基于Socket的Java网络编程集粹 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E5_9F_BA_E4_BA_8ESock_c104_144431.htm 其实,简单的分析一下，就可以看出客户和服务通讯的主要通道就是Socket本身，而服务器通过accept方法就是同意和客户建立通讯.这样当客户建立Socket的同时。服务器也会使用这一根连线来先后通讯，那么既然如此只要我们存在多条连线就可以了。那么我们的程序可以变为如下: 服务器: import java.io.*. import java.net.*. public class MyServer { public static void main(String[] args) throws IOException{ ServerSocket server=new ServerSocket(5678). while(true){ Socket client=server.accept(). BufferedReader in=new BufferedReader(new InputStreamReader(client.getInputStream())). PrintWriter out=new PrintWriter(client.getOutputStream()). while(true){ String str=in.readLine(). System.out.println(str). out.println("has receive...."). out.flush(). if(str.equals("end")) break. } client.close(). } } } 这里仅仅只是加了一个外层的While循环，这个循环的目的就是当一个客户进来就为它分配一个Socket直到这个客户完成一次和服务器的交互，这里也就是接受到客户的"End"消息.那么现在就实现了多客户之间的交互了。但是.问题又来了，这样做虽然解决了多客户,可是是排队执行的。也就是说当一个客户和服务器完成一次通讯之后下一个客户才可以进来和服务器交互，无法做到同时服务，那么要如何才能同时达到既能相互之间交流又能同时交流呢？很显然这是一个并行执行的问题了。所以线程是最好的解决方案。那么下面的问题是如何使用线程.首先要做的事情是创建线程

并使得其可以和网络连线取得联系。然后由线程来执行刚才的操作，要创建线程要么直接继承Thread要么实现Runnable接口，要建立和Socket的联系只要传递引用就可以了.而要执行线程就必须重写run方法，而run方法所做的事情就是刚才单线程版本main所做的事情，因此我们的程序变成了这样: import java.net.*. import java.io.*. public class MultiUser extends Thread{ private Socket client. public MultiUser(Socket c){ this.client=c. } public void run(){ try{ BufferedReader in=new BufferedReader(new InputStreamReader(client.getInputStream())). PrintWriter out=new PrintWriter(client.getOutputStream()). //Mutil User but can parallel while(true){ String str=in.readLine(). System.out.println(str). out.println("has receive...."). out.flush(). if(str.equals("end")) break. } client.close(). }catch(IOException ex){ }finally{ } } public static void main(String[] args)throws IOException{ ServerSocket server=new ServerSocket(5678). while(true){ //transfer location change Single User or Multi User MultiUser mu=new MultiUser(server.accept()). mu.start(). } } } 我的类直接从Thread类继承了下来.并且通过构造函数传递引用和客户Socket建立了联系，这样每个线程就有了。一个通讯管道.同样我们可以填写run方法，把之前的操作交给线程来完成，这样多客户并行的Socket就建立起来了。 以上的代码使用的是BufferedReader in=new BufferedReader(new InputStreamReader(client.getInputStream())).PrintWriter out=new PrintWriter(client.getOutputStream()). 还有一种方法是使用DataInputStream isFromClient = new DataInputStream(client.getInputStream()).DataOutputStream

osToClient = new DataOutputStream(client.getOutputStream()).
关于这两种输入输出流的不同，我也只知道前一种对字符串支持比较好，后面对于读取一个字符串需要处理，但是可以支持很多种类型的输出。对于传递字符串而言前一种应该是很好的选择了。