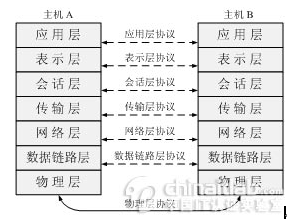
# 网络编程

**本机的IP地址可用127.0.0.1表示，也可以用localhost表示。**

## 一、OSI参考模型(OSI七层模型)

OSI参考模型(OSI/RM)：Open System Interconnection Reference Model，即开放系统互连参考模型。它是由国际标准化组织（International Standard Organization，ISO）提出的一个网络系统互连模型。

OSI七层模型：



1.物理层作用：与传输媒体的接口,完成传输媒体上的信号和二进制数据间的转换。物理层的传输单位为比特（bit）0 1。

2.数据链路层的作用：物理地址寻址、数据的成帧、流量控制、数据的检错、重发,链路管理等。数据链路层的单位称为帧（frame）。

3.网络层的作用：提供主机到主机的通信。网络层将实现的功能包括：选择路由，流量控制，协议的转换，分段和重组，对用户的分组、字符等计数等等。网络层的单位是 (packet)。低三层的通信对象通常是路由器。

4.传输层的作用：提供端到端，应用到应用的通路。传输层协议的代表包括：TCP、UDP、SPX等。

5.会话层的作用：会话层管理主机之间的会话进程，即负责建立、管理、终止进程之间的会话。

6.表示层的作用：表示层对上层数据或信息进行变换以保证一个主机应用层信息可以被另一个主机的应用程序理解。表示层的数据转换包括数据的加密、压缩、格式转换等。

7.应用层的作用：应用层为操作系统或网络应用程序提供访问网络服务的接口。应用层协议的代表包括：Telnet、FTP、HTTP、SNMP等。

## 二、OSI七层模型的数据封装过程(理解这个很重要)：

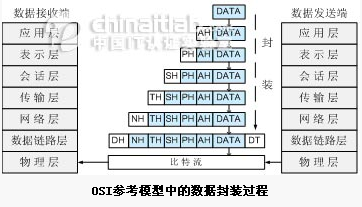
1.当一台主机需要传送用户的数据（DATA）时，数据首先通过应用层的接口进入应用层。在应用层，用户的数据被加上应用层的报头（Application Header，AH），形成应用层协议数据单元（Protocol Data Unit，PDU），然后被递交到下一层-表示层。

2.表示层并不"关心"上层-应用层的数据格式而是把整个应用层递交的数据包看成是一个整体进行封装，即加上表示层的报头（Presentation Header，PH）。然后，递交到下层-会话层。

3.同样，会话层、传输层、网络层、数据链路层也都要分别给上层递交下来的数据加上自己的报头。它们是：会话层报头（Session Header，SH）、传输层报头（Transport Header，TH）、网络层报头（Network Header，NH）和数据链路层报头（Data link Header，DH）。其中，数据链路层还要给网络层递交的数据加上数据链路层报尾（Data link Termination，DT）形成最终的一帧数据。

4.当一帧数据通过物理层传送到目标主机的物理层时，该主机的物理层把它递交到上层-数据链路层。数据链路层负责去掉数据帧的帧头部DH和尾部DT（同时还进行数据校验）。如果数据没有出错，则递交到上层-网络层。

5.同样，网络层、传输层、会话层、表示层、应用层也要做类似的工作。最终，原始数据被递交到目标主机的具体应用程序中。



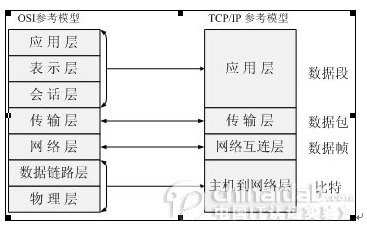
## 三、TCP/IP模型

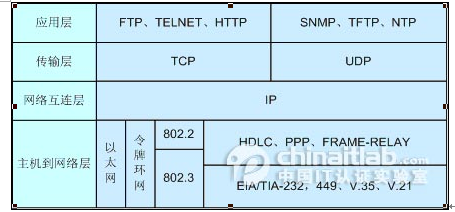
由于ISO制定的OSI参考模型的过于庞大、复杂，因此由技术人员自己开发的**TCP/IP协议栈**获得了更为广泛的应用。

TCP/IP参考模型分为四个层次：**应用层、传输层、网络互连层和主机到网络层。**

1.**主机到网络层**，有时也称作数据链路层或网络接口层。负责数据帧的发送和接收，帧是独立的网络信息传输单元。网络接口层将帧放在网上，或从网上把帧取下来。

2.**网络互连层**：是整个TCP/IP协议栈的核心。它的功能是把分组发往目标网络或主机。网络互连层定义了分组格式和协议，即IP协议（Internet Protocol）。





3.**传输层**：在TCP/IP模型中，传输层的功能是使源端主机和目标端主机上的对等实体可以进行会话。在传输层定义了两种服务质量不同的协议，即：**传输控制协议TCP（transmission control protocol）和用户数据报协议UDP（user datagram protocol）**。

①**TCP协议是一个面向连接的、可靠的协议**。它将一台主机发出的字节流无差错地发往互联网上的其他主机。在发送端，它负责把上层传送下来的字节流分成报文段并传递给下层。在接收端，它负责把收到的报文进行重组后递交给上层。TCP协议还要处理端到端的流量控制，以避免缓慢接收的接收方没有足够的缓冲区接收发送方发送的大量数据。通过TCP协议传输，得到的是一个**顺序的无差错的数据流**。

它可以保证数据从连接的一方传递到另一方，并且发送数据的顺序和所接收数据的顺序一致。

TCP采用了超时重传、发送和接收端到端的确认分组等机制。

②**UDP协议是一个不可靠的、无连接协议**，主要适用于不需要对报文进行排序和流量控制的场合。

UDP没有检测错误的机制，也不重发丢失的数据。接收到的数据包的顺序可能与发送的数据包的顺序不一致。每个数据报都是一个独立的信息，包括完整的源地址或目的地址。

**采用UDP进行通信时，事先不需要建立连接。而采用TCP进行通信时，首先要建立一个连接。TCP的通信质量比UDP高，UDP的开销比TCP小。**

当我们使用java编写网络程序时，不用关心TCP和UDP的细节。使用java.net包中的类就可以编写出平台无关的网络程序。

4. **应用层**：应用层面向不同的网络应用引入了不同的应用层协议。

①有**基于TCP协议**的：文件传输协议（File Transfer Protocol，FTP）、虚拟终端协议（TELNET）、超文本链接协议（Hyper Text Transfer Protocol，HTTP）

②有**基于UDP协议**的：简单网络管理协议(Simple Network Management Protocol，SNMP)、简单文件传输协议(Trivial File Transfer Protocol，TFTP)、网络时间协议(Network Time Protocol，NTP)。

网桥是在链路层上对网络进行互连，而路由器则是在网络层上对网络进行互连。TCP /IP倾向于使用路由器而不是网桥来连接网络。

ARP协议是“Address Resolution Protocol”（地址解析协议）的缩写。

## 四、端口

在网络上可以用IP地址来唯一的标识一台计算机。IP地址（IPv4）是四个用点隔开的数字，总共32位，每个数字8位（表示范围：0～255），例如：192.168.10.22。（IPv6地址有128位，地址范围更大）。

**端口port与IP地址一起可以为网络的应用程序之间提供一种地址标识功能**。如同一台计算机上的多个服务程序，每个服务程序都有相应的port提供服务。

port通常称为握手点handshake point，它被客户用来定位服务器计算机上的服务应用程序。**端口号范围：0～65535。**可以是范围中的任何一个数字。通常，OS将1024以下的端口号保留给系统服务用。自己一般使用1024之后的端口号。

客户端程序要和服务程序交互，首先要找到服务程序所在的机器（可以通过IP地址），然后在这台机器上找到服务程序（通过port）。客户端程序必须事先知道它所请求的服务程序对应的端口号。

## 五、网络编程

网络编程的**目的**就是指直接或间接地通过网络协议与其他计算机进行通讯。

网络编程中有两个主要的问题：1.如何准确的定位网络上一台或多台主机？2.找到主机后如何可靠高效的进行数据传输？

在TCP/IP协议中IP层主要负责网络主机的定位，由IP地址可以唯一地确定Internet上的一台主机。TCP层则提供面向应用的可靠的或非可靠的数据传输机制。虽然在TCP/IP协议的名称中只有TCP这个协议名，但是TCP/IP协议在传输层中是同时存在TCP和UDP两个协议的。

### 1.TCP和UDP传输数据的区别

**TCP协议是一个面向连接的、可靠的协议。UDP协议是一个不可靠的、无连接协议。**

**①发送/接收数据的顺序：**TCP协议发送数据的顺序和所接收数据的顺序一致，得到的是一个顺序的无差错的数据流。UDP协议接收到的数据包的顺序可能与发送的数据包的顺序不一致。

**②是否建立连接：**TCP协议，它是一个面向连接的协议，在socket之间进行数据传输之前必需要建立连接。UDP协议中，每个数据报中都给出了完整的地址信息，因此无需建立发送方和接收方的连接。

**③传输数据的大小**：TCP协议传输的数据没有大小限制。UDP协议传输数据时是有大小限制的，每个被传输的数据报必须限定在64KB之内。

**④传输效率**：TCP协议传输数据的效率不如UDP，因为TCP中多了一个连接建立的时间以及对数据内容正确性的检验时间等。

**⑤**TCP的通信质量比UDP高，UDP的开销比TCP小**。**

**常见的TCP协议有：Http(超文本传输协议)、FTP(文件传输协议)。**

**常见的UDP协议有：NTP(网络时间协议)、SNMP(简单网络管理协议)。**

2.有了可靠传输的TCP协议为什么还要非可靠的UDP协议？

答：①TCP对数据内容正确性的检验必然占用计算机的处理时间和网络的带宽，因此TCP传输的效率不如UDP高②在许多应用中并不需要保证严格的传输可靠性，如视频会议系统，并不要求音频视频数据绝对的正确，只要保证连贯性即可。

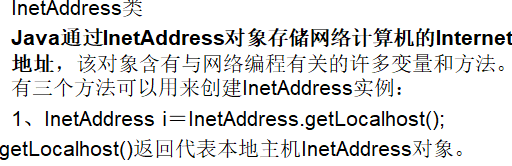
## 六、URL地址

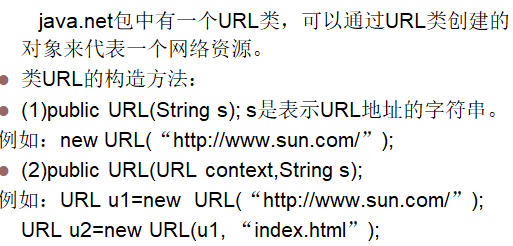
**URL(Uniform Resource Locator)是统一资源定位器，以一个字符串的形式表示Internet上某一资源的位置。**

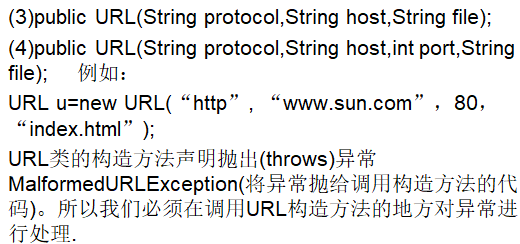
URL地址由两部分组成：协议名和资源名，两者用“ : ”隔开。协议名指出了访问该资源所使用的网络协议，如http、ftp等。资源名是网络资源的完整地址，包括主机名、端口号、文件名和文件的一个引用。

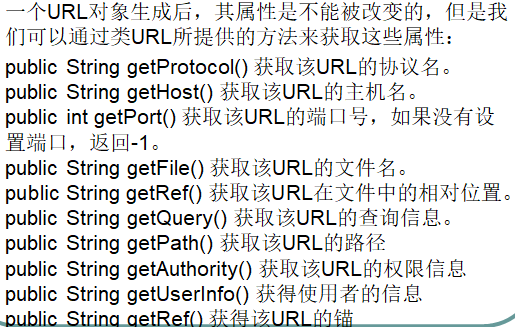
java.net包中有一个URL类，可以通过URL类创建的对象来代表一个网络资源(有空了解一下)

用空了解URL类和InetAddress类









## 七、Socket编程

在各种网络的C/S(Client/Server)应用中，客户机与服务器之间的通信组件是多种多样的，大部分通信组件内最底层的核心通信机制都是使用传输层接口的socket机制来实现的。

**socket是在两个应用程序之间用来进行双向数据传输的网络通信端点，一般由一个地址加一个端口号来识别**。**socket是进程之间通信的抽象连接点**。客户程序和服务程序通过一个双向的通信连接实现数据交换，这个双向通路的每一端就是一个socket。

**1.两种socket**：**流式socket、数据报式socket**。

**①TCP是一种流式socket协议。流式socket提供了双向的、有序的、无重复的数据流服务。**

**②UDP是一种数据报式socket协议。数据报式socket支持双向的数据流，但不能保证有序、无重复。**

2.在网络编程时，需要产生一个 socket 来建立与另一台机器的连接。一个连线对应于两种socket类：**ServerSocket 用来让服务器监听连线、Socket (一般命名为ClientSocket)用来让客户端建立连线。**连接建立之后，可以从socket中获得**InputStream和OutputStream对象**，输入输出流对象可以对数据进行处理。当客户端建立连线后，ServerSocket就会**通过accept( )方法返回对应的Socket**，通过返回的Socket就可以开始通信。

ServerSocket的主要作用是等待客户机连接上来，然后返回一个Socket。**生成ServerSocket对象时只需指定port**，因为它已经位于运行服务程序的机器上了。

**在产生Socket(ClientSocket)对象时，必须同时指定IP地址和port**(服务程序所对应的端口号)。

**端口号范围：0～65535。**可以是范围中的任何一个数字。通常， 1024以下的端口号保留给系统服务用。自己一般使用1024之后的端口号。

**3.TCP中Socket的编写(有空再网上多学习点)**

**服务器端程序的编写**：

**①步骤一：以某个端口号为参数**，**创建一个ServerSocket对象**。**该对象代表在指定端口监听客户机连接请求的socket**。

**②步骤二：服务程序使用ServerSocket对象的accept()方法接收来自客户程序的请求。该方法接收到从客户程序发来的连接请求之后返回一个新建的Socket类的对象。**该对象代表和客户方建立的通信链路在服务程序内的通信端点。该对象可以通过getInetAddress()方法返回连接的客户机的地址。注意accept()方法若没接收到客户程序发来的请求则一直等待在那。

**③步骤三：获取输入输出流对象，使服务器能够通过输入输出流与客户机交互**。

**④步骤四：通信完毕关闭流和socket。**

**public** ServerSocket(**int** port)：这个构造方法是在指定的端口创建一个ServerSocket对象。若端口号为0，则在一个临时的端口创建socket对象。即将连接的客户请求最多为50个，若队列已满，则服务器拒绝新的连接请求。

**public** ServerSocket(**int** port, **int** backlog)：这个构造方法第二个参数是队列中等待连接的请求的最大数量。

**public** ServerSocket(**int** port, **int** backlog, InetAddress bindAddr)：这个构造方法第三个参数是服务器将要绑定的地址。

**客户端程序的编写：**

**①步骤一：创建一个Socket对象，实现与服务器的连接**。

**public** Socket(String host, **int** port)：**参数一是服务器名称的字符串，也可以是服务器的IP地址。参数二是端口号。**服务程序要在这个端口号上提供服务，**即创建ServerSocket时指定的端口号**。

**②步骤二：客户程序使用socket对象得到输入输出流，然后进行通信。**

**③步骤三：通信结束，关闭流和socket**。

Socket**.getInetAddress()**：返回已经连接的Socket的IP地址。

[Socket](eclipse-javadoc:%E2%98%82=tcpPro/E:%5C/JavaRunEnviroment%5C/lib%5C/rt.jar%3Cjava.net(Socket.class%E2%98%83Socket)**.getOutputStream()**：返回一个OutputStream对象。

[Socket](eclipse-javadoc:%E2%98%82=tcpPro/E:%5C/JavaRunEnviroment%5C/lib%5C/rt.jar%3Cjava.net(Socket.class%E2%98%83Socket).**getInputStream()**：返回一个InputStream对象。

**注意**Socket中得到的是输入输出流是字节流，对于读取文本内容不太方便，最好转换成字符流。

**注意**客户程序和服务程序之间进行通信时，一定要注意流的关闭顺序，不能提前关闭了输入输出流，否则通信过程中会抛Socket is Closed异常。一般抛Socket is Closed异常有两种情况：提前关闭了Socket或者提前关闭了输入输出流。

**注意**下面在使用输出流BufferedWriter时，写完内容一定要记得换行，即bfw.**newLine()**，否则可能会报异常。

Socket.getLocalAddress()、Socket的一些方法用空了解一下。

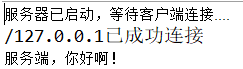
**注意**while和try...catch一起连在一起使用时，while放在try...catch中间，而不是while中间放try...catch。因为while循环中间放try..catch，若一报异常则循环中断，被catch捕获异常。若while放在try..catch外面，若出现异常，则死循环的方式不断捕获出现的异常。

**单线程中客户程序和服务程序通信的例子**：



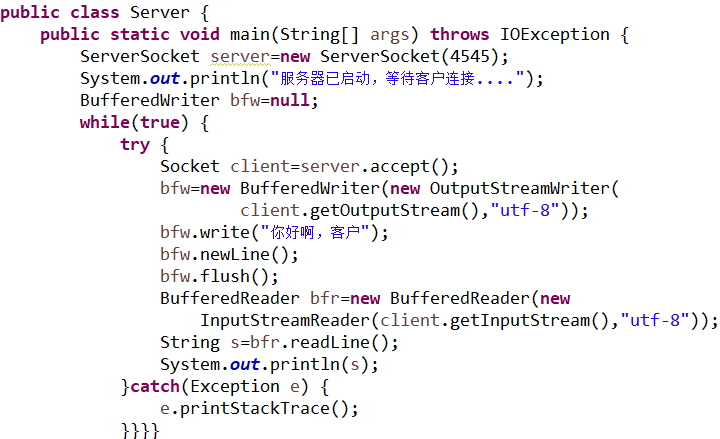


在服务程序的控制台上显示的内容如下：

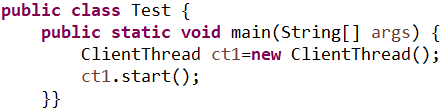


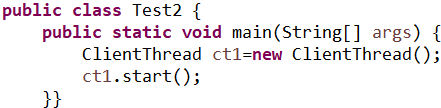
在客户程序的控制台上显示的内容：“客户端，你好啊”。注意这句话是服务程序中发送过来的。

**同一个服务程序和多个客户通信的例子：**

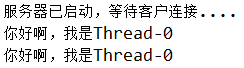


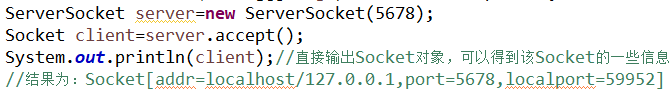






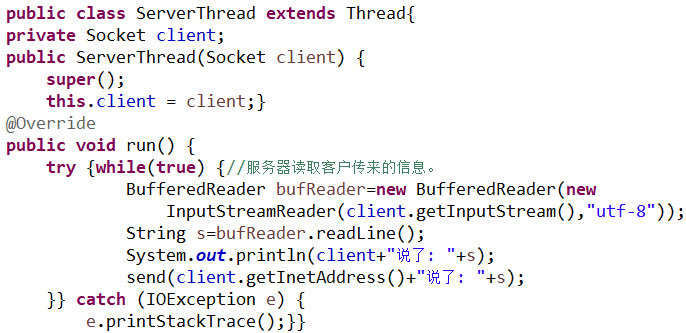
服务程序的控制台上的输出结果为：



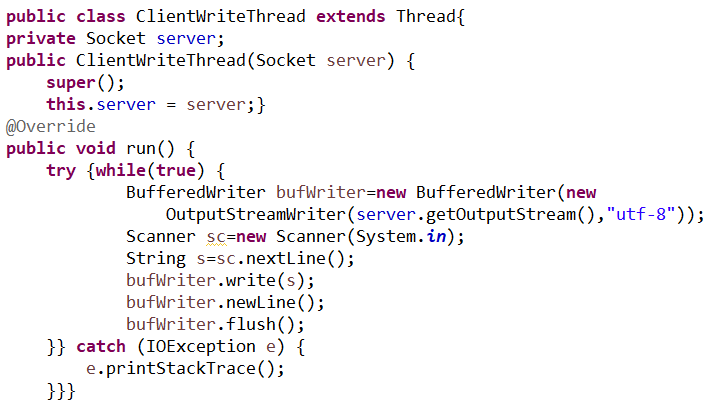


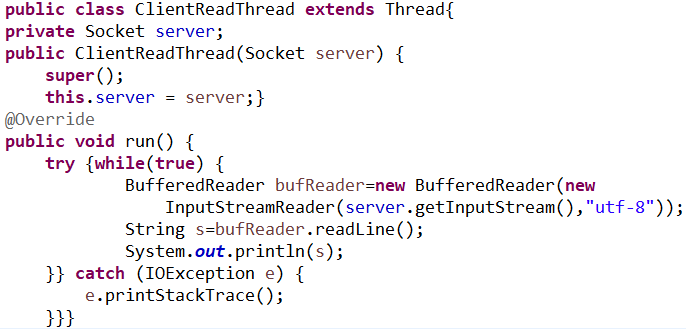
**客户向服务程序发送信息，服务程序可以将信息显示在其它客户的控制台上。例子(好好理解这个例子)(代码以后需改进)：**

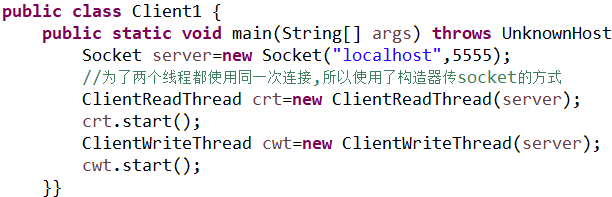


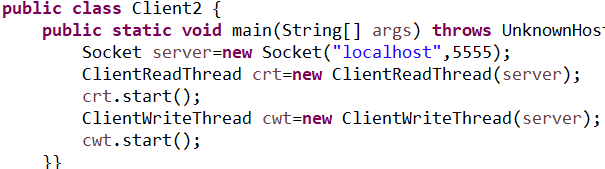








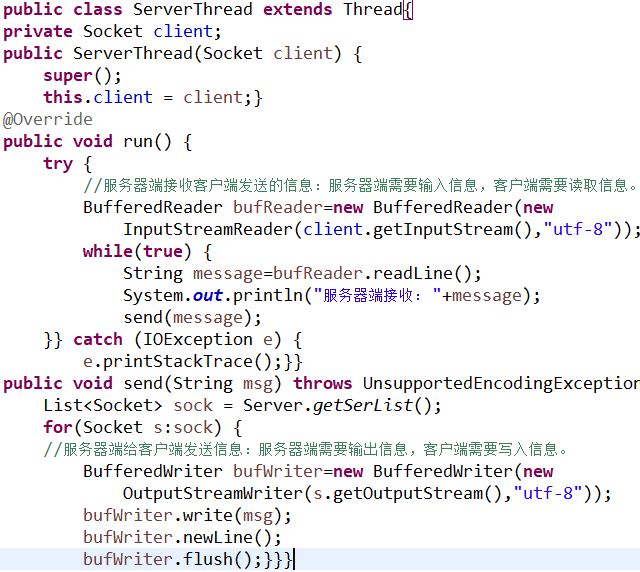




上面这个例子有点bug，即当客户关闭了谈话，服务程序会报异常。以后改进：看当一个客户关闭了谈话，服务程序向其它客户发送该客户退出了聊天室的信息，并把客户退出聊天导致报的异常解决。

**聊天窗口可视化例子(有点bug)**：











一个Server往往对应一个ServerThread线程，这个线程是用来接收Client发送过来的信息或者向Client发送信息。一个Client往往对应一个ClientReadThread线程和一个ClientWriteThread线程。**ServerThread线程中，服务器端接收客户端发送的信息：服务器端需要输入(InputStream)信息，客户端需要读取(read)信息，因此对应客户端的Socket.getInputStream然后read信息；服务器端给客户端发送信息：服务器端需要输出(OutputStream)信息，客户端需要写入(write)信息。ClientReadThread线程是客户端接收服务器发送的信息：客户端需要输入信息，服务端需要读取信息。ClientWriteThread线程是客户端向服务器发送信息：客户端要输出信息，服务端要写入信息。**

**4.UDP 中Socket的编写(以后有空网上多学点)**：

UDP适用于不要求错误检查和可靠性的网络应用程序，可靠性差，但速度快。在网络编程中，大多数采用TCP，偶尔采用UDP。

**UDP中使用DatagramSocket类和DatagramPacket类**。

**①DatagramSocket：用来发送和接收datagram packet。**

**public** DatagramSocket()**：**可以绑定本地主机的任何可用的端口。

**public** DatagramSocket(**int** port)：绑定本地主机中指定的端口。

**public** DatagramSocket(**int** port, InetAddress laddr)：绑定指定IP地址上的指定端口。

**②DatagramPocket：表示一个datagram packet**

**public** DatagramPacket(**byte** buf[], **int** length)：这个构造器是**用来接收数据的**。参数一是btye类型的缓冲数组，该数组用来保存将要接收的数据包。**参数length是读取数据的字节长度**。**注意这个length不能超过字节数组的长度，即buf.length**。

**public** DatagramPacket(**byte** buf[],**int** length,InetAddress address, **int** port)：该构造器是**用来发送数据的**。参数一是以byte的形式存放的数据包中的数据。参数length为数据包的长度，参数address为目标地址，参数port为目标端口号。**注意length的长度不能超过buf.length**。

TCP中Socket的编写过程中，要**先运行服务程序**，服务程序启动之后采用accept()方法等待客户程序进行连接，若客户程序没有连接则一直等待着。而UDP中Socket的编写过程中，先接收数据的一方要先启动，否则会接收不到数据。因为UDP中发送数据是无连接的，若先接收数据一方不先启动等待接收数据，则数据会丢失，接收不到。

**String s=new String(byte[] bytes)**：可以将byte类型的数组的数据换成字符串。

**InetAddress.getByName(String host)**：该方法可以给定String类型的主机名然后转换成InetAddress地址。



DatagramSocket**.send(DatagramPacket p)**：将数据包发送出去

DatagramSocket**.receive(DatagramPacket p)**：接收数据包

**UDP中服务端和客户端相互通信例子：**



