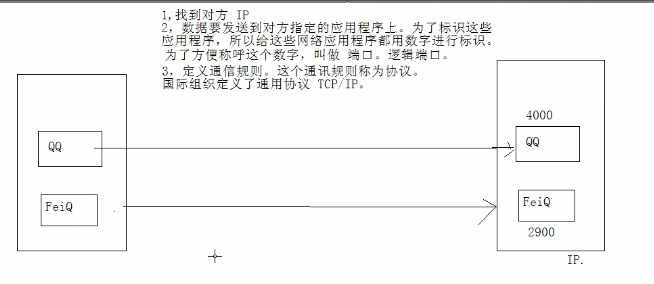
**P292 网络编程 (概述）**



**P293 网络编程 (概述2）**

本地端IP地址：127.0.0.1

**P294 网络编程 (网络模型）**



如果我们想把一些数据从A机器发送给B机器，这些数据是怎么传输的呢？

-数据封包

1.根据应用层特征，将数据封装，传输给表示层

2.根据表示层特征，将数据封装，传输给会话层

3.根据会话层特征，将数据封装，传输给传输层。此时数据基本封装完成

---------------------------------------

4.传输层有TCP/IP协议，根据传输层特征将数据封装，准备传输，然后传输给网络层

5.网络层给封装数据IP地址（告诉数据往哪里去），传输给链路层

----------------------------------------

6.链路层再封装给物理层

7.通过物理层设备（网线、蓝牙、光纤、witf）从A的物理层传给B的物理层

-数据拆包

从B的物理层一步步拆包到B的应用层。（问：比如A的会话层对数据进行了封装，为什么B的会话层能拆开呢？因为A和B都遵循OSI参考模型）

直到应用层才读到该数据要传给那个端口，就给那个端口对应的程序。

网络编程在网际层和传输层。JavaWeb开发在应用层。

应用层常见协议：http和ftp

传输层常见协议：TCP和UDP

网际层常见协议：IP

**P295 网络编程 (IP地址）**



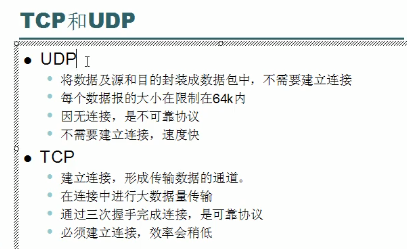


像baidu这类主机名，往往会有多个IP地址，所以18行也可以改为：

InetAddress[] ias = InetAddress.getAllByName(“www.baidu.com”);

以获取所有的IP地址

**P296 网络编程 (TCP和UDP）**



UDP实例：聊天软件；电脑课上的远程控制、桌面共享软件；斗鱼等直播软件

TCP必须建立连接，也就是对方必须在才能传输。如何确定对方在不在呢？通过三次握手完成。

UDP类似对讲机，TCP类似电话

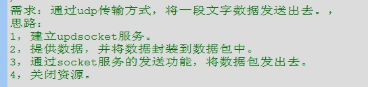
**P297 网络编程 (Socket）**

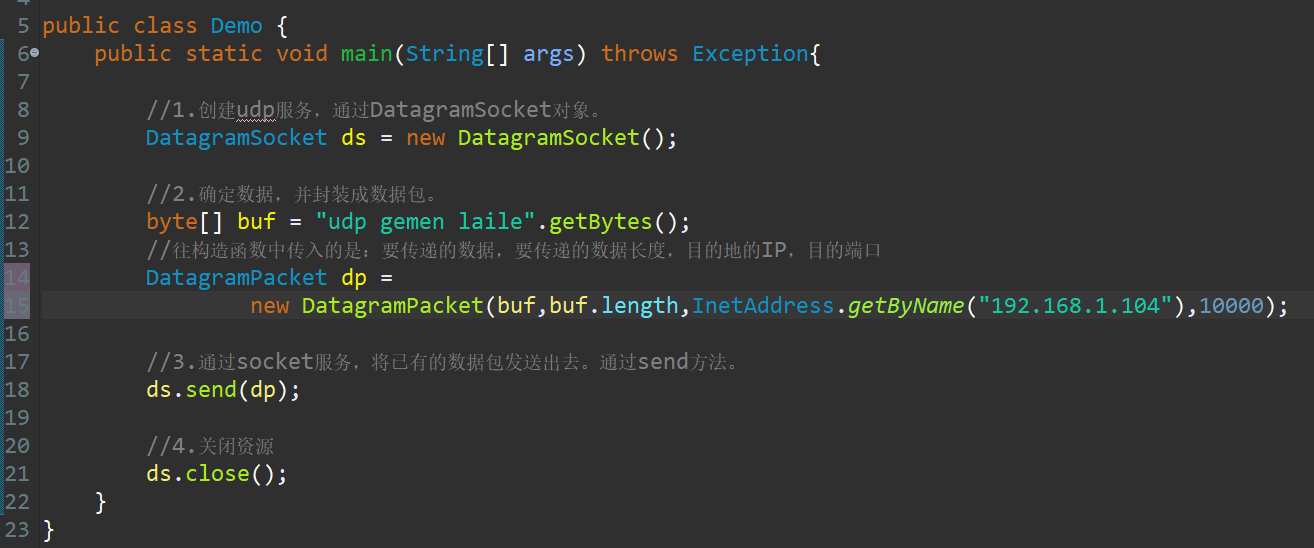


Socket相当于计算机中的应用程序。只有确定了传出的Socket和接收的Socket，数据才能进行传输。

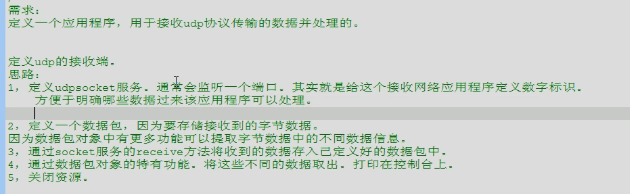
**P298 网络编程 (UDP-发送端）**

DatagramSocket类表示用来发送和接收数据报包的套接字

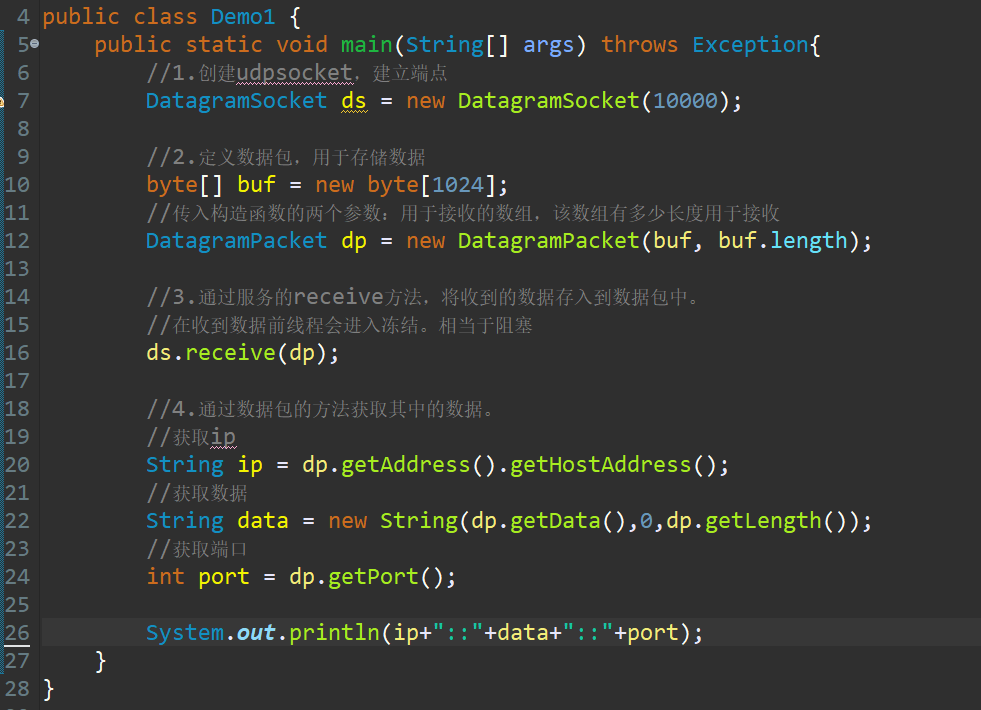




**P299 网络编程 (UDP-接收端）**

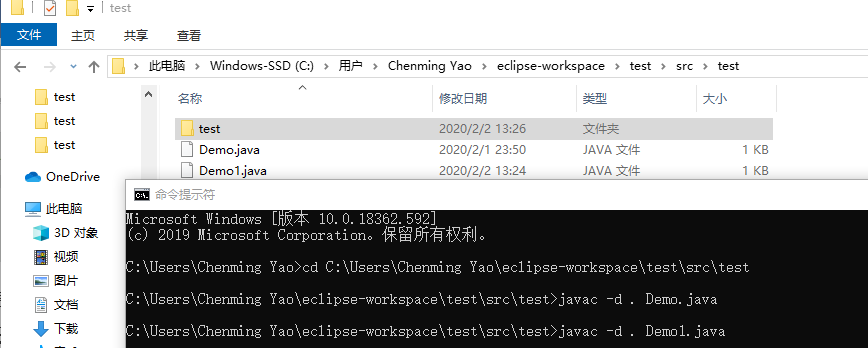


关于思路1：从udp发送端我们可以看到，你要发送一个数据，必须要有目的地的ip地址和端口，至于收不收得到udp发送端不管。但是现在的需求是，我们确实的想要接收到这批数据，怎么办呢？关键是ip地址和端口要能对上。Ip地址我们控制不了，如果发送端的ip地址本就就是接收端的ip地址，那摆明就不是发给你的。Ip统一后，我们唯一能做的就是统一端口了。

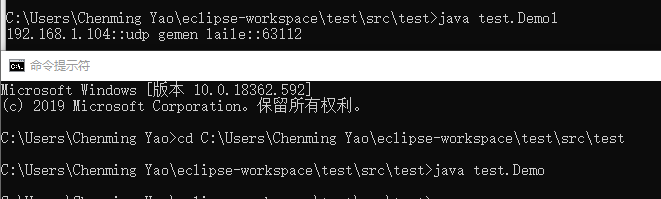


那么我们有了发送端和接收端，下面就来测试一下。

因为两个java都是在ide中写的，天然带包，所以我们带包编译一下



编译完成后，因为是udp，所以先运行哪个都是没事的。先发送后接收，会使得数据丢失。只有先接收后发送，才会接到数据。



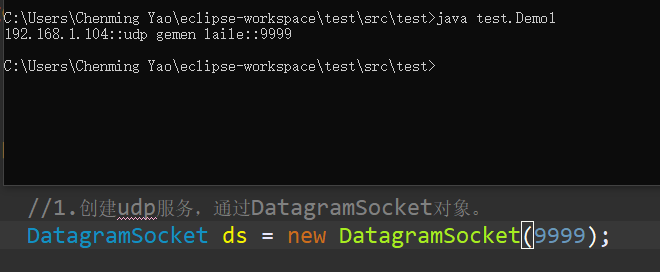
Cmd在运行接收端后，一直处于等待阶段，直到发送端运行，接收端才接收到数据并结束（并不是接收一个就结束，而是在我们自己写的代码中，接受了一个就close了）。

从接收的数据我们可以看出包括3个部分：

发送端的IP，发送的数据，发送端的端口

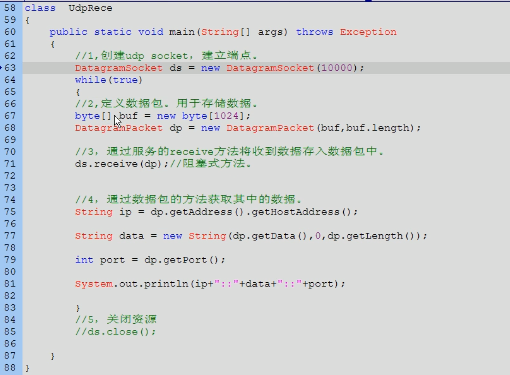


我们又发送了一次，发现发送端的端口变化了。因为当初没定义，所以是系统默认分配的。当然也可以像接收端一样自己定义一个，这样接收端收到的端口就是自己定义的发送端端口



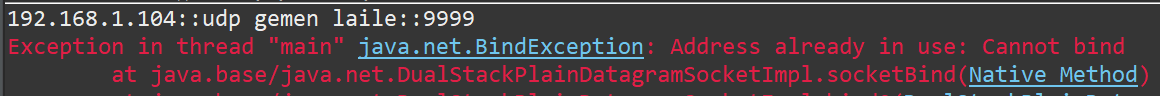
**P300 网络编程 (UDP-键盘录入方式数据）**

需求，上面编写的接收端由于没有设置循环，所以接收一次就关闭了资源，如果想要让接收端一直接收数据，那么可以设置一个循环



64-83行是循环体。那么可不可以把创建端点也放入循环体呢？

不可以。想象循环：第一次你new了一个端口，阻塞在第71行。接收到数据后仍在循环中，会再次new一个端口，而这个10000端口已经被占用了。结果就是接收第一次的数据然后报错。



那么，既然是端口被占用，那么我们更换每次new的端口呢？可以是可以，但是不要忘了发送端，就是发送给固定的端口，你端口在变的话，不就只能接收到一次数据了吗？后面的因为端口不对应就都丢失了。

根据键盘录入我们新做了发送端



然后做了变化不大的接收端

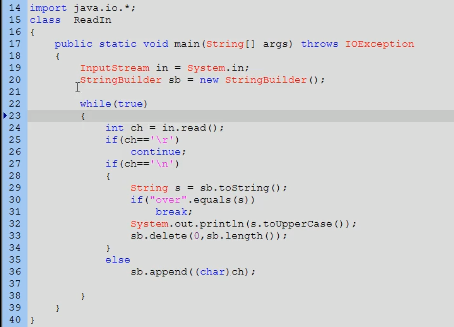


结果：每次键盘录入回车后的数据，都接收到了。就是你输入一行，它接收一行。

问题：输入一行就接收一行，那么在发送端中，这个循环的循环体在哪呢？

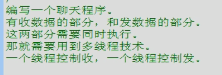
注意26行，readLine()方法也是阻塞机制，在收到数据之前会冻结。而我们的键盘无论输入什么都不会是null。

readLine的底层实现如下，之前有笔记，这里再提一下



**P301 网络编程 (UDP-聊天）**

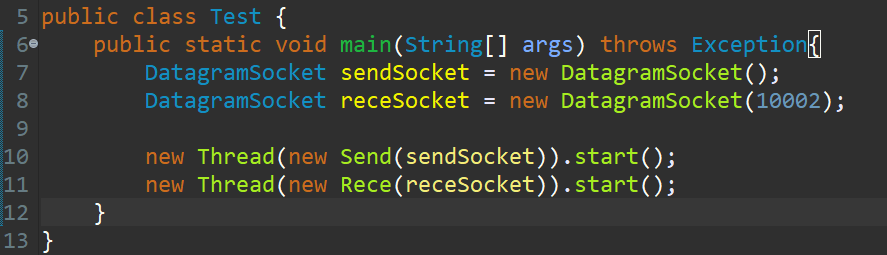
需求：



因为收和发动作是不一致的，所以要定义两个run方法。

而且这两个方法要封装到不同的类中。

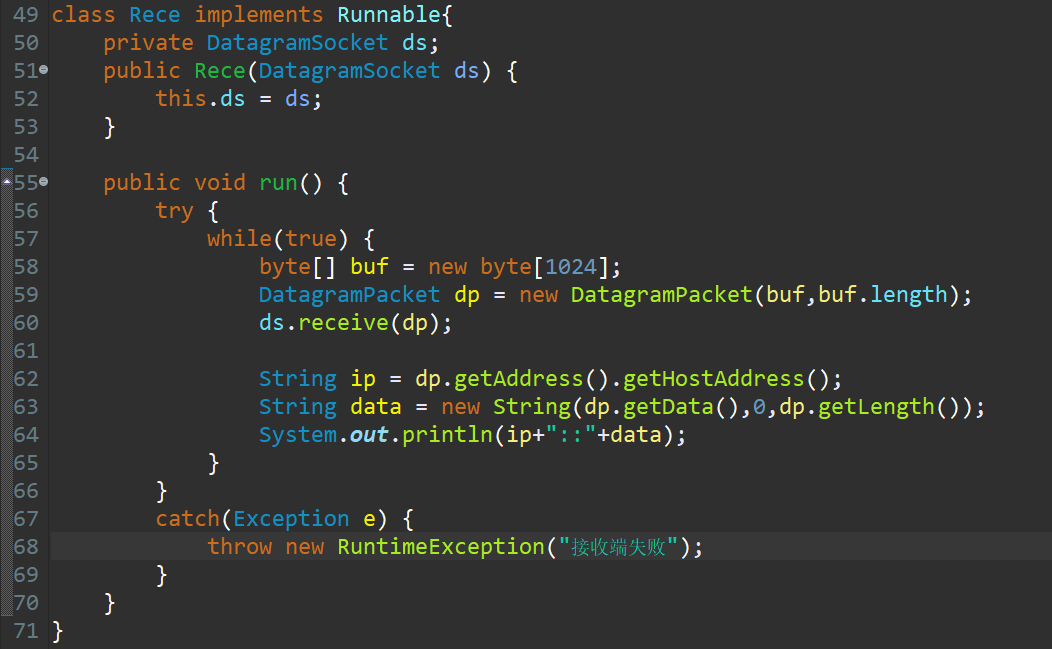
主函数



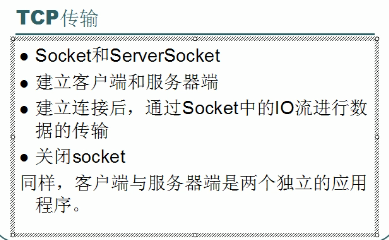
发送端



接收端

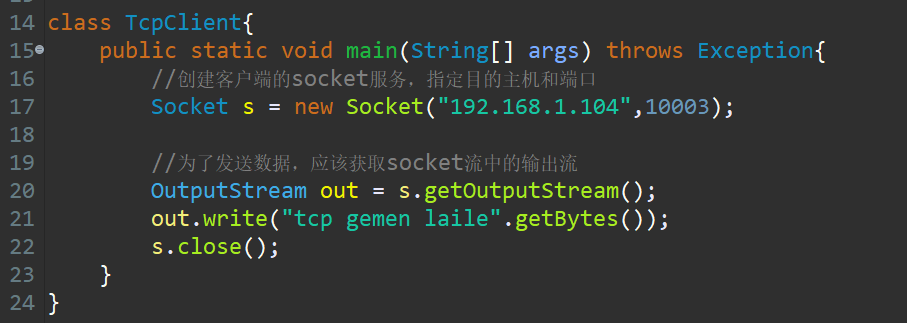


**P302 网络编程 (TCP传输）**

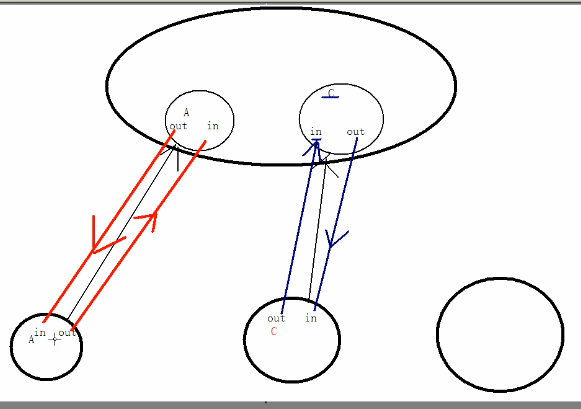


TCP分客户端和服务端，客户端对应的对象是Socket，服务端对应的对象是ServerSocket。

TCP客户端



服务端的基本原理：由客户端发起请求，如果请求成功，那么服务端会建立一个该客户端的对应对象，服务端的in和客户端的out连，服务端的out和客户端的in连



服务端：

1.建立服务端的socket服务，ServerSocket();

2.获取连接过来的客户端对象：通过ServerSocket的accept方法（这个方法是阻塞式的，没有连接就会等）。该方法返回一个socket对象

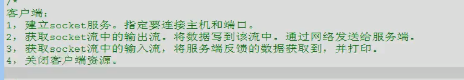
3.客户端如果发过来数据，那么服务端要使用对应的客户端对象，并获取到该客户端对象的读取流来读取发过来的数据

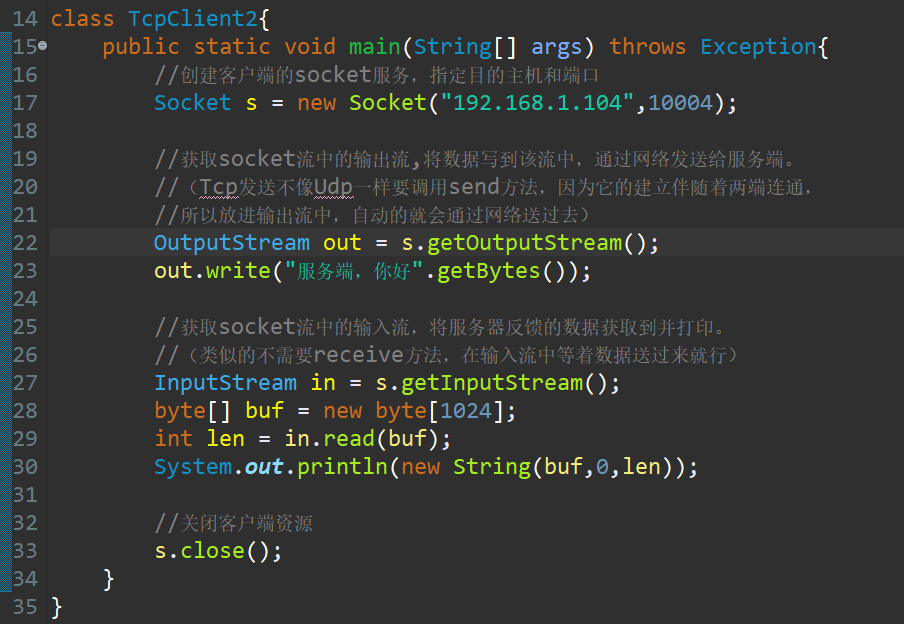


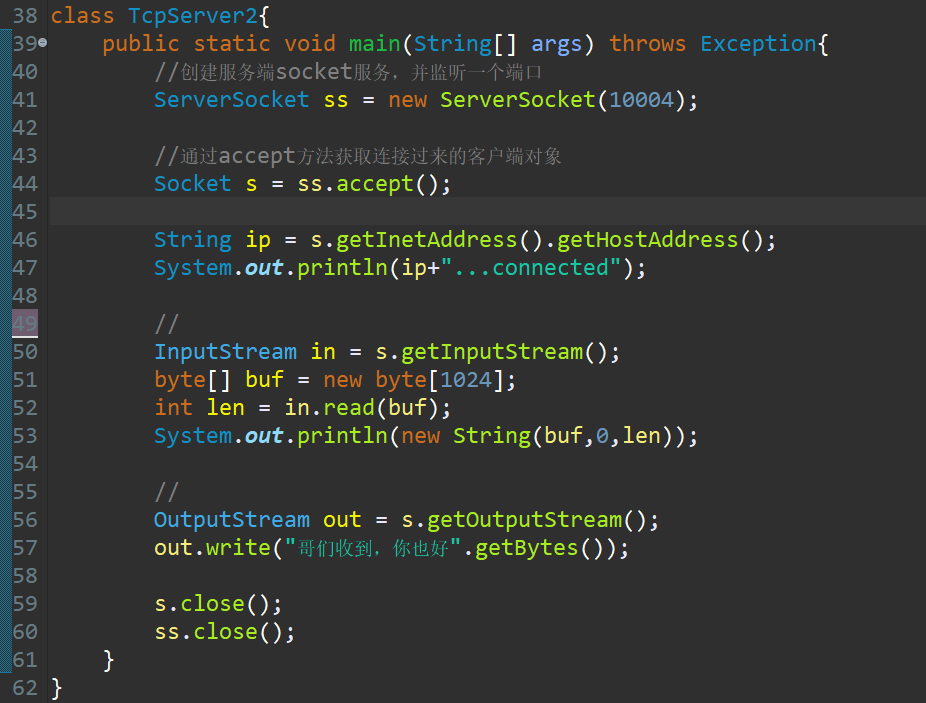
**P303 网络编程 (TCP传输2）**

演示tcp的传输的客户端和服务端的互访

需求：客户端给服务端发送数据，服务端收到后，给客户端反馈信息。

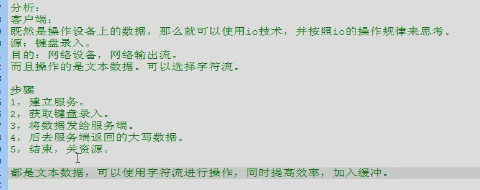


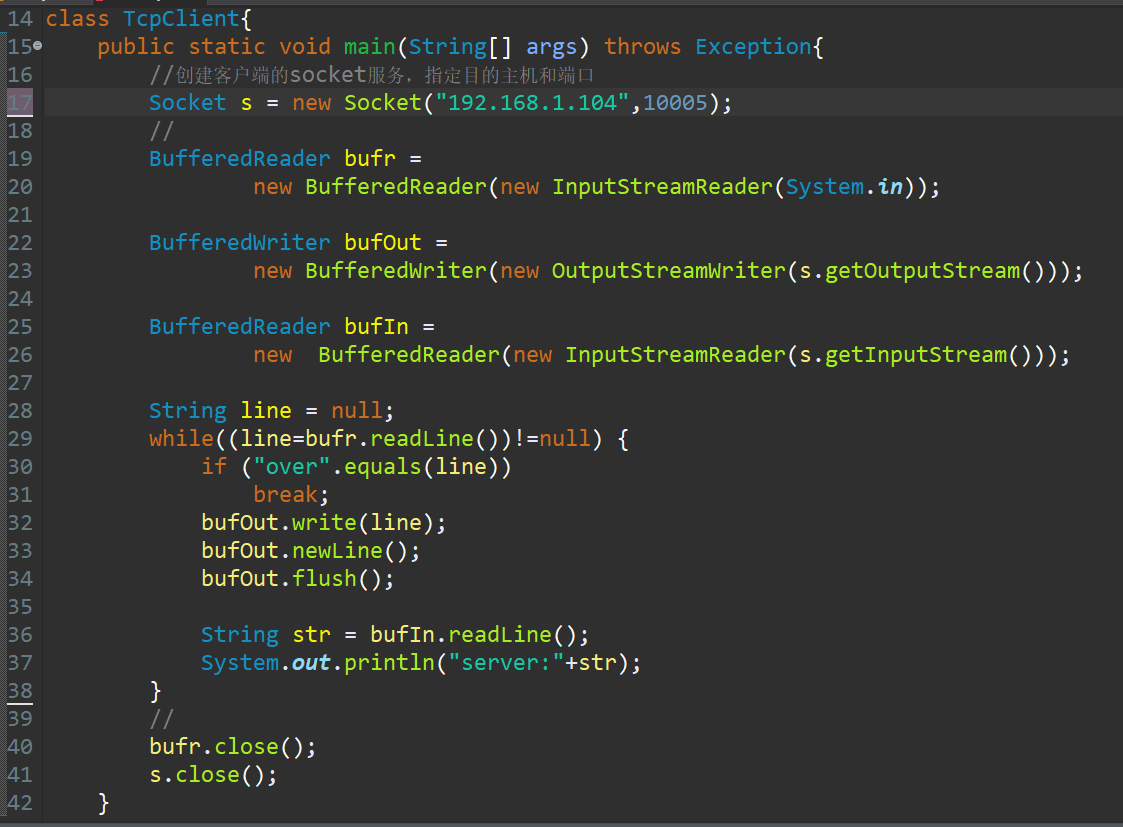


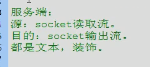


**P304 网络编程 (TCP练习）**

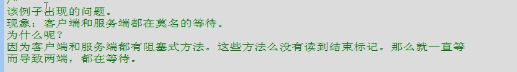








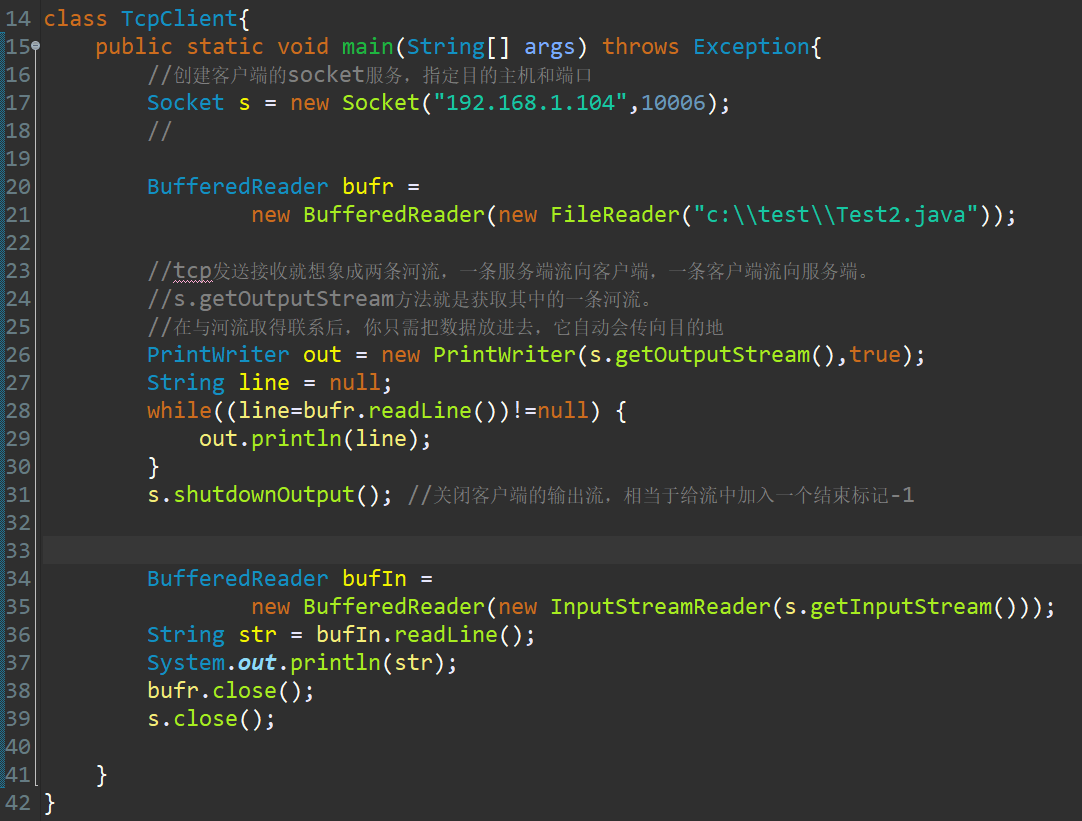




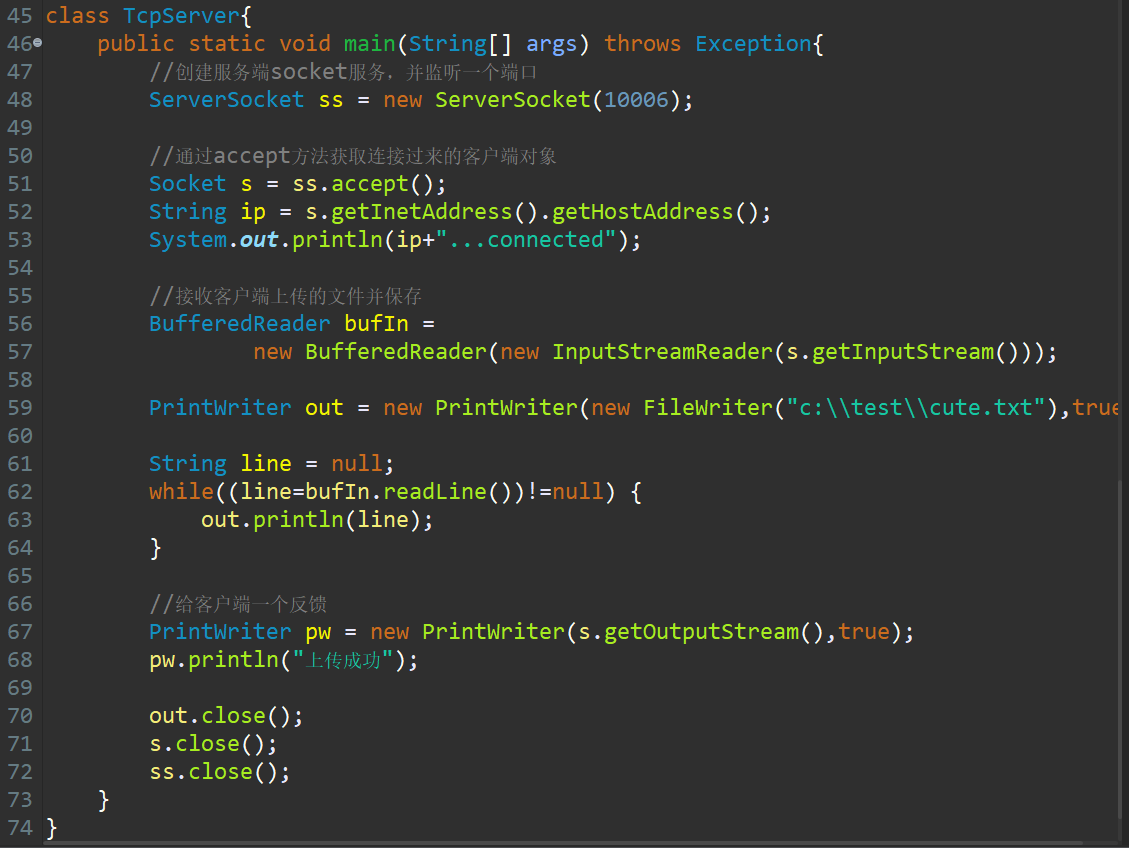
就是客户端的32行write写入流的只有文本而没有回车符。到了服务端的65行没有读到回车符，readLine方法仍在等待客户端的进一步输入而阻塞。同时客户端的36行没有读到服务端的返回而阻塞。

**P305 网络编程 (TCP复制文件）**

需求：从客户端上传一个文件到服务端，服务端再把这个文件储存起来



关键是弄清楚IO的源和目的

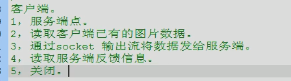


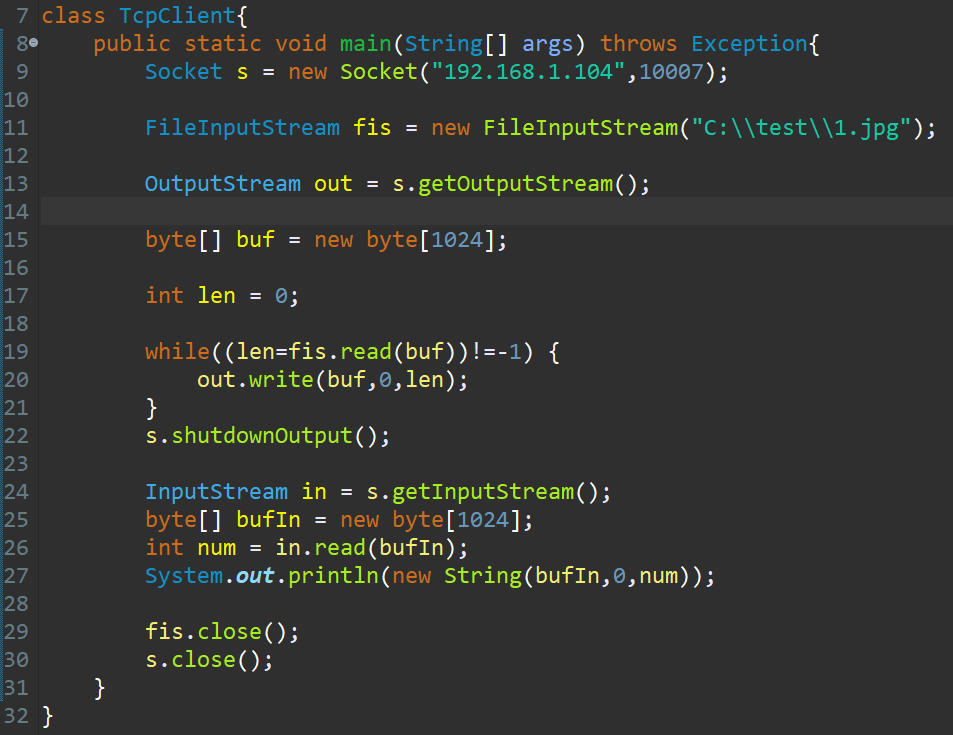
注意客户端的31行。如果没有这31行，客户端在上传文件之后，服务端不知道已经全部上传完毕，服务端的readLine会阻塞。服务端的阻塞就导致客户端36行的readLine阻塞。所以要有31行来告诉服务端：上传已全部完毕。

**P306 网络编程 (TCP-上传图片）**

需求：客户端上传图片给服务端

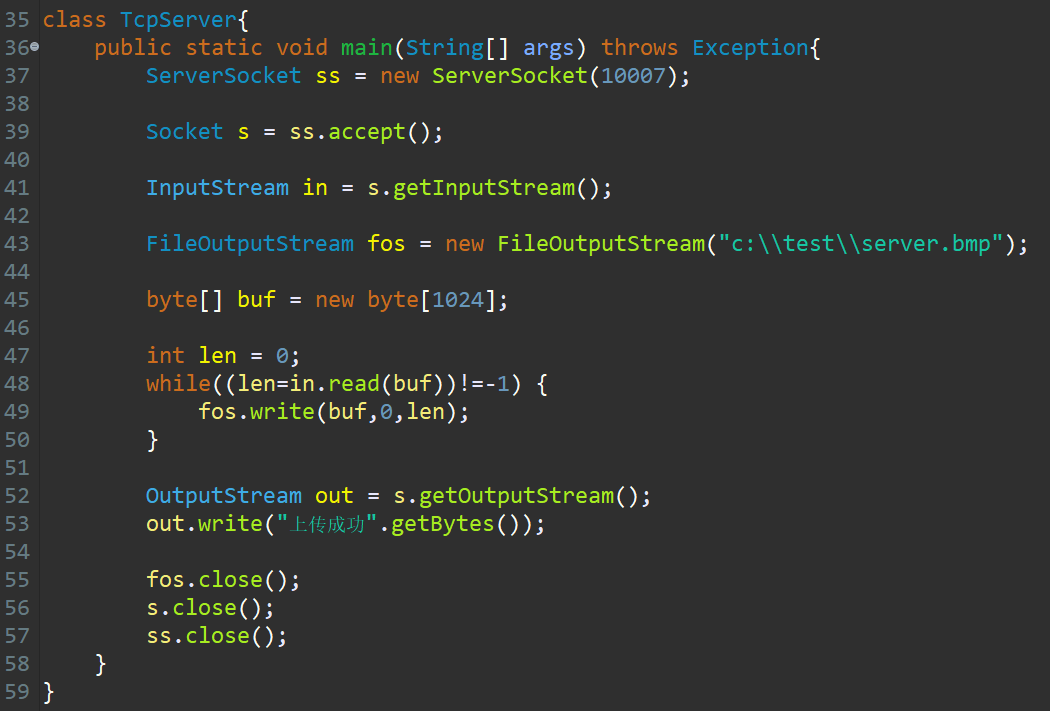
分析客户端。跟P305一样





搞清楚IO的源和目的就很简单：源是fis，目的是out

服务端



源是in，目的是fos

**P307 网络编程 (TCP-客户端并发上传图片）**

如果服务端想同时接收图片，该怎么办？

把服务端中的39-56行嵌套进while(true)循环可以吗？

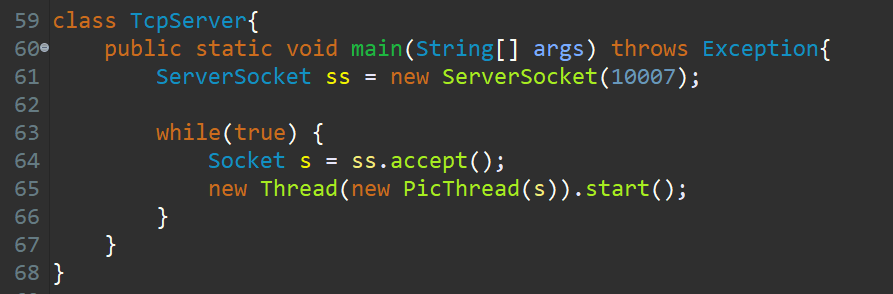
这么做只能A传完，才能传B。A如果在传，B是连接不上服务器的，所以不行。

那么为了可以让多个客户端同时并发访问服务端，服务端最好就是将每个客户端封装到一个单独的线程中，这样就可以同时处理多个客户端请求。

如何定义线程呢？

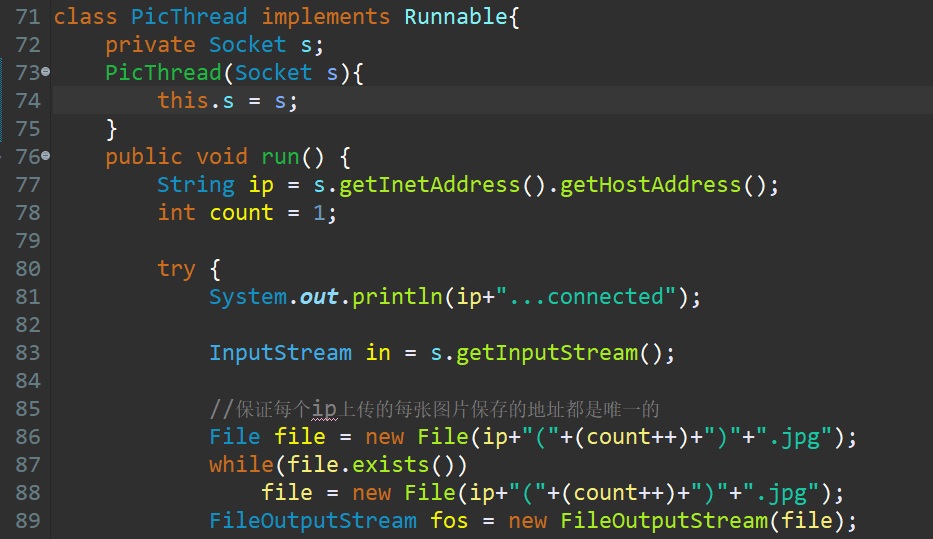
只要明确了每个客户端在服务端执行的代码即可。将该代码存入run方法中。

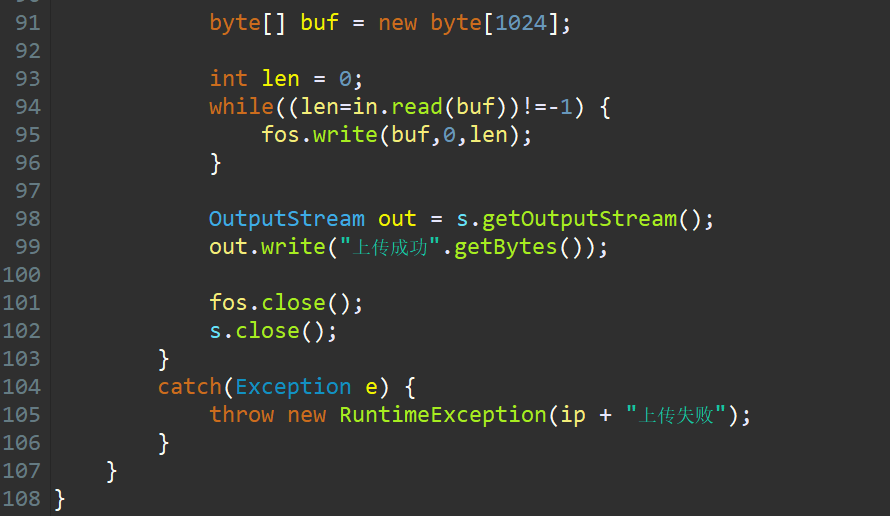
客户端



主线程一直就在等待客户端的连接。每当有客户端连接过来，就开启一个线程与它对接。

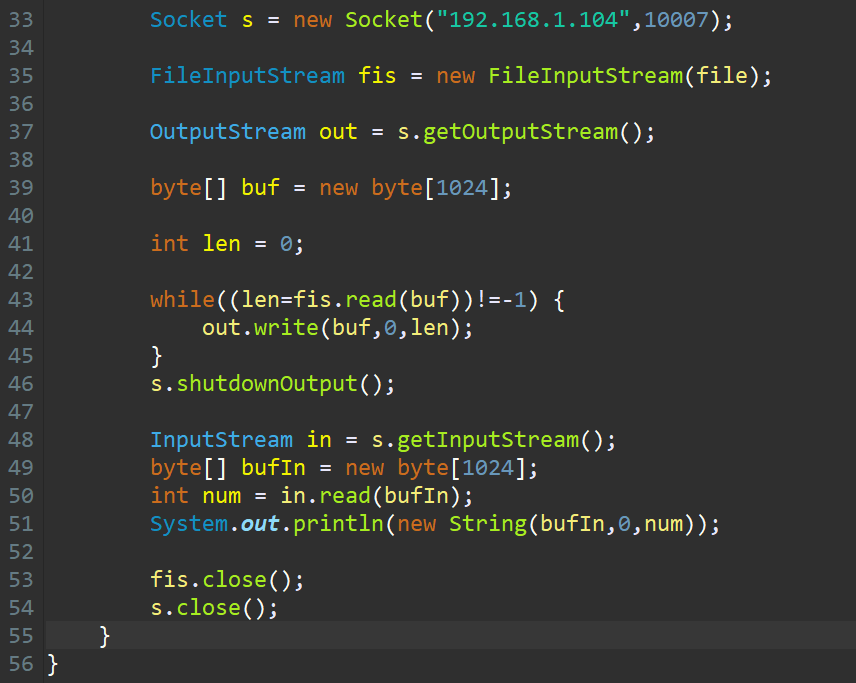
多线程





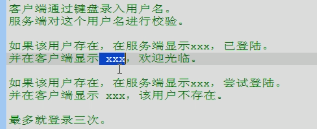
客户端。对上传的文件进行控制





**P308 网络编程 (TCP-客户端并发登录）**

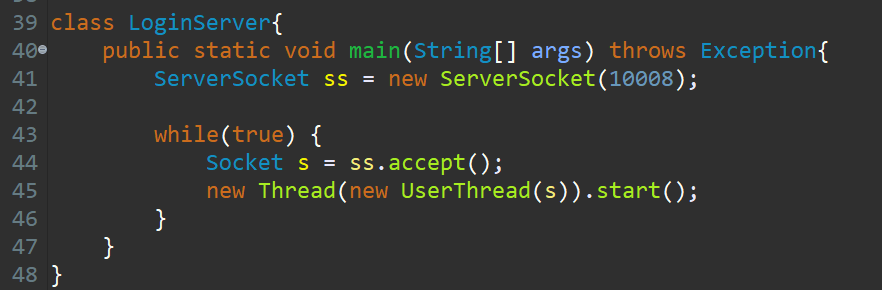
需求：

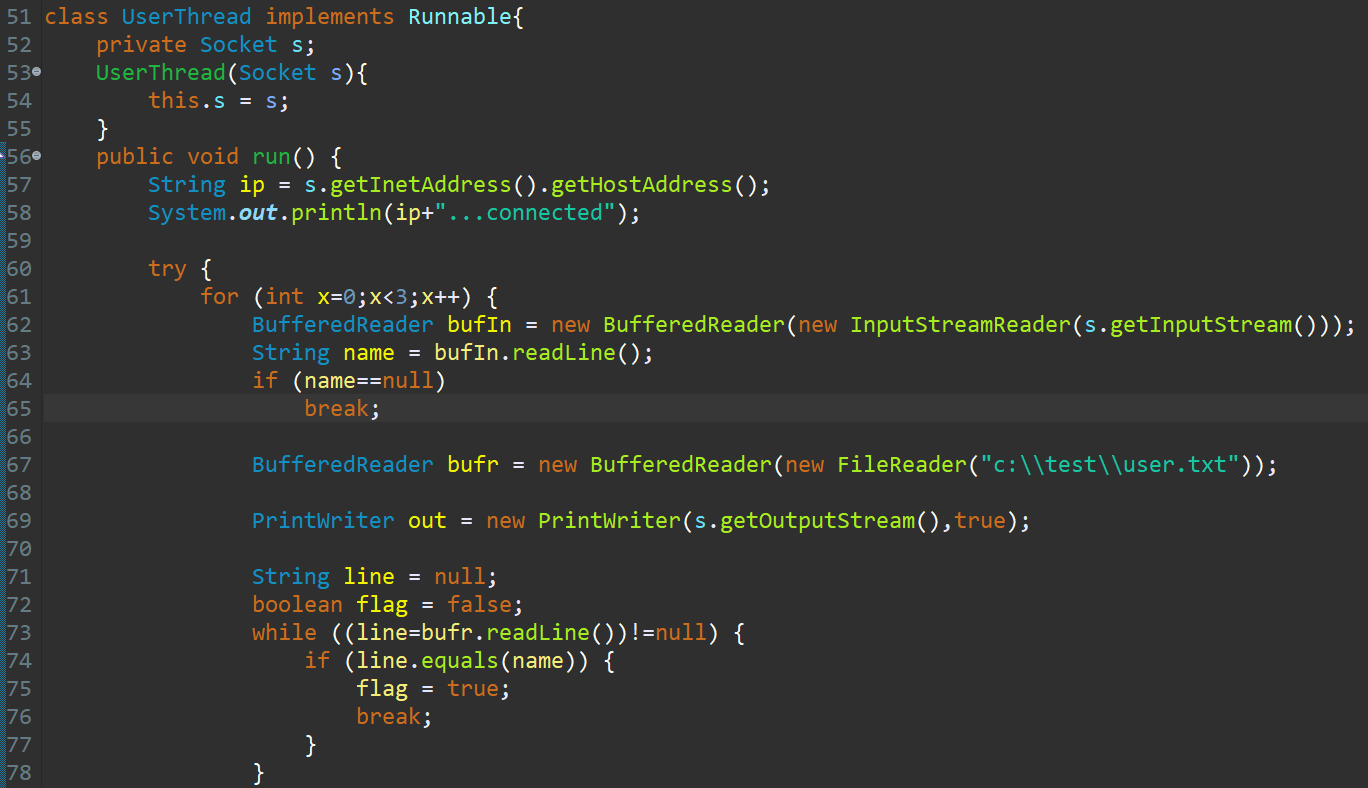


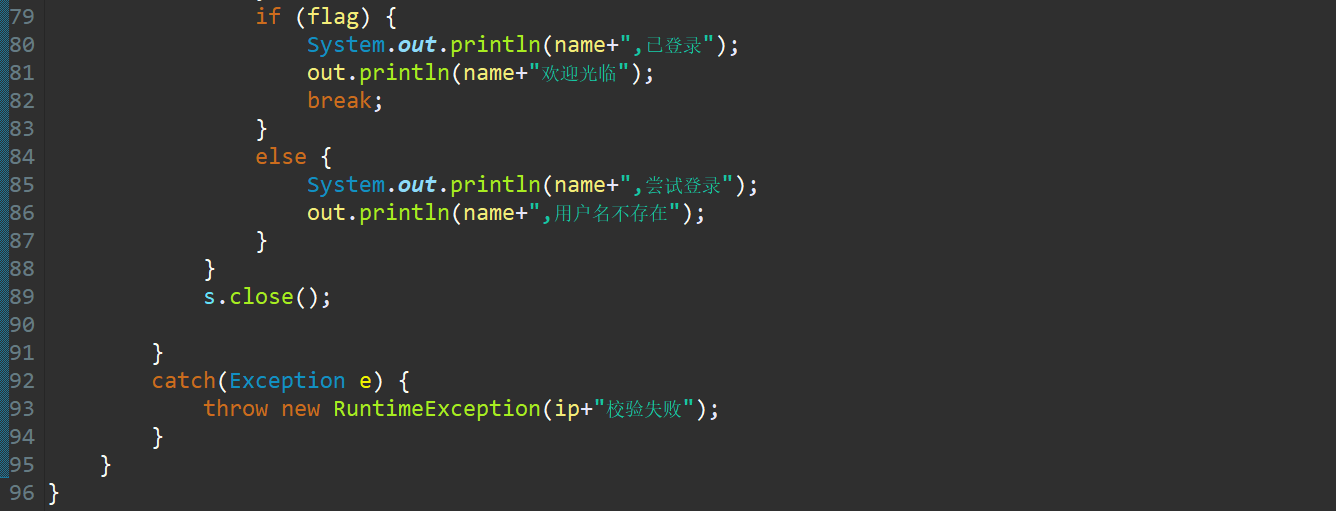


-bufr用于获取键盘录入；out用于向服务端发送数据；bufIn用于获取服务端反馈的数据

服务端使用多线程技术







注意64-65行。如果客户端不想尝试，ctrlc结束了，那么客户端从21行跳到了32行，并最终关闭了客户端。此时服务端的63行，我们发现如果客户端关闭，readLine方法将不会阻塞，因为每次的readLine都会读到-1（也就是null）。

**P309 网络编程 (浏览器客户端-自定义服务端）**

需求：我们做一个服务端，然后等待浏览器这一客户端访问过来。



**P310 网络编程 (浏览器客户端-Tomcat服务端）**

客户端：浏览器

服务端：Tomcat服务器

**P311 网络编程 (自定义浏览器-Tomcat服务端）**

//

**P312 网络编程 (自定义图像界面浏览器-Tomcat服务端）**

//

**P313 网络编程 (URL-URLConnection）**



**P314 网络编程 (网络编程（小知识点））**

//

**P315 网络编程 (网络编程（域名解析））**

问题：我们在浏览器中输入一个网址后，浏览器到底做了什么事情呢？

1.找本地是否有域名对应的ip地址

2.如果没有，去域名解析服务器（DNS）找主机名和ip地址的映射关系表。DNS返回域名对应的ip地址

3.这时才用拿到的ip地址去访问目标服务器

