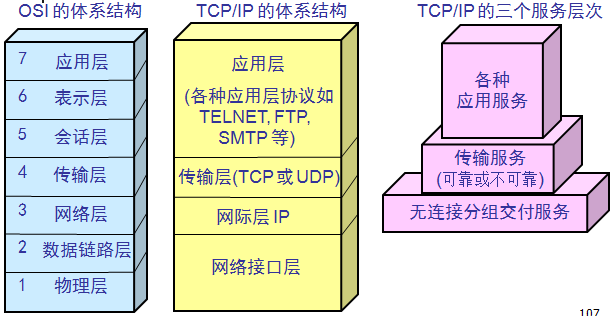
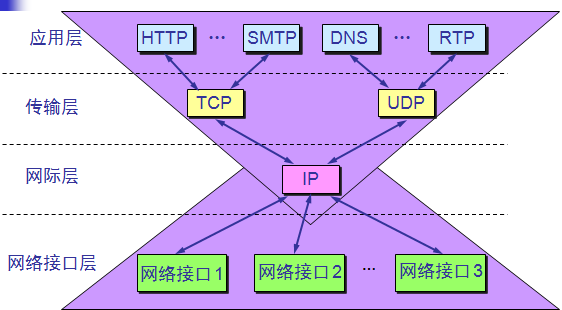
# Java网络编程





## URL和URI

<https://www.zhihu.com/question/21950864> 可参看这篇

### URL

URL是Uniform Resoure Locator（统一资源定位器）的缩写。URL地址格式排列为：scheme://host:Port/path

URL是Uniform Resource Locator的缩写，即统一资源定位器，它是一个识别Internet中哪里有信息资源，并且将 Internet提供的服务统一编址的系统。通过URL可以到达任何一个地方寻找需要的东西，比如文件、数据库、图像、新闻组等等，可以这样说，URL是 Internet上的地址簿。URL一般由三个部分构成，各个部分如下：

1. 服务器标识符 通过选择服务器标识符能够确定将要访问的服务器的类型， URL中的服务器标识符可以有HTTP://、FTP://、GOPHER://、TELNET://、NWES://等等类型，分别指定为采用超文本传输协议连接、采用文件传输协议连接、与GOPHER服务器连接、与TELNET会话连接、与USENET新闻组相连接。

2. 信息资源地址 信息资源地址是由两部分构成的，一是机器名称，如www.tsinghua.edu.cn是用来指示资源所存在的机器，另一个是通信端口号（port number），是连接时所使用的通信端口号。端口是Internet用来辨别特定信息服务用的一种软件标识，其设置范围是0到65535之间的整数，一般情况下使用的是标准端口号，可以不用写出。在需要特殊服务时会用到非标准端口号，这时就要写出，如http://www.tsinghua.edu.cn:81。常见的Internet提供服务的端口号，如HTTP的标准端口号为80，TELNET的标准端口号为23，FTP的标准端口号为21等等。

3. 路径名 路径名是给出资源在所在机器上的完整文件名，一般情况下只有用户知道所要找的资源在什么地方时才会给出这个选项。如http://www.tsinghua.edu.cn/index.html等。

### URI

网络上可用的每种资源 - HTML文档、图像、视频片段、程序等 - 由一个通过通用资源标志符(Universal Resource Identifier, 简称"URI")进行定位。

　　URI一般由三部分组成:

访问资源的命名机制。

存放资源的主机名。

资源自身的名称，由路径表示。

　　考虑下面的URI，它表示了当前的HTML 4.0规范:

　　http://soft.webjx.com/lesson/148/2623648.shtml

　　这个URI是这样的:这是一个可通过HTTP协议访问的资源，位于主

　　机soft.webjx.com上，通过路径“/lesson/148/”访问。在HTML文档中其它资源包括"mailto"(收发email)和"ftp"(FTP访问)。

　　这是URI的另一个例子，指向一个用户的邮箱:

　　<A href=mailto:web@webjx.com>写给站长</A>

　　注:大多数读者可能熟悉"URL"，而不是URI。URL是RUI命名机制的一个子集。

　　片段标志符

　　有的URI指向一个资源的内部。 这种URI以"#"结束，并跟着一个anchor标志

　　符(称为片段标志符)。例如，下面是一个指向section\_2的URI:

　　http://somesite.com/html/top.htm#section\_2

　　相对URI

　　相对URI 不包含任何命名规范信息。它的路径通常指同一台机器上的资源。相

　　对URI可能含有相对路径(如，“..”表示上一层路径)，还可能包含片段标志符。

　　为了说明相对URI，假设我们有一个基本的URI http://www.acme.com/support/intro.htm

　　下面的链接中使用了相对URI:

　　<A href="104/2627604.shtml">网页陶吧</A>

　　它扩展成完全的URI就是 "http://homepage.webjx.com/104/2627604.shtml"。

　　下面是一个图像的相对URI:

　　<IMG src="http://www.webjx.com/htmldata/TLimages/img/head/logo.gif" alt="logo">

　　它扩展成完全的URI就是 "http://www.webjx.com/TLimages/img/head/logo.gif"。

　　在HTML中，URI被用来:

链接到另一个文档或资源(参看A和LINK元素)。

链接到一个外部样式表或脚本(参看LINK和SCRIPT元素)。

在页内包含图像、对象或applet(参看IMAG、OBJECT、APPLET和INPUT元素)。

建立图像映射(参看MAP和AREA元素)。

提交一个表单(参看FORM)。

建立一个框架文档(参看FRAME和IFRAME元素)。

引用一个外部参考(参看Q、BLOCKQUOTE, INS和DEL元素)。

指向一个描述文档的metadata(参看HEAD元素)。

|  |
| --- |
| package com.test.net;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.net.MalformedURLException;  import java.net.URL;  public class URLTest {  //Uniform Resource Locator  //统一资源定位符  //https://ss1.bdstatic.com/5eN1bjq8AAUYm2zgoY3K/r/www/cache/static/protocol/https/baiduia/baiduia\_b45d552b.js    //1.协议部分：http、https、ftp等  //2.ip地址  //3.资源在服务器上面的位置和名称(虚拟地址，由web服务器软件来映射生成)  public static void main(String [] args) throws IOException{  try {  URL url = new URL("http://www.baidu.com");  InputStream is = url.openStream();  InputStreamReader isr = new InputStreamReader(is);  BufferedReader br = new BufferedReader(isr);  String s = br.readLine();  while(s!=null){  System.out.println(s);  s = br.readLine();  }  br.close();  } catch (MalformedURLException e) {  e.printStackTrace();  }  }  } |

### URLDecoder和URLEcoder

URLEncode和URLDecode用于完成普通字符串和 application/x-www-from-urlencoded MIME字符串之间的相互转化

如果传递的字符串中包含非西欧字符的字符串，会被转化成%XX%XX   XX为十六进制的数字

**[java]**

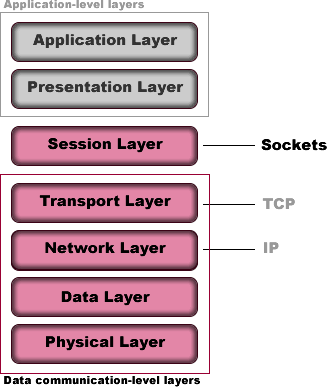
1. **try** {
2. // 将application/x-www-from-urlencoded字符串转换成普通字符串
3. String keyWord = URLDecoder.decode("%C4%E3%BA%C3", "GBK");
4. System.out.println(keyWord);  //输出你好
5. // 将普通字符创转换成application/x-www-from-urlencoded字符串
6. String urlString = URLEncoder.encode("你好", "GBK");  //输出%C4%E3%BA%C3
8. System.out.println(urlString);
9. } **catch** (UnsupportedEncodingException e) {
10. // TODO Auto-generated catch block
11. e.printStackTrace();
12. }

### URLConnection

|  |
| --- |
| **package** com.test.net;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.InputStreamReader;  **import** java.net.HttpURLConnection;  **import** java.net.URL;  **import** java.net.URLConnection;  **public** **class** URLConnectionTest {  **public** **static** **void** main(String[] args) **throws** Exception {  URL localURL = **new** URL("https://www.baidu.com/s?ie=utf-8&f=8&rsv\_bp=0&rsv\_idx=1&tn=baidu&wd=%E9%9F%A9%E8%B1%AB%E5%8D%9A&rsv\_pq=b22901b80005c692&rsv\_t=300bhWZmSGqR20csdeod43RTWcYhS%2Fmh7lns2QeUk6E6aNnOwvUqU5c%2BFj4&rqlang=cn&rsv\_enter=1&rsv\_sug3=3&rsv\_sug1=1&rsv\_sug7=000&rsv\_sug2=0&inputT=1734&rsv\_sug4=2230&rsv\_sug=2");  URLConnection connection = localURL.openConnection();  HttpURLConnection httpURLConnection = (HttpURLConnection) connection;  httpURLConnection.setRequestProperty("Accept-Charset", "utf-8");  httpURLConnection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");  InputStream inputStream = **null**;  InputStreamReader inputStreamReader = **null**;  BufferedReader reader = **null**;  StringBuffer resultBuffer = **new** StringBuffer();  String tempLine = **null**;  **if** (httpURLConnection.getResponseCode() >= 300) {  **throw** **new** Exception("HTTP Request is not success, Response code is " + httpURLConnection.getResponseCode());  }  **try** {  inputStream = httpURLConnection.getInputStream();  inputStreamReader = **new** InputStreamReader(inputStream);  reader = **new** BufferedReader(inputStreamReader);  **while** ((tempLine = reader.readLine()) != **null**) {  resultBuffer.append(tempLine);  }  } **finally** {  **if** (reader != **null**) {  reader.close();  }  **if** (inputStreamReader != **null**) {  inputStreamReader.close();  }  **if** (inputStream != **null**) {  inputStream.close();  }  }      System.***out***.println(resultBuffer);  }  } |

## 套接字

套接字大致驻留在 OSI 模型的会话层（见图）。会话层夹在其上面向应用的层和其下的实时数据通信层之间。会话层为两台计算机之间的数据流提供管理和控制服务。作为该层的一部分，套接字提供一个隐藏从导线上获取比特和字节的复杂性的抽象。换句话说，套接字允许我们让应用程序表明它想发送一些字节即可传输数据。套接字隐藏了完成该项工作的具体细节。



套接字是一种软件抽象，用于表达两台机器之间的连接“终端”。对于一个给定的连接，每台机器上都有一个套接字，您也可以想象它们之间有一条虚拟的“电缆”，“电缆”的每一端都插入到套接字中。当然，机器之间的物理硬件和电缆连接都是完全未知的。抽象的全部目的是使我们无须知道不必知道的细节。

简言之，一台机器上的套接字与另一台机器上的套接字交谈就创建一条通信通道。程序员可以用该通道来在两台机器之间发送数据。当您发送数据时，TCP/IP 协议栈的每一层都会添加适当的报头信息来包装数据。这些报头帮助协议栈把您的数据送到目的地。好消息是 Java 语言通过"流"为您的代码提供数据，从而隐藏了所有这些细节，这也是为什么它们有时候被叫做流套接字（streaming socket）的原因。

把套接字想成两端电话上的听筒 ― 我和您通过专用通道在我们的电话听筒上讲话和聆听。直到我们决定挂断电话，对话才会结束（除非我们在使用蜂窝电话）。而且我们各自的电话线路都占线，直到我们挂断电话。

一般而言，Java 语言中的套接字有以下两种形式：

TCP 套接字（由 Socket 类实现，稍后我们将讨论这个类）

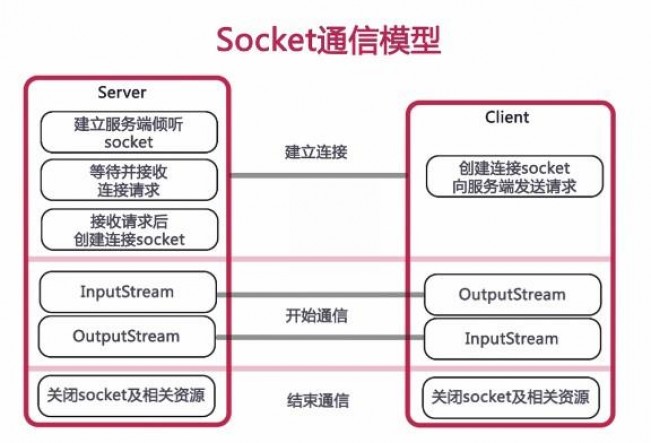
UDP 套接字（由 DatagramSocket 类实现）

TCP 和 UDP 扮演相同角色，但做法不同。两者都接收传输协议数据包并将其内容向前传送到表示层。TCP 把消息分解成数据包（数据报，datagrams）并在接收端以正确的顺序把它们重新装配起来。TCP 还处理对遗失数据包的重传请求。有了 TCP，位于上层的层要担心的事情就少多了。UDP 不提供装配和重传请求这些功能。它只是向前传送信息包。位于上层的层必须确保消息是完整的并且是以正确的顺序装配的。

一般而言，UDP 强加给您的应用程序的性能开销更小，但只在应用程序不会突然交换大量数据并且不必装配大量数据报以完成一条消息的时候。否则，TCP 才是最简单或许也是最高效的选择。

### TCP套接字

TCP是面向连接的可靠的传输模型，整个传输过程中维持着TCP的连接，内容的传输通过流来完成。



Socket通信的步骤

① 创建 Socket

② 打开连接到Socket的输入/输出流

③ 按照协议对Socket进行读/写操作

④ 关闭输入输出流、关闭Socket

服务器端：

① 创建ServerSocket对象，绑定监听端口

② 通过accept()方法监听客户端请求

③ 连接建立后，通过输入流读取客户端发送的请求信息

④ 通过输出流向客户端发送乡音信息

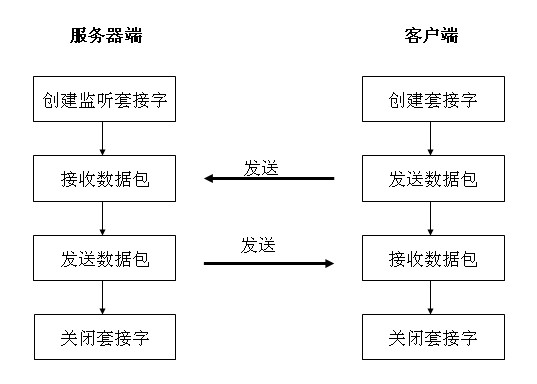
⑤ 关闭相关资源

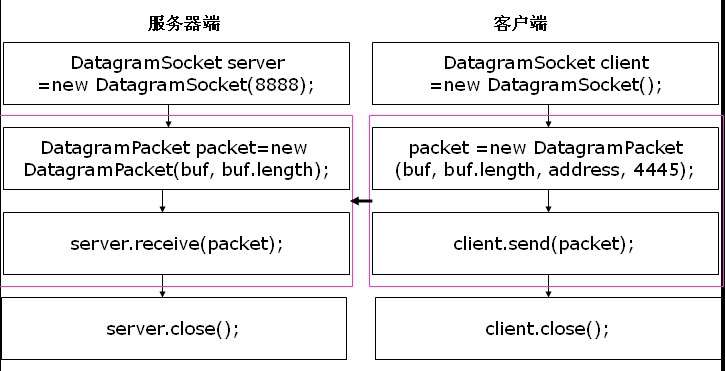
|  |
| --- |
| package com.test.net;  import java.io.DataOutputStream;  import java.io.IOException;  import java.io.OutputStream;  import java.net.InetAddress;  import java.net.ServerSocket;  import java.net.Socket;  public class SocketServerTest {  public static void main(String [] args){  try {  ServerSocket server = new ServerSocket(8888);  while(true){  //获取来访的Socket  Socket socket = server.accept();  OutputStream os = socket.getOutputStream();  DataOutputStream dos = new DataOutputStream(os);  InetAddress ia = InetAddress.getLocalHost();    dos.writeUTF(ia.getHostName()+":你好，客户端，你的地址信息是：" + socket.getInetAddress() + "你的端口是："+  socket.getPort());  dos.writeUTF("bye");  dos.close();  socket.close();  }  } catch (IOException e) {  e.printStackTrace();  }    }  } |

|  |
| --- |
| package com.test.net;  import java.io.DataInputStream;  import java.io.InputStream;  import java.net.Socket;  public class SocketClientTest {  public static void main(String[] args) throws Exception{  Socket socket = new Socket("127.0.0.1",8888);    InputStream is = socket.getInputStream();  DataInputStream dis = new DataInputStream(is);  System.out.println(dis.readUTF());  System.out.println(dis.readUTF());  dis.close();  socket.close();  }  } |

### UDP套接字

UDP是面向无连接的不可靠的传输方式，传输的数据包中包含了完整的地址信息，传输的数据包大小有64K的限制。





|  |
| --- |
| **package** com.test.net;  **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetAddress;  **class** UDPClient{  **public** **static** **void** main(String[] args)**throws** IOException{  DatagramSocket client = **new** DatagramSocket();    String sendStr = "Hello! I'm Client";  **byte**[] sendBuf;  sendBuf = sendStr.getBytes();  InetAddress addr = InetAddress.*getByName*("127.0.0.1");  **int** port = 5050;  DatagramPacket sendPacket  = **new** DatagramPacket(sendBuf ,sendBuf.length , addr , port);  client.send(sendPacket);  **byte**[] recvBuf = **new** **byte**[100];  DatagramPacket recvPacket  = **new** DatagramPacket(recvBuf , recvBuf.length);  client.receive(recvPacket);  String recvStr = **new** String(recvPacket.getData() , 0 ,recvPacket.getLength());  System.***out***.println("收到:" + recvStr);  client.close();  }  } |

|  |
| --- |
| **package** com.test.net;  **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetAddress;  **class** UDPServer{  **public** **static** **void** main(String[] args)**throws** IOException{  DatagramSocket server = **new** DatagramSocket(5050);  **byte**[] recvBuf = **new** **byte**[100];  DatagramPacket recvPacket  = **new** DatagramPacket(recvBuf , recvBuf.length);  server.receive(recvPacket);  String recvStr = **new** String(recvPacket.getData() , 0 , recvPacket.getLength());  System.***out***.println("Hello World!" + recvStr);  **int** port = recvPacket.getPort();  InetAddress addr = recvPacket.getAddress();  String sendStr = "Hello ! I'm Server";  **byte**[] sendBuf;  sendBuf = sendStr.getBytes();  DatagramPacket sendPacket  = **new** DatagramPacket(sendBuf , sendBuf.length , addr , port );  server.send(sendPacket);  server.close();  }  } |

### 聊天案例

|  |
| --- |
| package com.test.net;  import java.io.IOException;  import java.net.ServerSocket;  import java.net.Socket;  public class ChatServerV1 {  public static void main(String[] args){  try {  ServerSocket server = new ServerSocket(8888);  while(true){  Socket socket = server.accept();  ReceiveMessage receiveMessage = new ReceiveMessage(socket);  SendMessage sendMessage = new SendMessage(socket);    Thread receiver = new Thread(receiveMessage,"服务端接收");  Thread sender = new Thread(sendMessage,"服务端发送");  receiver.start();  sender.start();  }  } catch (IOException e) {  e.printStackTrace();  }    }  } |

|  |
| --- |
| package com.test.net;  import java.io.IOException;  import java.net.Socket;  import java.net.UnknownHostException;  public class ChatClientV1 {  public static void main(String[] args){  try {  Socket socket = new Socket("localhost",8888);  ReceiveMessage receiveMessage = new ReceiveMessage(socket);  SendMessage sendMessage = new SendMessage(socket);    Thread receiver = new Thread(receiveMessage,"客户端接收信息");  Thread sender = new Thread(sendMessage,"客户端发送消息");  receiver.start();  sender.start();  } catch (UnknownHostException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }    }  } |

|  |
| --- |
| package com.test.net;  import java.io.DataInputStream;  import java.io.IOException;  import java.io.InputStream;  import java.net.Socket;  public class ReceiveMessage implements Runnable{  private Socket socket;    public ReceiveMessage(Socket socket){  this.socket = socket;  }    @Override  public void run() {  try {  InputStream inputStream = socket.getInputStream();  DataInputStream dataInputStream = new DataInputStream(inputStream);  while(true){  String message = dataInputStream.readUTF();  System.out.println(socket.getInetAddress() + ":" + message);  }  } catch (IOException e) {  e.printStackTrace();  }  }  } |

|  |
| --- |
| package com.test.net;  import java.io.BufferedReader;  import java.io.DataOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.io.OutputStream;  import java.net.Socket;  public class SendMessage implements Runnable{  private Socket socket;    public SendMessage(Socket socket){  this.socket = socket;  }    @Override  public void run() {  try {  OutputStream outputStream = socket.getOutputStream();  DataOutputStream dataOutputStream = new DataOutputStream(outputStream);  InputStream inputStream = System.in;  BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));  while(true){  String message = bufferedReader.readLine();  dataOutputStream.writeUTF(message);  if(message.equals("bye")){  break;  }  }  bufferedReader.close();  dataOutputStream.close();  } catch (IOException e) {  e.printStackTrace();  }    }    } |

|  |
| --- |
| package com.test.net;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.ObjectInputStream;  import java.io.ObjectOutputStream;  import java.io.Serializable;  public class Serizialzation {  public static void main(String[] args){  Student s1 = new Student(1,"小网","123456");  Student s2 = new Student(3,"小李","123456");  //写文件  try {  FileOutputStream fos = new FileOutputStream("C:\\Users\\JokerQ\\Desktop\\student.txt");  ObjectOutputStream oos = new ObjectOutputStream(fos);  oos.writeObject(s1);  oos.writeObject(s2);  oos.close();  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }    //读文件  try {  FileInputStream fis = new FileInputStream("C:\\Users\\JokerQ\\Desktop\\student.txt");  ObjectInputStream ois = new ObjectInputStream(fis);  Student s3 = (Student)ois.readObject();  Student s4 = (Student)ois.readObject();  ois.close();    System.out.println("s3信息：" + s3.getId() +","+ s3.getName() +","+ s3.getPassword());  System.out.println("s4信息：" + s4.getId() +","+ s4.getName() +","+ s4.getPassword());  } catch (Exception e) {  e.printStackTrace();  }  }  }  class Student implements Serializable{  private int id;  private String name;  private String password;  private String ip;    public Student(int id,String name,String password){  this.id = id;  this.name = name;  this.password = password;  }    public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getIp() {  return ip;  }  public void setIp(String ip) {  this.ip = ip;  }    } |

|  |
| --- |
| package com.test.net;  import java.io.BufferedReader;  import java.io.DataInputStream;  import java.io.DataOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.io.OutputStream;  import java.net.ServerSocket;  import java.net.Socket;  public class One2OneClient {  public static void main(String[] args) throws Exception {  // ServerSocket server=new ServerSocket(8888);  final SocketTalkServer3 server3 = new SocketTalkServer3();  Thread t2 = new Thread() {  @Override  public void run() {  try {  server3.test2();  } catch (IOException e) {  e.printStackTrace();  }  }  };  t2.start();  // Thread.sleep(500);  Thread t1 = new Thread() {  @Override  public void run() {  try {  server3.test1();  } catch (IOException e) {  e.printStackTrace();  }  }  };  t1.start();  }  }  class SocketTalkServer3{  private ServerSocket server;  private Socket socket;    public void test1() throws IOException {  InputStream is = socket.getInputStream();  DataInputStream dis = new DataInputStream(is);  while (true) {  String s = dis.readUTF();  System.out.println(socket.getInetAddress() + ":" + s);  if (s.equals("bye")) {  break;  }  }  }  public void test2() throws IOException {  OutputStream os = socket.getOutputStream();  DataOutputStream dos = new DataOutputStream(os);  InputStream sis = System.in;  BufferedReader br = new BufferedReader(new InputStreamReader(sis));  while (true) {  String replay = br.readLine();  dos.writeUTF(replay);  if (replay.equals("bye")) {  break;  }  }  }  } |