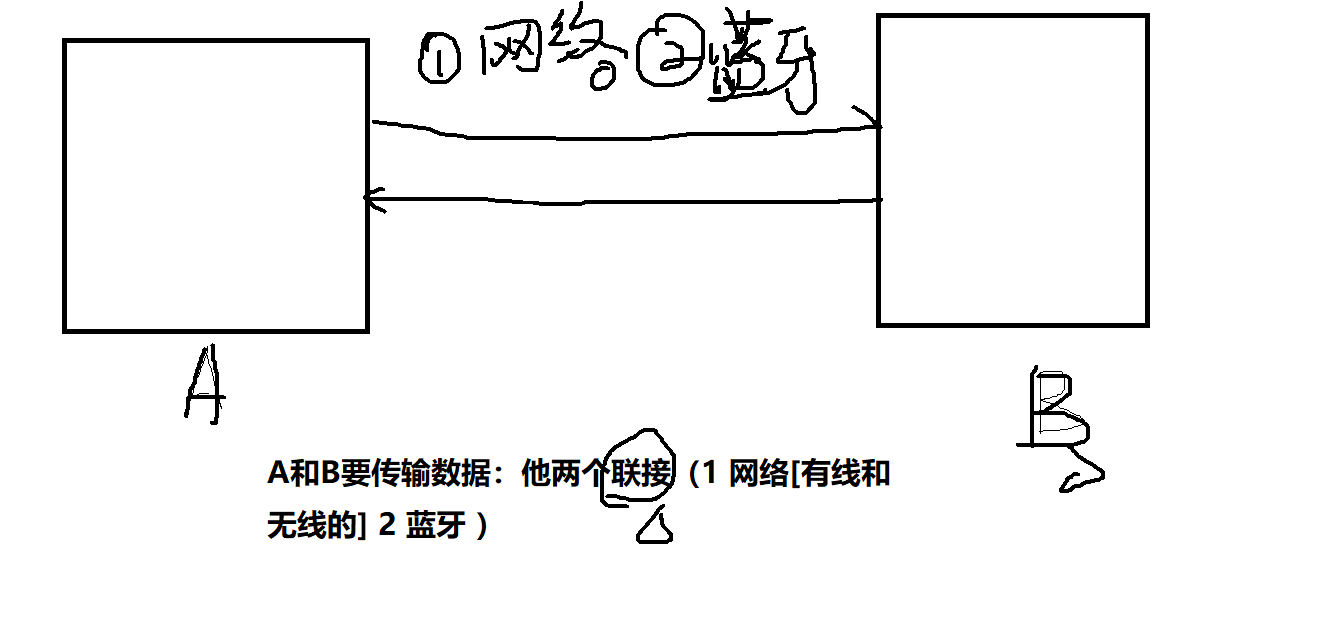
# 1 基本概念

## 1.1 两台电脑发送信息

我们前面主要学习是一台电脑读写自己磁盘上的数据，我们实际开发中需要一台电脑读取另外一台电脑的数据，这个时间我们需要解决下面问题。

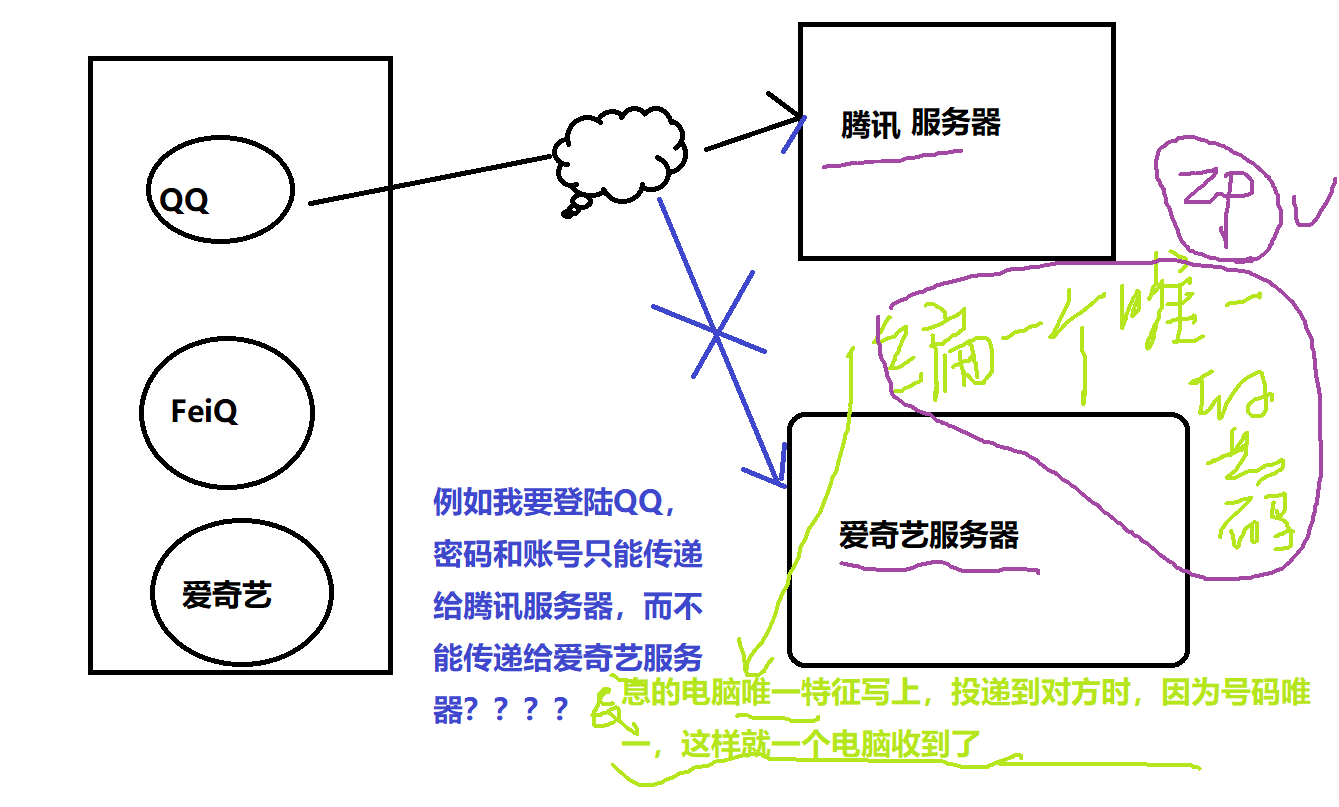
### 1.1.1 联接起来

正如你要能看到对方，需要有光；你要听到对方的声音，需要有空气(声音是空气震动传播)。我们两台电脑之间要传递信息，也需要媒介。通常的媒介有：1)长距离传输技术：计算机网络 2)短距离传输技术：蓝牙



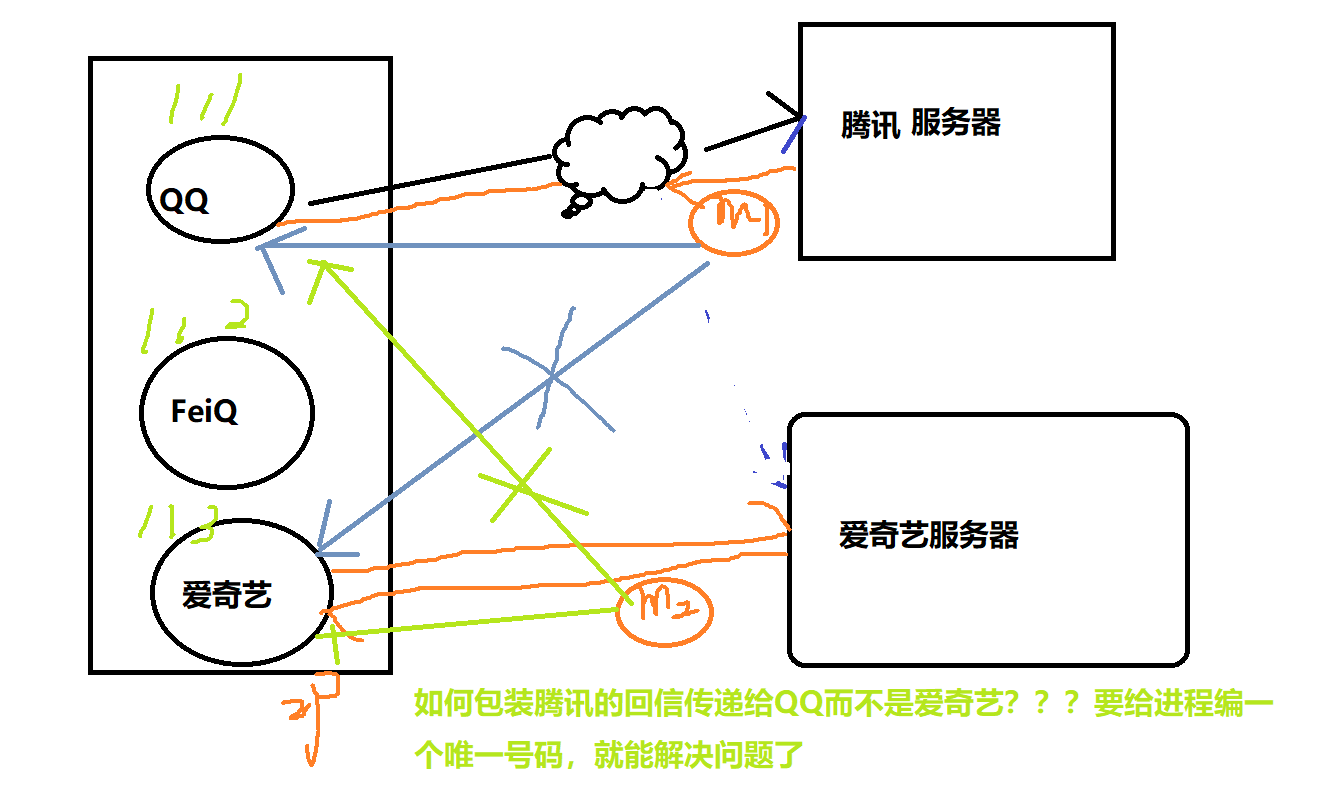
### 1.1.2 每个电脑要有唯一标识

多台电脑联接起来，我们保证信息是传递给电脑A而不是传递给B呢？？？我们电脑要有一个唯一的标识，在网络领域我们称之为ip.



### 1.1.3 唯一标识一个进程

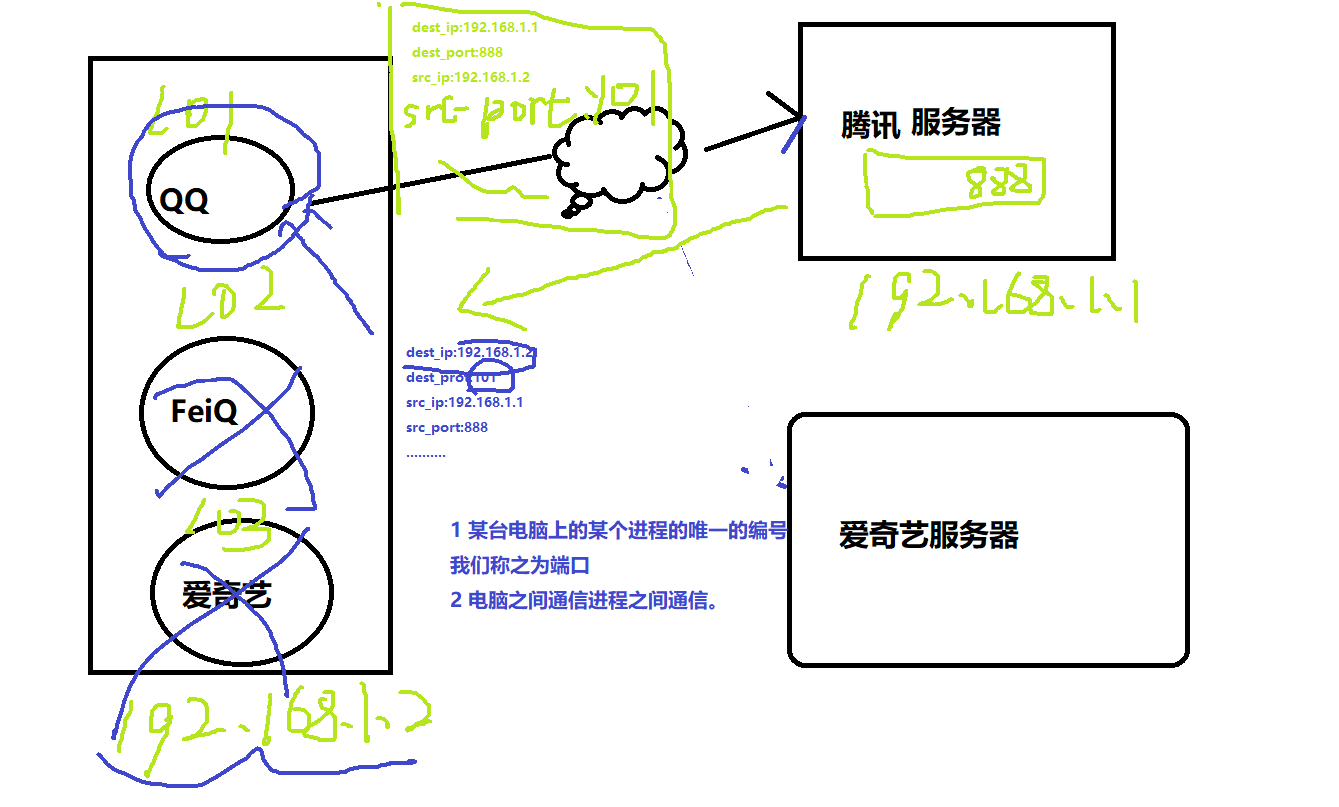
一个电脑可以运行多个程序，因而有多个进程，且同一台电脑一般只有一个IP，那么对方的回信，如何保证是进程A接收而不是进程B接收呢？？？？靠IP不行，因为他们是同一个IP，因而我们需要给某个电脑不同进程编一个号码，用于唯一区分一个进程，这个唯一的号码我们称之为端口。



注意：端口是用来区分某台电脑上不同进程，而IP用于区分不同电脑。因而不同电脑上端口号可以重复，而同一台电脑上端口号不可以重复。

### 1.1.4 如何通信

电脑之间通信精确地说是两个进程在通信，因而在发送信息时候，我们需要用目的IP表达发给那个电脑，目的端口表达该电脑某个进程接收。同时为了回信时，能直到该信息应该那个进程接收，我们源IP和端口也带上去。



## 1.2 信息传递两种方式

### 1.2.1 预先建立联接

例如你要打电话，你需要首先建立联接，如果联接不成功，你就不说话。当你说话的内容对方木有收到，你再说一遍。话是按照顺序到达对方。

在计算机网络领域方式我们称之为TCP。特点：

1. 预先联接
2. 可靠
3. 有序

### 1.2.2 只管投递

例如发送一个封信给某个人，不需要跟对方建立联接，只要投递出去，不用等待对方收到。

在计算机网络领域这种方式我们成为UDP，特点是：

1. 不需要事先建立联接
2. 不可靠：不保证信一定到对方
3. 不保证顺序：

### 1.2.3 UDP和TCP的对比

|  |  |  |
| --- | --- | --- |
|  | UDP | TCP |
| 优点 | 效率高(不需要建立联接) | 可靠(一定能收到和有序) |
| 缺点 | 不可靠(不保证一定收到和有序) | 效率低(建立联接需要耗费时间的) |

例如在线看电影，你关注的是流畅度(数据传输的效率)，偶尔出现丢失数据不是太大问题，这个时间我们用UDP。

# 2 TCP编程

## 2.1 相关的类

客户端类：java.net.Socket

服务器端类：**java.net.ServerSocket**

## 2.2 服务器单向发送消息

### 2.2.1 服务器端代码

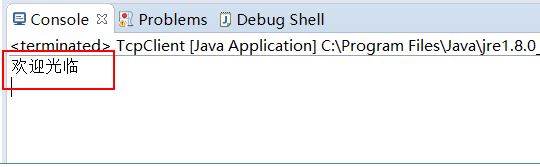
|  |
| --- |
| **import** java.io.IOException;  **import** java.io.PrintWriter;  **import** java.net.ServerSocket;  **import** java.net.Socket;  **public** **class** TcpServer {  /\*\*  \* 我们首先是服务器发送一个信息给顾客  \*/  **public** **static** **void** main(String[] args) **throws** IOException {  //1 开饭店  ServerSocket serverSocket=**new** ServerSocket(8888);  //2 等顾客来,如果有顾客来了交给包厢服务器服务员交流  Socket baoxiao=serverSocket.accept();  //3 包厢服务器和顾客  PrintWriter out=**new** PrintWriter(baoxiao.getOutputStream());  out.println("欢迎光临");  out.close();  }  } |

### 2.2.2 客户端代码

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  **public** **class** TcpClient {  **public** **static** **void** main(String[] args) **throws** UnknownHostException, IOException {  //找饭店  Socket socket=**new** Socket("localhost", 8888);  BufferedReader br=**new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));  String msg=br.readLine();  System.***out***.println(msg);  br.close();    }  } |

### 2.2.3 测试

因为现实生活中，是饭店先开门迎客，等顾客来，因而是服务器先运行，然后客户端再运行。



## 2.3 服务器双向发送消息

### 2.3.1 服务器端代码

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.ServerSocket;  **import** java.net.Socket;  **public** **class** TcpServer {  /\*\*  \* 我们首先是服务器发送一个信息给顾客  \*/  **public** **static** **void** main(String[] args) **throws** IOException {  //1 开饭店  ServerSocket serverSocket=**new** ServerSocket(8888);  //2 等顾客来,如果有顾客来了交给包厢服务器服务员交流  Socket baoxiao=serverSocket.accept();  //3 包厢服务器和顾客  PrintWriter out=**new** PrintWriter(baoxiao.getOutputStream());  out.println("欢迎光临");  out.flush();//强制发出  //收消息  BufferedReader br=**new** BufferedReader(**new** InputStreamReader(baoxiao.getInputStream()));  String msg=br.readLine();  System.***out***.println(msg);    out.close();  br.close();  }  } |

### 2.3.2 客户端代码

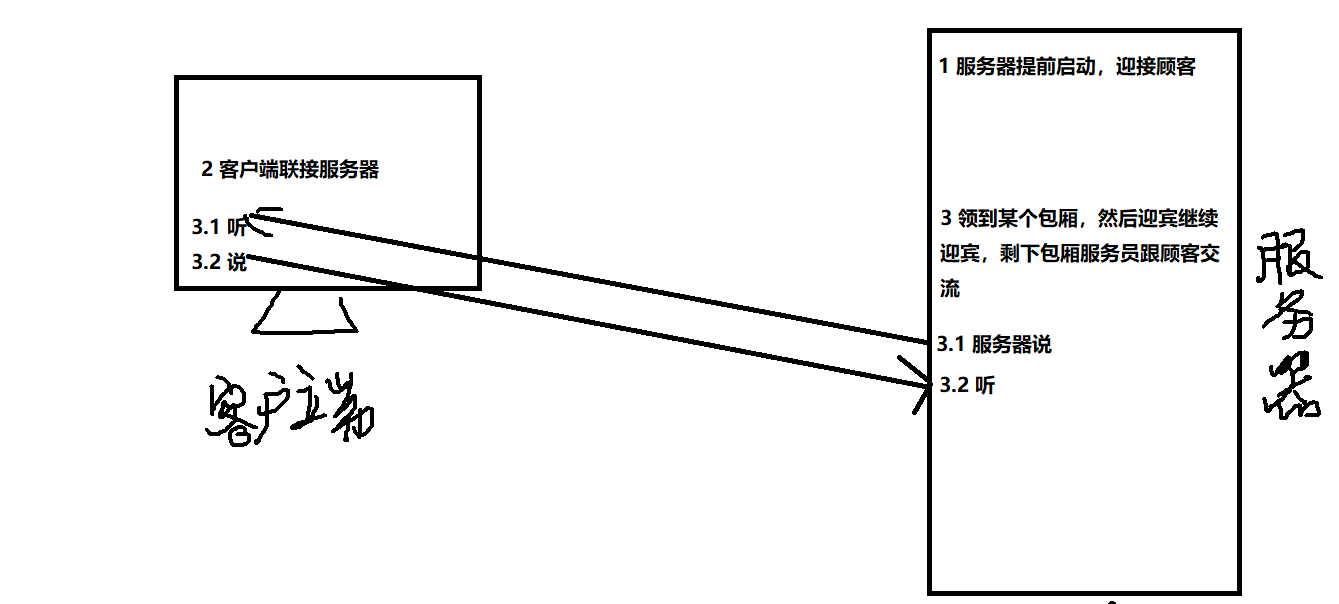
|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  **public** **class** TcpClient {  **public** **static** **void** main(String[] args) **throws** UnknownHostException, IOException {  //找饭店  Socket socket=**new** Socket("localhost", 8888);  BufferedReader br=**new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));  String msg=br.readLine();  System.***out***.println(msg);  //回话：您好美女  PrintWriter out=**new** PrintWriter(socket.getOutputStream());  out.println("您好美女");  out.flush();  br.close();  out.close();    }  } |

## 2.4 服务多个客户

对于服务性行业两个很成熟的模式是银行和饭店：

1. 首先饭店到点开门，由迎宾站在门口等待顾客到来
2. 顾客找到饭店
3. 迎宾把顾客领到某个包厢交给包厢服务员，不用等顾客吃完饭而继续迎宾
4. 接下来就是顾客和包厢服务器交流，完成吃饭动作

这种工作模式经过很长时间的检验，很成熟，我们编程时就去模拟它。



在这里我们需要强调一点的是：迎宾和吃饭时两个流程，我们前面的编码是在一个流程中实现。迎宾是一个循环过程，而包厢吃饭不是循环过程。我们需要重构上午代码实现。

### 2.4.1 服务器端代码—吃饭流程

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.Socket;  **public** **class** ServerThread **extends** Thread {  **private** Socket baoxiao;  **public** ServerThread(Socket baoxiao) {  **this**.baoxiao = baoxiao;  }  @Override  **public** **void** run() {  PrintWriter out = **null**;  BufferedReader br =**null**;  **try** {  // 3 包厢服务器和顾客  out = **new** PrintWriter(baoxiao.getOutputStream());  out.println("欢迎光临");  out.flush();// 强制发出  // 收消息  br = **new** BufferedReader(**new** InputStreamReader(baoxiao.getInputStream()));  String msg = br.readLine();  System.***out***.println(msg);  } **catch** (IOException e) {  e.printStackTrace();  } **finally** {  **if** (out!=**null**) {    out.close();  }  **if** (br!=**null**) {    **try** {  br.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  }  } |

### 2.4.2 服务器端代码—迎宾流程

|  |
| --- |
| **import** java.io.IOException;  **import** java.net.ServerSocket;  **import** java.net.Socket;  **public** **class** TcpServer {  /\*\*  \* 我们首先是服务器发送一个信息给顾客  \*/  **public** **static** **void** main(String[] args) **throws** IOException {  //1 迎宾  ServerSocket serverSocket=**new** ServerSocket(8888);  **while** (**true**) {    //2 等顾客来,如果有顾客来了交给包厢服务器服务员交流  Socket baoxiao=serverSocket.accept();  ///由于迎宾直到是否有顾客来，因而由它启动  **new** ServerThread(baoxiao).start();  }    }  } |

### 2.4.2 客户端代码

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  **public** **class** TcpClient {  **public** **static** **void** main(String[] args) **throws** UnknownHostException, IOException {  //找饭店  Socket socket=**new** Socket("localhost", 8888);  BufferedReader br=**new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));  String msg=br.readLine();  System.***out***.println(msg);  //回话：您好美女  PrintWriter out=**new** PrintWriter(socket.getOutputStream());  out.println("您好美女");  out.flush();  br.close();  out.close();  }  } |

## 2.5 处理异常

### 2.5.1 迎宾流程

|  |
| --- |
| **import** java.io.IOException;  **import** java.net.ServerSocket;  **import** java.net.Socket;  **public** **class** TcpServer {  /\*\*  \* 我们首先是服务器发送一个信息给顾客  \*/  **public** **static** **void** main(String[] args){  ServerSocket serverSocket=**null**;  **try** {  serverSocket=**new** ServerSocket(8888);  //1 迎宾  **while** (**true**) {  //2 等顾客来,如果有顾客来了交给包厢服务器服务员交流  Socket baoxiao=serverSocket.accept();  ///由于迎宾直到是否有顾客来，因而由它启动  **new** ServerThread(baoxiao).start();  }  } **catch** (IOException e) {  e.printStackTrace();  }**finally** {  **if** (serverSocket!=**null**) {  **try** {  serverSocket.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  }  } |

### 2.5.2 吃饭流程

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.Socket;  **public** **class** ServerThread **extends** Thread {  **private** Socket baoxiao;  **public** ServerThread(Socket baoxiao) {  **this**.baoxiao = baoxiao;  }  @Override  **public** **void** run() {  PrintWriter out = **null**;  BufferedReader br =**null**;  **try** {  // 3 包厢服务器和顾客  out = **new** PrintWriter(baoxiao.getOutputStream());  out.println("欢迎光临");  out.flush();// 强制发出  // 收消息  br = **new** BufferedReader(**new** InputStreamReader(baoxiao.getInputStream()));  String msg = br.readLine();  System.***out***.println(msg);  } **catch** (IOException e) {  e.printStackTrace();  } **finally** {  **if** (out!=**null**) {    out.close();  }  **if** (br!=**null**) {    **try** {  br.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  }  }  } |

### 2.5.3 客户端

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.io.PrintWriter;  **import** java.net.Socket;  **import** java.net.UnknownHostException;  **public** **class** TcpClient {  **public** **static** **void** main(String[] args) {  BufferedReader br=**null**;  PrintWriter out=**null**;  Socket socket=**null**;  **try** {  //找饭店  socket=**new** Socket("localhost", 8888);  br=**new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));  String msg=br.readLine();  System.***out***.println(msg);  //回话：您好美女  out=**new** PrintWriter(socket.getOutputStream());  out.println("您好美女");  out.flush();    } **catch** (UnknownHostException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }**finally** {  **if** (br!=**null**) {    **try** {  br.close();  } **catch** (IOException e) {  e.printStackTrace();  }  }  **if** (out!=**null**) {    out.close();  }  **if** (socket!=**null**) {  **try** {  socket.close();//打招呼  } **catch** (IOException e) {  e.printStackTrace();  }  }  }    }  } |

# 3 UDP编程

## 3.1 相关类

由java.net.DatagramSocket来表示，它类似于邮局(信)/手机(短信)

信类:java.net.DatagramPacket来表示

## 3.2 服务一个客户

### 3.2.1 服务器

|  |
| --- |
| **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **public** **class** UdpServer {  **public** **static** **void** main(String[] args) **throws** IOException {  //邮局  DatagramSocket ds=**new** DatagramSocket(8888);  //等顾客信过来  //信的存放  **byte**[] buf=**new** **byte**[30];  DatagramPacket p=**new** DatagramPacket(buf, buf.length);  ds.receive(p);  System.***out***.println(**new** String(buf,0,p.getLength()));  //回信:给谁个回信，从信封上读取  String msg="hello dasb";  DatagramPacket p2=**new** DatagramPacket(msg.getBytes(), msg.getBytes().length, p.getSocketAddress());  ds.send(p2);    }  } |

### 3.2.2 客户端

|  |
| --- |
| **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetSocketAddress;  **import** java.net.SocketException;  **public** **class** UdpClient {  **public** **static** **void** main(String[] args) **throws** IOException {  //1 找个邮局  DatagramSocket ds=**new** DatagramSocket();  //发送  String msg="Hello author";  DatagramPacket p1=**new** DatagramPacket(msg.getBytes(), msg.getBytes().length, **new** InetSocketAddress("localhost", 8888));  ds.send(p1);  //收一收  **byte**[] buf=**new** **byte**[30];  DatagramPacket p2=**new** DatagramPacket(buf, buf.length);  ds.receive(p2);  System.***out***.println(**new** String(buf,0,p2.getLength()));    }  } |

## 3.3 服务多个客户

### 3.3.1 服务器

|  |
| --- |
| **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.SocketException;  **public** **class** UdpServer {  **public** **static** **void** main(String[] args) {  DatagramSocket ds=**null**;  **try** {  //邮局  ds=**new** DatagramSocket(8888);  **while** (**true**) {  //等顾客信过来  //信的存放  **byte**[] buf=**new** **byte**[30];  DatagramPacket p=**new** DatagramPacket(buf, buf.length);  ds.receive(p);  System.***out***.println(**new** String(buf,0,p.getLength()));  //回信:给谁个回信，从信封上读取  String msg="hello "+p.getAddress();  DatagramPacket p2=**new** DatagramPacket(msg.getBytes(), msg.getBytes().length, p.getSocketAddress());  ds.send(p2);  }  } **catch** (SocketException e) {    e.printStackTrace();  } **catch** (IOException e) {    e.printStackTrace();  }**finally** {  **if** (ds!=**null**) {  ds.close();  }  }  }  } |

### 3.3.2 客户端

|  |
| --- |
| **import** java.io.IOException;  **import** java.net.DatagramPacket;  **import** java.net.DatagramSocket;  **import** java.net.InetSocketAddress;  **import** java.net.SocketException;  **public** **class** UdpClient {  **public** **static** **void** main(String[] args) {  DatagramSocket ds=**null**;  **try** {  //1 找个邮局  ds=**new** DatagramSocket();  //发送  String msg="Hello author";  DatagramPacket p1=**new** DatagramPacket(msg.getBytes(), msg.getBytes().length, **new** InetSocketAddress("localhost", 8888));  ds.send(p1);  //收一收  **byte**[] buf=**new** **byte**[30];  DatagramPacket p2=**new** DatagramPacket(buf, buf.length);  ds.receive(p2);  System.***out***.println(**new** String(buf,0,p2.getLength()));  } **catch** (SocketException e) {  e.printStackTrace();  } **catch** (IOException e) {  e.printStackTrace();  }**finally** {  **if** (ds!=**null**) {  ds.close();  }  }    }  } |

# 4 协议

## 4.1 需求

### 4.1.1 要遵循同一个标准

我们互联网全球计算机组成一个网络，因而采用统一的标准来组网。

例如中国电压220V，美国110V，我们的笔记本变压器无法联通了，因为它们标准不一样！！！！

例如我们水晶头可以全球通用，因为遵循相同标准

### 4.1.2 无政府的国际权威组织指定

我们的标准让世界各国都遵守，因而标准的制定者要：

1. 权威：不权威别人不遵守
2. 无政府：不会政治利益
3. 国际：因为标准让全球各国遵守

### 4.1.2 国际权威组织

1. W3C:

万维网联盟创建于1994年，是Web技术领域最具权威和影响力的国际中立性技术标准机构。到目前为止，W3C已发布了200多项影响深远的Web技术标准及实施指南.



1. ISO

国际标准化组织（International Organization for Standardization，ISO）简称ISO，是一个全球性的[非政府组织](https://baike.baidu.com/item/%E9%9D%9E%E6%94%BF%E5%BA%9C%E7%BB%84%E7%BB%87/1416088)，是国际标准化领域中一个十分重要的组织。

1. IETF

互联网工程任务组，成立于1985年底，是全球互联网最具权威的技术标准化组织，主要任务是负责互联网相关技术规范的研发和制定，当前绝大多数国际互联网技术标准出自IETF。

## 4.2 协议

国际标准化组织定义的标准称之为协议，协议就是多方的约定。

例如IP就是一个标准，内容<https://tools.ietf.org/html/rfc791>

再如HTTP也是标准，（HTTP，HyperText Transfer Protocol)是[互联网](https://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91)上应用最为广泛的一种[网络协议](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE/328636)。所有的[WWW](https://baike.baidu.com/item/WWW)文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收[HTML](https://baike.baidu.com/item/HTML)页面的方法。<https://tools.ietf.org/html/rfc2616>

再如为了在全球范围唯一标识一个资源(一个网页，一个影片，一个图片等)，我们也定义了一个标准，叫URI，内容在 <https://tools.ietf.org/html/rfc3986>

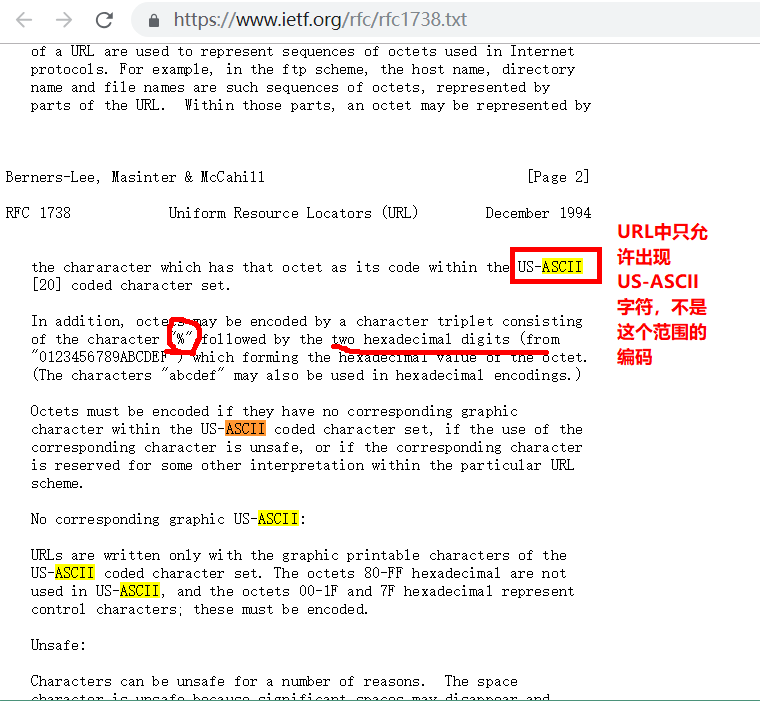
## 4.3 URL

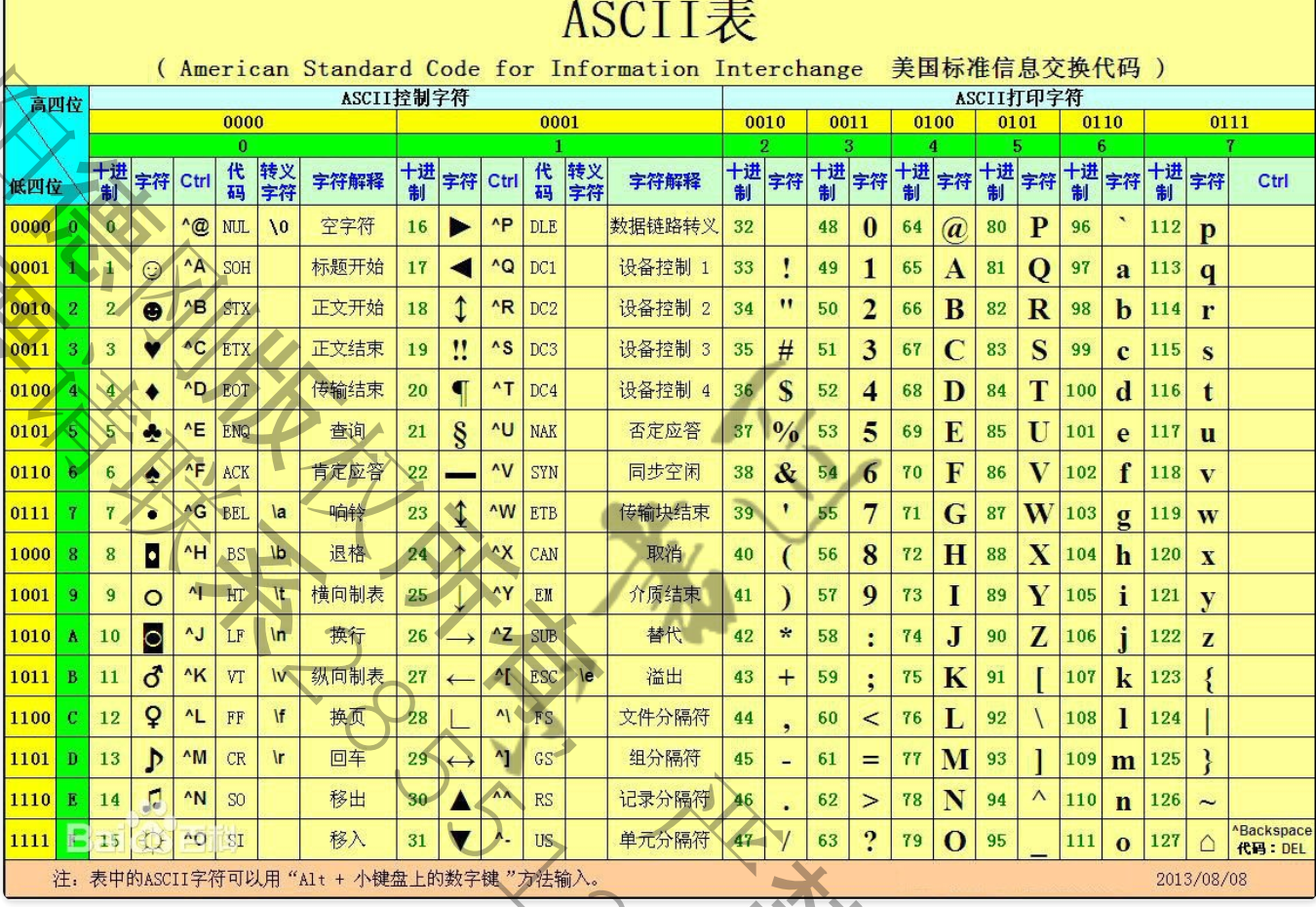
### 4.3.1 基本使用

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **import** java.net.URL;  **import** java.net.URLConnection;  **public** **class** URLTest01 {  **public** **static** **void** main(String[] args) **throws** IOException {  URL url=**new** URL("https://tools.ietf.org/html/rfc3986");  //建立联接  URLConnection conn=url.openConnection();  BufferedReader br=**new** BufferedReader(**new** InputStreamReader(conn.getInputStream()));  String line=**null**;  **while** ((line=br.readLine())!=**null**) {  System.***out***.println(line);  }    br.close();  }  } |

### 4.3.2 URL编码和解码

根据URL标准： <https://www.ietf.org/rfc/rfc1738.txt>





因而我们需要对不是这个范围的编码成这个范围的。

在Java中提供两个类来完成这个事情：

|  |
| --- |
| **import** java.io.UnsupportedEncodingException;  **import** java.net.URLDecoder;  **import** java.net.URLEncoder;  **public** **class** URLEn {  **public** **static** **void** main(String[] args) **throws** UnsupportedEncodingException {  String str="中国";  System.***out***.println(URLEncoder.*encode*(str,"utf-8"));  str="%D6%D0%B9%FA";  System.***out***.println(URLDecoder.*decode*(str,"utf-8"));  }  } |