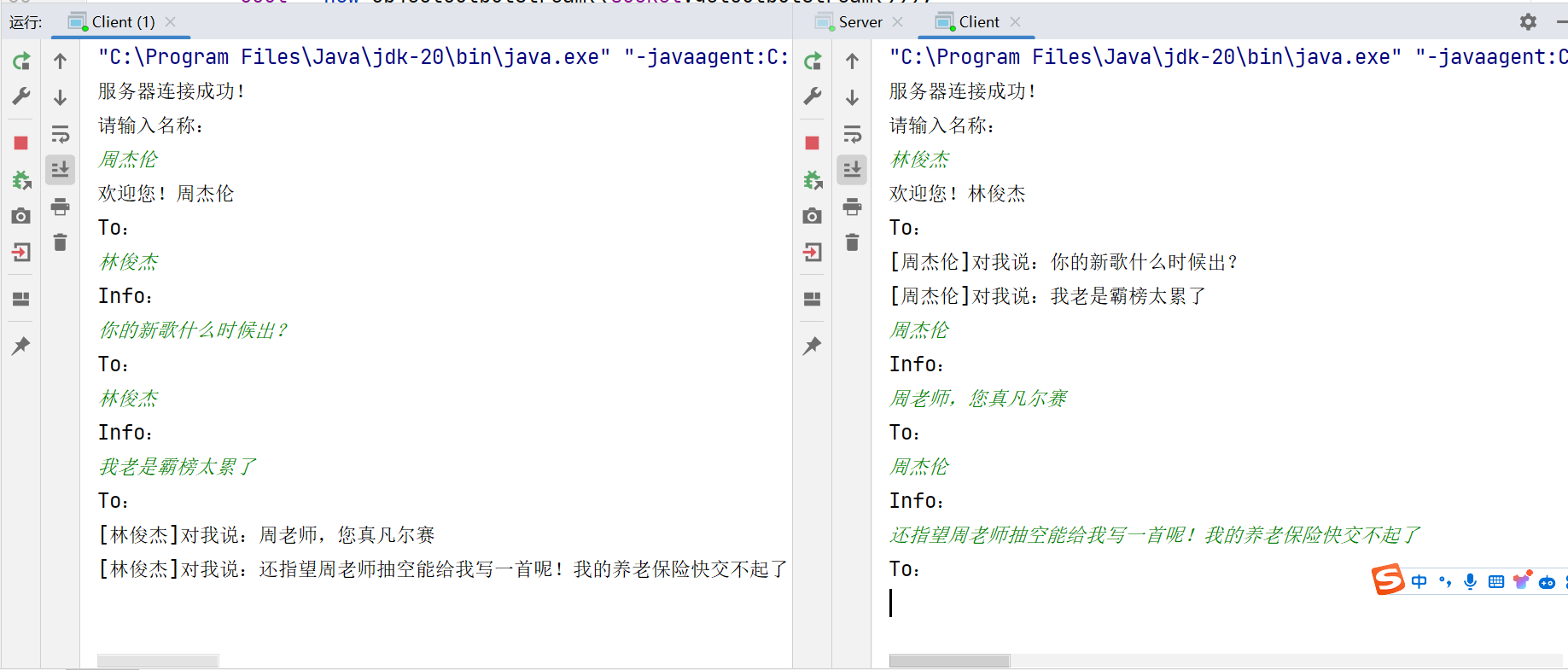
封装Message消息对象：



输出消息并打印在控制台：



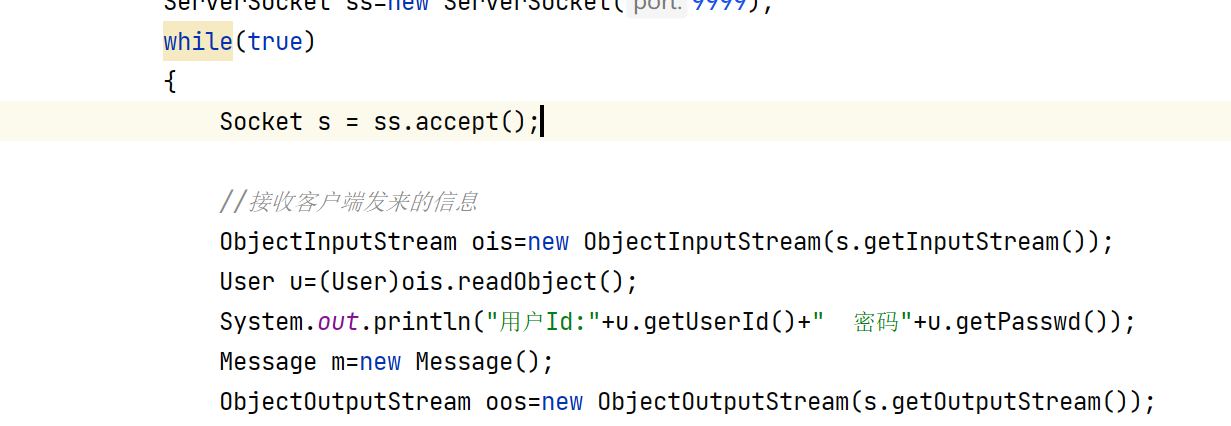
运行结果：



**三． 端到端的通信，实现并行通信模式（不再是你说一句，我说一**

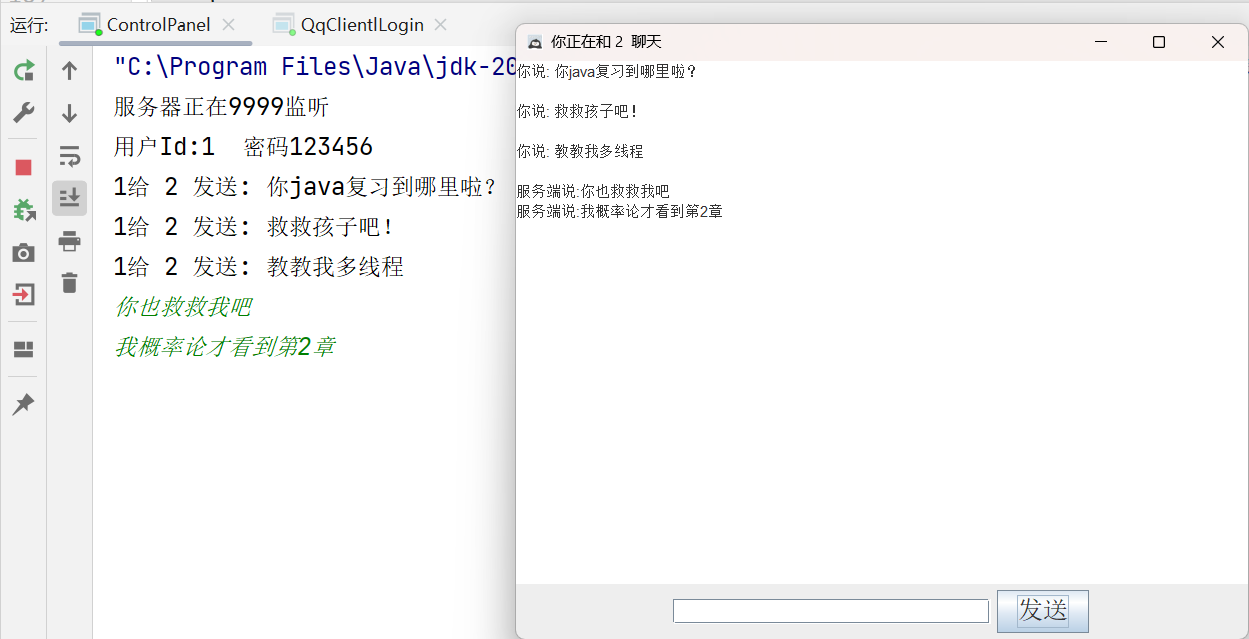
**句，端的信息发送不受另一端的影响），20 分**

多线程处理： 代码运行在一个多线程环境中，每个客户端连接可能在自己的线程中执行。这样，一个客户端在阻塞的同时不会影响其他客户端的运行，因为阻塞是在当前线程内进行的。



每当有客户端连接时，都会创建一个新的线程来处理该客户端的连接。这样可以避免主线程在 ss.accept() 处阻塞

运行结果：

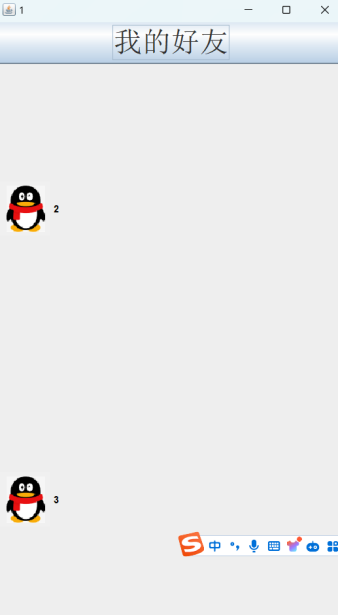


**四． 实现界面，50 分**

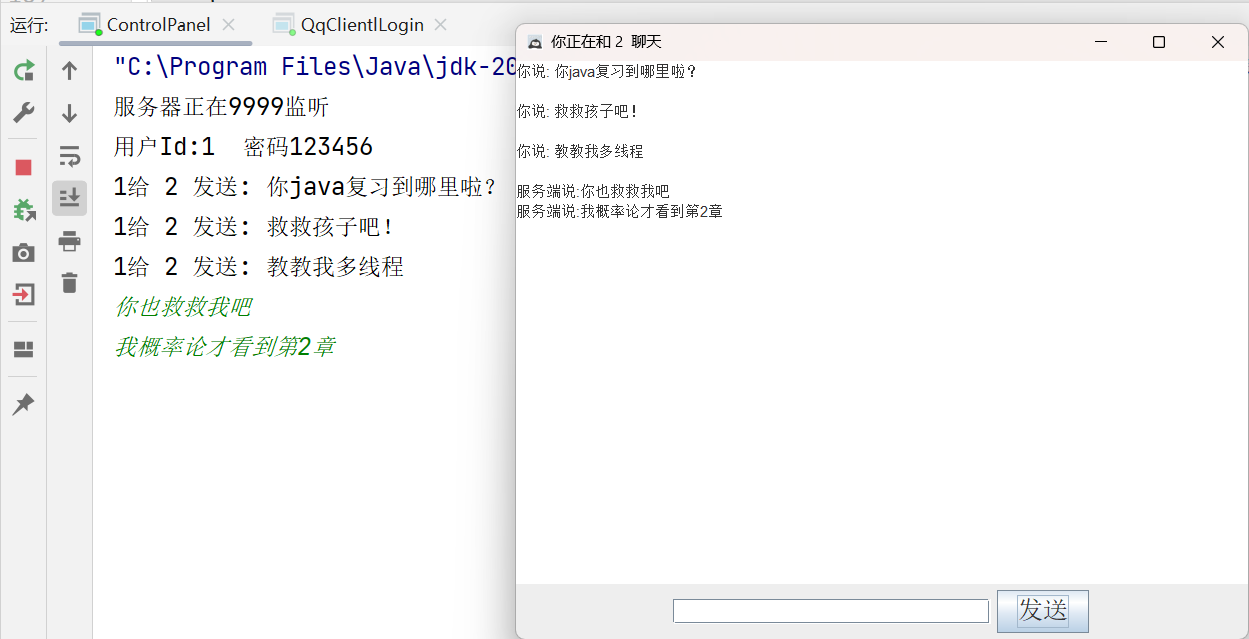
**1. 登陆界面，登陆失败提示界面；**



1. **简单的好友管理界面，不需要添加、删除好友功能；**

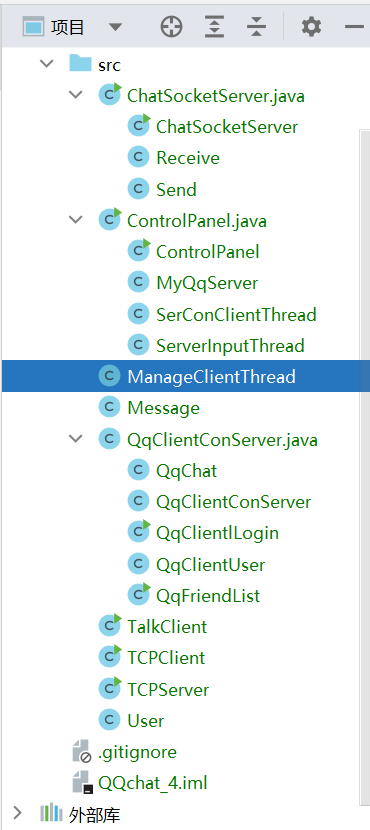


1. **聊天界面，请参照 QQ；**



**核心代码：**

项目目录**：**



**①服务端：ControlPanel**

登录界面通过设置MesType\_log返回值来判断是否成功

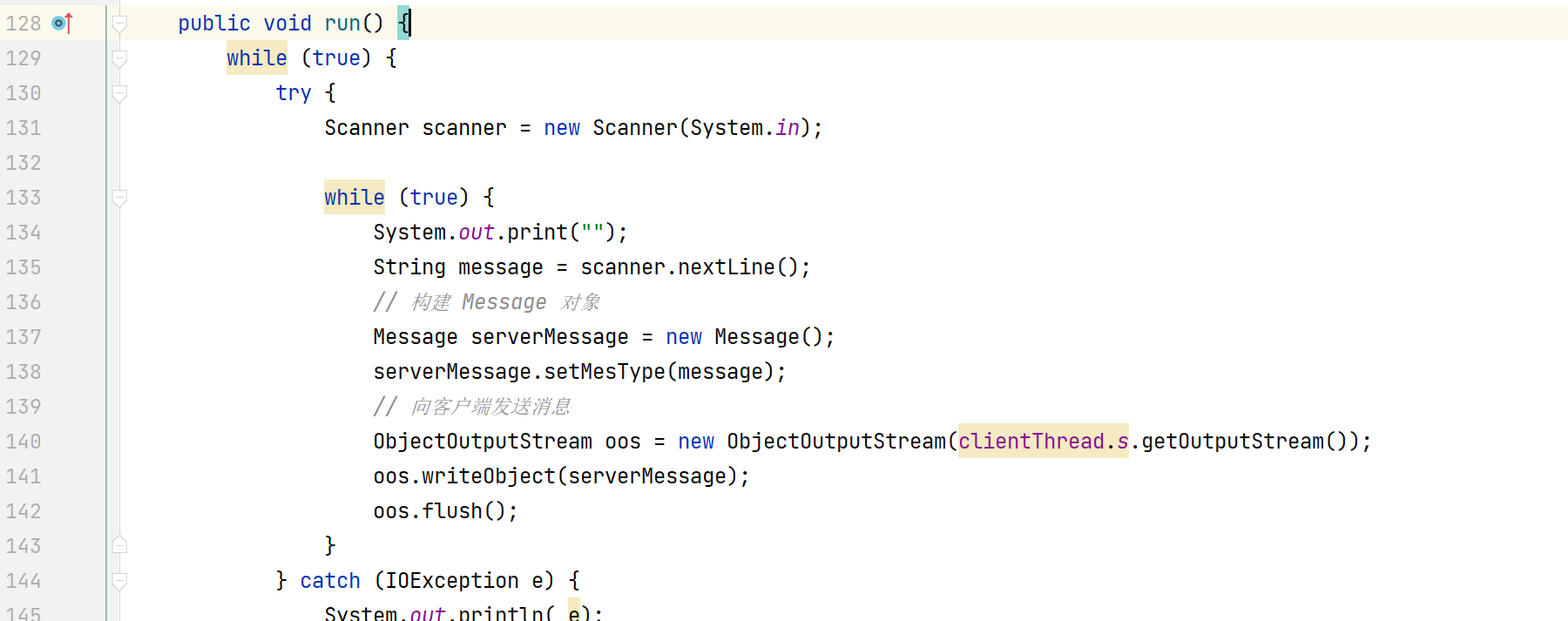




服务端和客户端的通信线程：

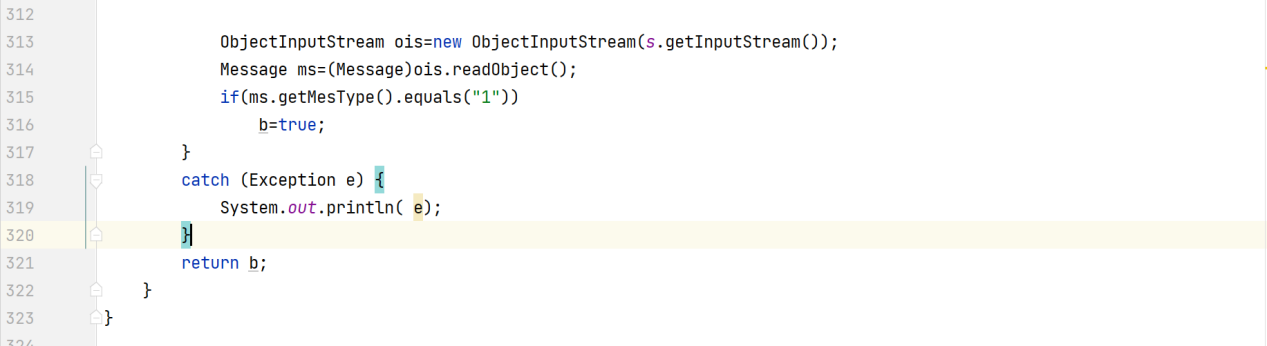


ServerInputThread类：通过ObjectOutputStream流向客户端发送消息

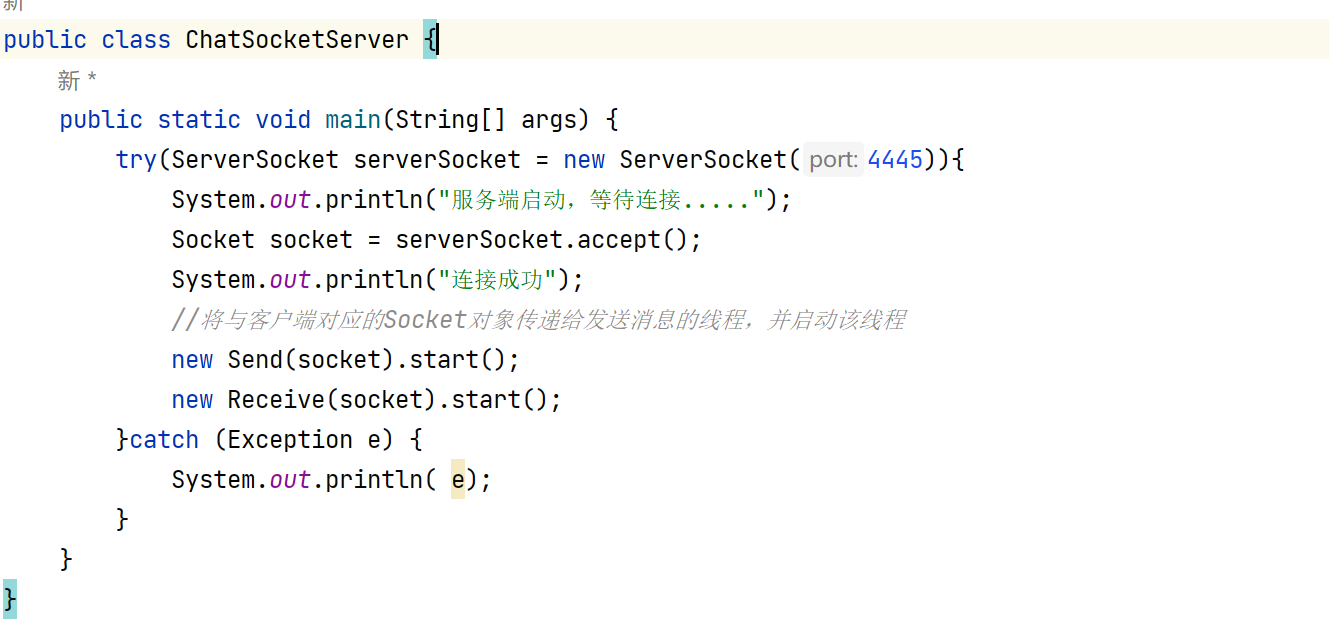


**②客户端：QQClient：**





**③信息交互：ChatSocketServer**

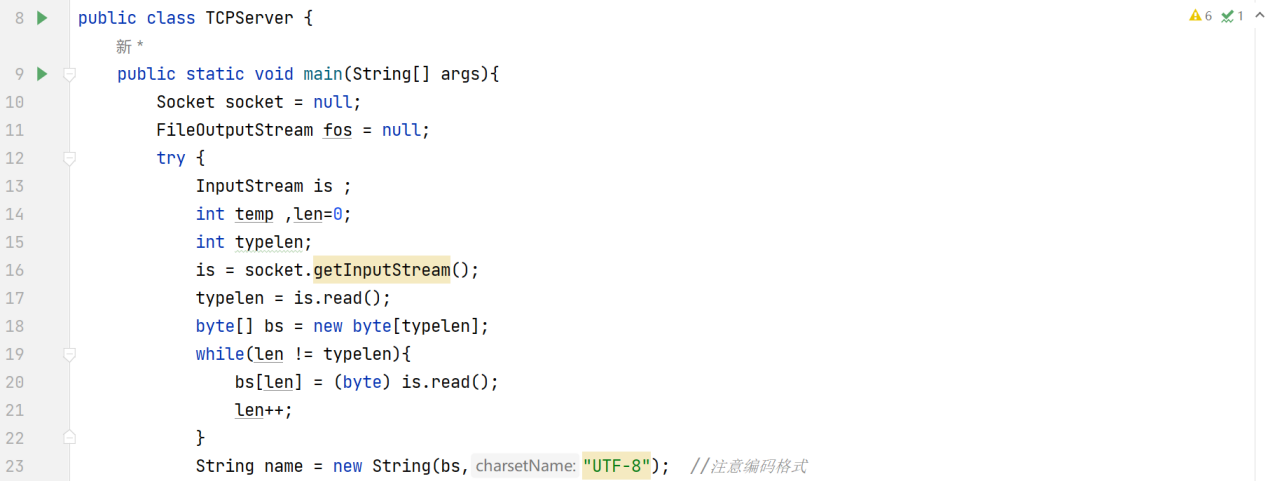


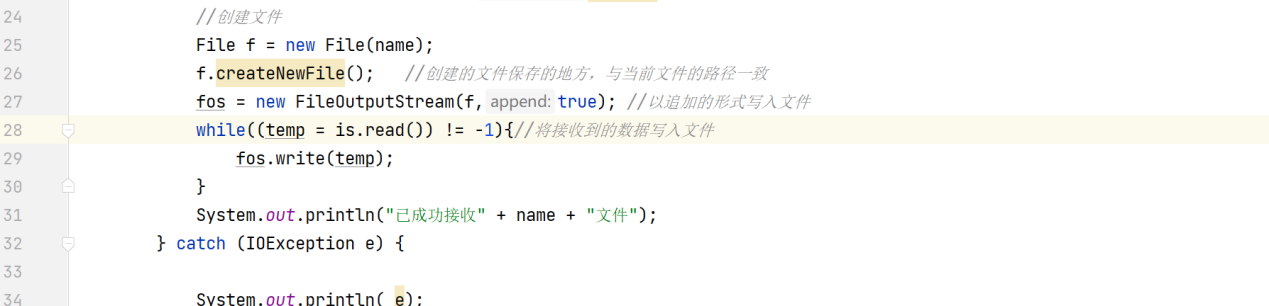
**五． 实现端到端的文件传输，30 分，选做**

思路：写两个类为传输文件的服务端和客户端，将文件地址的选择设置为用户输入，然后将端口号与主机地址与qq聊天对应，从获取的消息中判断是否有“//”，如果有，按照“//”用 FileInputStream将文件读出，再创建一个OutputStream来写入新文件。

核心代码：

**①TCPServer：**



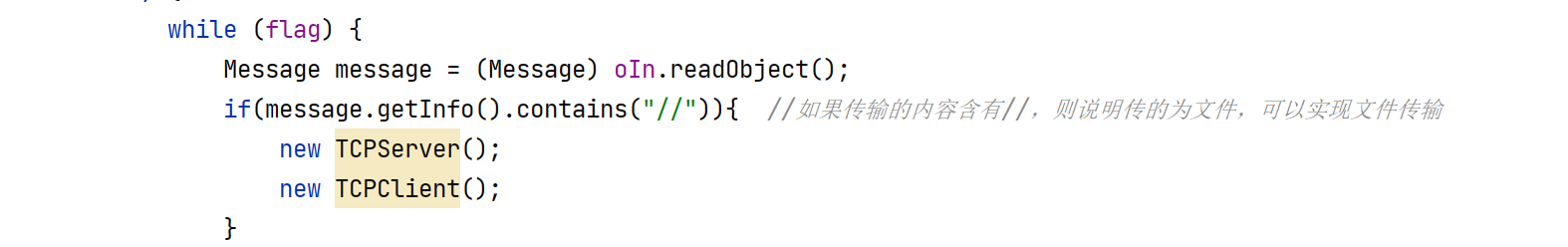


**②TCPClient：**





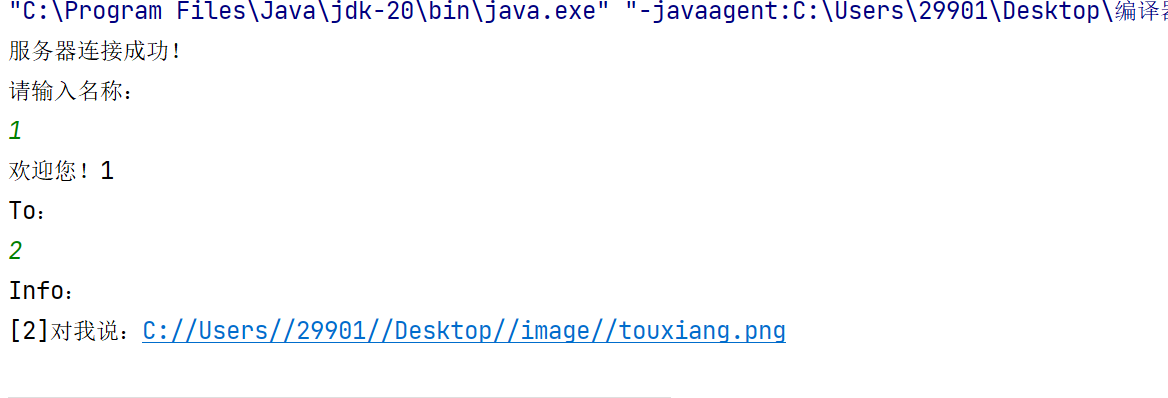
**③Client中的判断是文件代码：**



**运行结果：**

**多个客户端互传文件：**

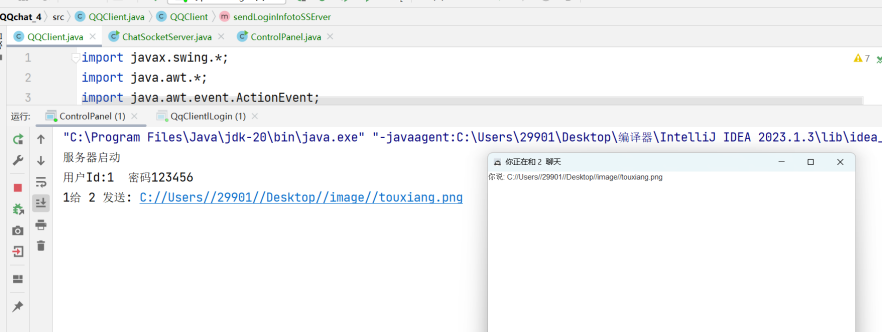


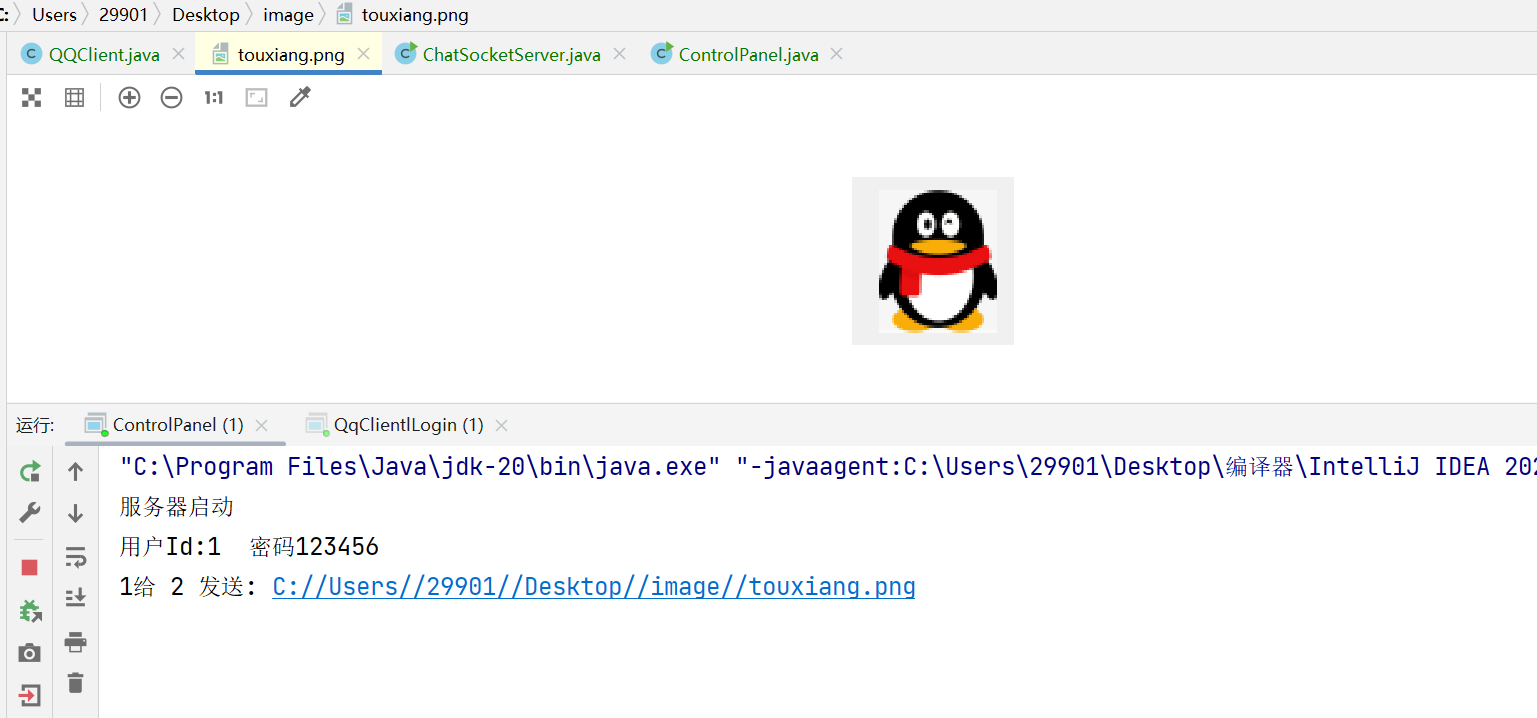




**客户端和服务端传文件：**

**客户端为1，服务端为2**





**六． 实现群聊天，30 分，选做**

实现思路：在任务二实现多个客户端聊天通过服务器中转的基础上，将聊天室设置在了服务端控制台上，通过客户端在输入“To”时判断消息是私聊还是群发，如果是To chat\_room则为群发广播消息，在服务端判断读入流消息中的发送对象是否为chat\_room如果是，则在遍历集合中所有的客户端之前将消息拦截，广播给所有用户，打印在聊天室--服务端控制台上，如果不是，继续之前操作，遍历寻找消息接收对象

**核心代码：**

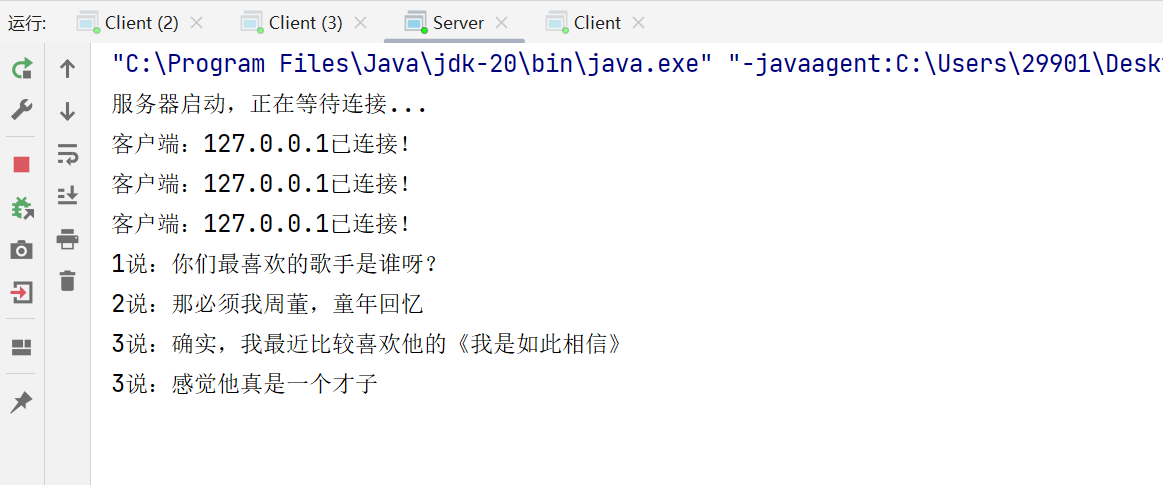
**①Server端：**

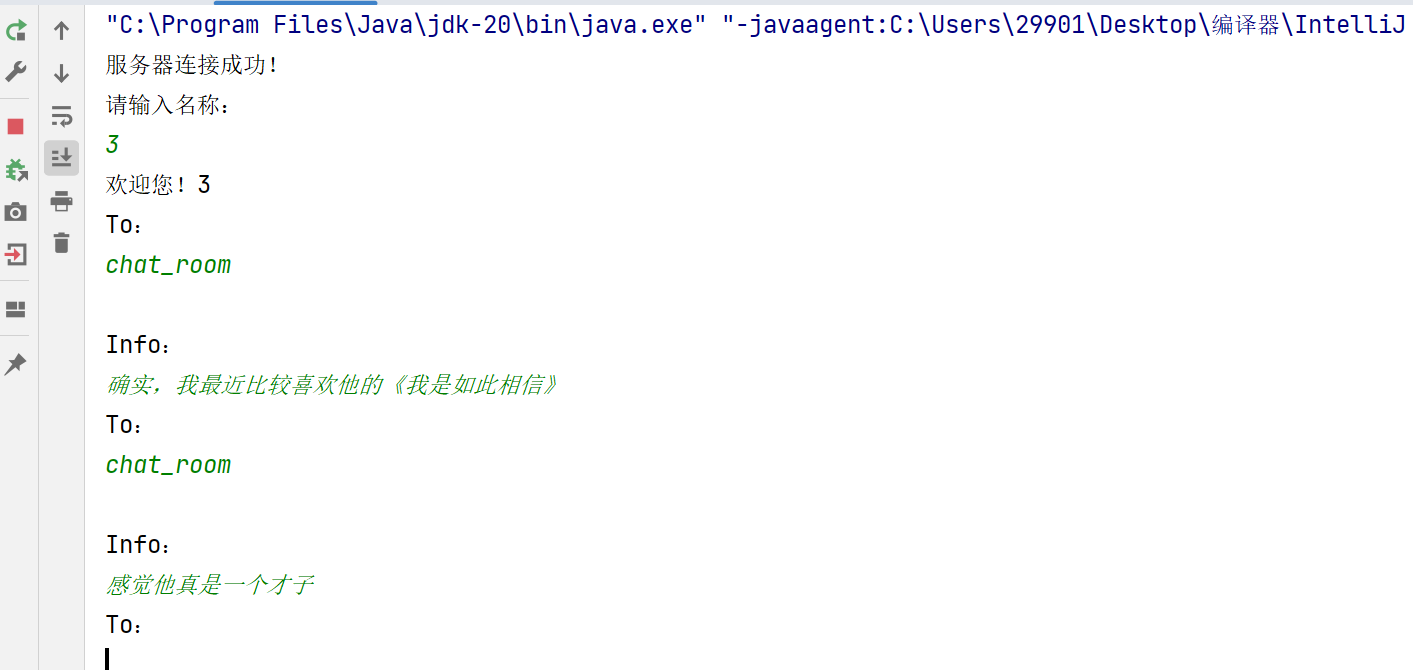


**②Client端**



运行结果：





1和2大致同3，不再赘述

### 出现的问题及解决方法：

**任务二：**

if(oIn != null){ *//没有数据就关闭*

oIn.close();

}

**最开始直接写oln = null时关闭，其实是没有对并发操作有正确的理解**

在通常情况下，输入流 oIn 在成功建立和初始化后会被赋予一个非 null 的值。然而，在多线程或者异步编程的情况下，可能存在这样的场景：

并发操作： 如果在一个线程中初始化输入流 oIn 的同时，另一个线程尝试关闭该流，那么在初始化完成之前，oIn 可能仍然为 null。

异常情况： 如果在初始化 oIn 的过程中发生了异常，导致输入流未成功创建，此时 oIn 也可能为 null。

通过检查 oIn != null，程序可以避免在这些情况下尝试关闭未成功初始化或实例化的输入流。这是一种常见的防御性编程做法，确保在关闭资源之前先验证其状态，防止潜在的异常情况。

**任务三：**

通过多线程的方式解决阻塞问题，一开始对于while(true)的位置也是不好找，尝试过将while（true）放在try里面，但是这会导致while（true）下面的句子无法到达的情况，对于异常和建立多线程的处理上还需要将强练习



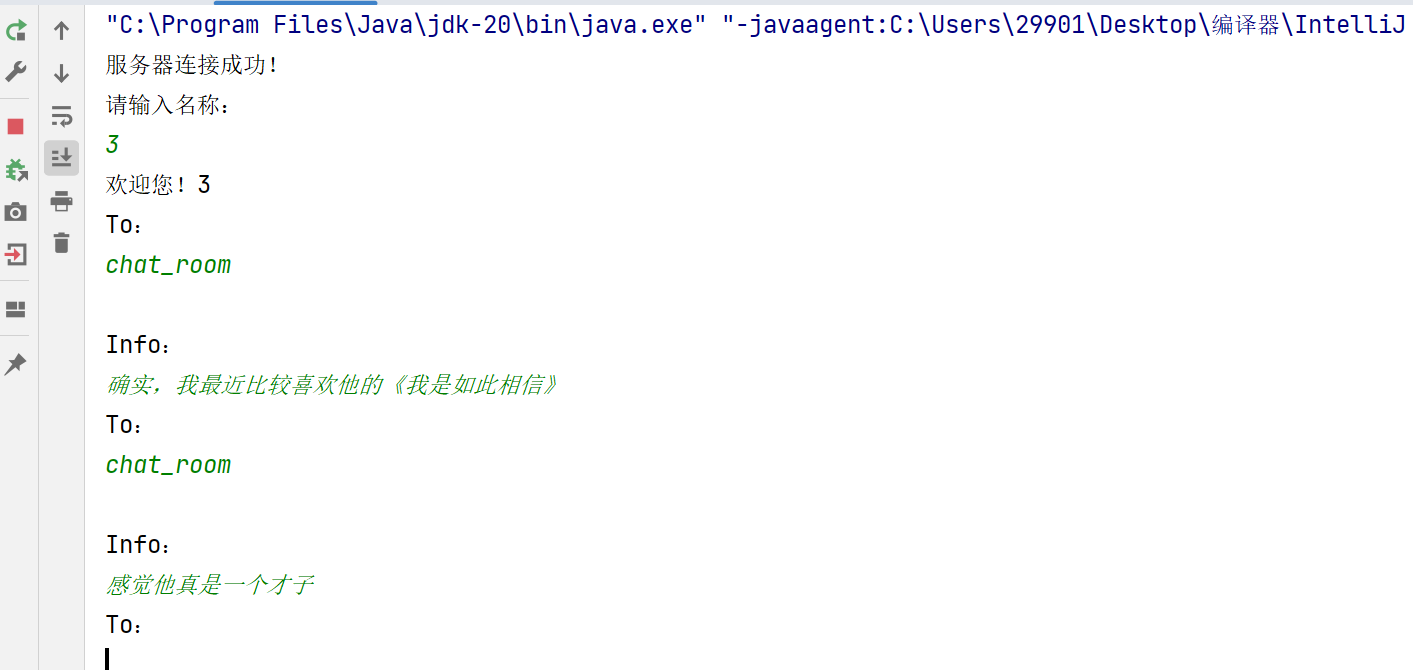
**任务四：**

①在插入图片时一开始的路径是图片在本机电脑上的绝对路径，这会导致将项目打包发送时图片丢失，一开始并不了解对于图片应放的位置及相对路径的规范要求，经查阅资料，将图片放在src目录的image文件夹中，将绝对路径改为src/image/图片名.png

②在消息为空时点击发送按钮该如何处理在一开始并没有想到，虽然结果都是差不多的，但是这行jta.setText(newMessage)增加了代码的灵活性和复用性，也方便在调试时找到问题所在



**任务六：**



由于自己的思路是将私聊与群聊结合在一起，所以每发送一个消息就需要判断这个消息是不是发送给群聊聊天室的，对于用户来说如果想要在群聊中发多条消息会比较麻烦，用户的体验感不是很好。

### 实验总结以及感悟：

这个实验加深了我对利用Socket，多线程，并使用swing实现简单的图形化界面等知识点的理解，实验的过程也是bug频出，一点点地微调，在图形化界面方面尤其是这样，总结我对一下知识点和问题的感悟，继续加深下一步的学习：

**①异常处理方面**：对于异常处理用的比较多的是 System.out.println( e);选取了最简单的异常处理方式，但是这可能不是很灵活，所以对于异常的处理还需要代码量的累计提高处理灵活度

**②在文件传输**这一块不是很熟练，整体思路大致如下：

通过socket.getInputStream() 获取输入流。读取一个字节作为文件名的长度。根据长度读取相应字节数组，并将其转换成字符串作为文件名。创建相应文件并准备一个 FileOutputStream 对象以追加的形式写入文件。通过循环读取输入流中的数据，并写入文件。输出成功接收的文件名。最后关闭。在判断是否为文件时用“//”作为标志，也可能会造成一些误会，有“//”但又不是文件，但TCPClient也会执行最后发现找不到这个地址，可能会使得资源造成一些浪费。

**③socket与swing**的学习与使用是这个实验任务的核心，在学习中比较顺畅，但在使用的时候就会卡顿，bug频出，界面的相关知识点比较碎，在调节位置和大小的过程也是耗时较长，但是这个没关系，因为java主要是做后端的，所以自己最需要提升和锻炼的是对项目整体思路和框架的思考。