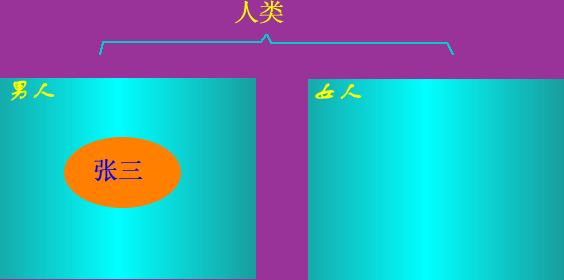
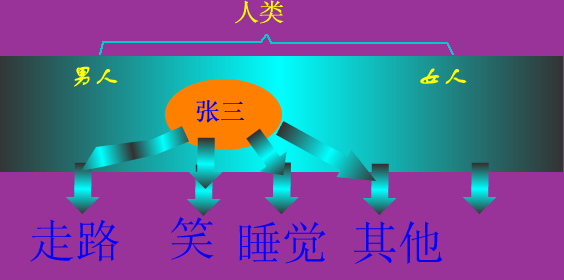
[一次性理清java 中的 类、抽象类、接口、继承、对象](http://blog.csdn.net/pisota/article/details/5332918)

分类： [JAVA SE](http://blog.csdn.net/pisota/article/category/656805) 2010-02-27 22:57

 类是广泛的概念，表示一个有共同性质的群体。如：人类。  
  
代码：  
       // 声明一个类“Human”  
  **class Human{  
   private String name;  
   public String getName(){  
     return name;  
   }  
 public void setName(String value){  
  this.name = value;  
 }  
                                  //......  
                               }**  
  
对象指的是具体的一个实实在在的东西。如、张三、李四。  
  
代码：  
  **Human zhangsan = new Human();**  
  
图：  
       
  
图中我们怎么创建张三和李四呢？？我们前面的代码好像还没涉及，恩好，看下面  
  
**Human human = new Human();**  
程序执行到这里之后，java虚拟机将会在内存中创建一个 Human 对象，并将这个对象的引用赋给 human 变量。  
  
接着：**human.setName("张三"); human.setName("李四");**好了对象（实例）创建好了，呵呵！  
  
图：  
        
  
“人”是一个基类，则“男人”是“人”的一个子类。如果“张三”是一个“男人”，也就是说“张三”是“男人”的一个对象，那么显然“张三”也是“人”这个基类的一个对象。  
   **class nanren extends  human(String args[ ]){  
           //.............  
        }**  
  
方法是类的行为，如人会笑、会坐、会走路 一样。  
  
图：  
        
          
  代码：  
**zhangsan.go();**

**zhangsan.xiao();**

**zhangsan.shuijiao();**

**zhangsan.qita();**  
  
接口只定义了一些方法，而没有实现这些方法，它规定了实现接口的类必须具备的方法（能力），它只是一个规定。  
  
我们规定：只要是人就可以走路、笑、睡觉、做其他的事，于是我们用代码实现这个规定  
**public interface you\_are\_human{  
public void go();  
public void xiao();  
public void shuijiao();  
public void qita();  
}**  
这时，张三想做人，但他只想拥有走路和笑的能力。首先，他必须声明自己是人，也就是前面对人的规定，  
然后想做什么由他自己决定：  
  
**public class zhangsan extends Human implements you\_are\_human{  
public void go(){  
// ......  
}  
public void xiao(){  
// ......  
}  
}**  
使用的时候：  
**you\_are\_human  zhangsan = new zhangsan();  
  
zhangsan.go();  
  
zhangsan.xiao();**   
  
那有的人就会问，这和前面我定义类有什么区别；  
  
这是因为：java的类的继承是从上到下的，爷爷1 》老爸1 》儿子1  爷爷2   》老爸2  》 儿子2, 而不能是：爷爷2  》 儿子1 这样就乱套了 ，是不，闹笑话；所以，类规定了不继承关系，防止乱套，使结构变得清晰。  
所以，当我们使用树型结构时，使用类。

由于，在生活中还有许多这样的情况，如，张三是中国人，那张三他也是美国人双重身份。  
再如，李四是java工程师,也是厨师。那怎么办？？  
  
我们可以这么做：规定多个身份  **public interface zhengjian{  
interface China(){  
public void China\_L(){  
//.................  
      }  
}  
interface  USA(){  
public  void USA\_L(){  
//...........  
           }**   
}  
  
**public class zhangsan extends zhengjian implements Chna , USA{  
public void China\_L(){  
// ......  
}  
public void USA\_L(){  
// ......  
}  
}**   
  
当张三去美国时，老美叫他拿个绿证来看：  
  
**zhangsan.USA\_L();**

 呵呵，再举个例子大家就清楚了：  
  
HP有“三星”和“IBM”两种，我通常而言不可能一个HP是既是“三星”又是“IBM”的吧~~~~~~~~类  
          “家庭式”和“办公式”两种，但往往我们能买到拥有两种款式相结合厂品是吧    ~~~~~~~~~~~接口  
  
中国人说起来，类有点“封建世袭制”，接口有点“违反常理乱伦”，呵呵http://www.blogjava.net/Emoticons/QQ/lol.gif开个玩笑。  
  
这样说完大家再去看些相关文章就会觉得很容易了 呵呵http://www.blogjava.net/Emoticons/QQ/icon17.gif  
  
剩下抽象类就简单了，那是抽象类（abstract）他还是类（class)，只不过是抽象的罢了。  
  
抽象类对某具体类型的部分实现------抽象类是对具体的抽象方法特征包括：方法的名字、参数的数目、参数的类型。  
  
不包括：返回类型、参数名字、和抛出的异常。  
  
[java](http://www.qqread.com/keywords/java.html)接口（抽象类）用来声明一个新的类型,Java设计师应当主要使用接口和抽象类将软件单位与内部和外部耦合(软件工程中内容)起来。  
  
在理想的情况下，一个具体类应当只实现接口和抽象类中声明的方法，而不应当给出多余的方法！  
  
抽象类仅提供一个类的部分实现。抽象类可以有实例变量、以及一个或多个构造函数。  
人这个概念够抽象了吧！  
public  abstract   class   Human{...}    
  
男人也是   
public   abstract   class  man{...}      
  
有一点必须讲清：接口不能继承抽象类，接口只能继承实实在在的类，  
  
如，我可以为某个HP厂品（类）规定一种规范（接口），但由于各个厂家推陈出新，所以，我很难对某一类型厂品规定，“家庭型hp”就很难下个结论（接口）。  
  
  
**从软件优化角度来考虑类和接口问题。类class条理清楚内聚性好，用于软件模块内部实现（汽车的内部机械）；接口interface很灵活耦合性好，用于模块外部实现（如，汽车操作板的按钮）。从而实现强内聚、松耦合的软件设计思想**