类间关系有很多种，在大的类别上可以分为两种：纵向关系、横向关系。

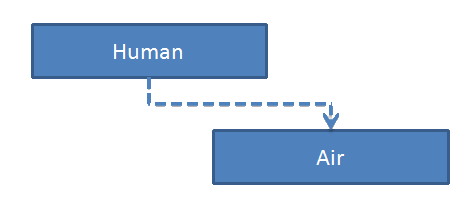
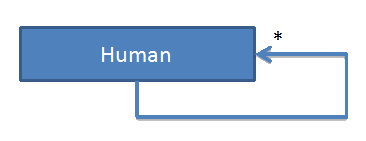
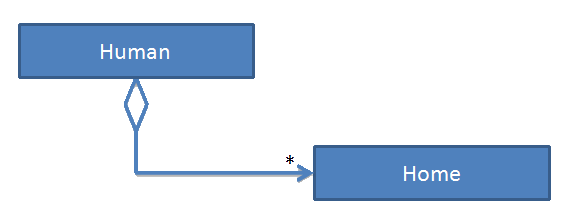
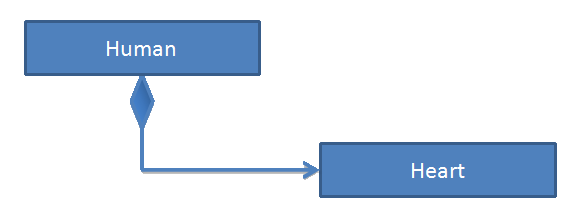
纵向关系就是继承关系，它的概念非常明确，也成为OO的三个重要特征之一，这里不过多的讨论。

横向关系较为微妙，按照UML的建议大体上可以分为四种：

1. 依赖    （Dependency）
2. 关联    （Association）
3. 聚合    （Aggregation）
4. 组合    （Composition）

它们的强弱关系是没有异议的：依赖 < 关联 < 聚合 < 组合

然而它们四个之间的差别却又不那么好拿捏，需要好好体会。

1. **依赖**：
   * UML表示法：虚线 + 箭头   
     http://images.cnblogs.com/cnblogs_com/floodpeak/1.gif
   * 关系：" ... uses a ..."
   * 此关系最为简单，也最好理解，所谓依赖就是某个对象的功能依赖于另外的某个对象，而被依赖的对象只是作为一种工具在使用，而并不持有对它的引用。
   * 典型的例子很多，比如：  
     class Human  
     {  
         public void breath()  
         {  
             Air freshAir = new Air();  
             freshAir.releasePower();  
         }  
         public static void main()  
         {  
             Human me = new Human();  
             while(true)  
             {  
                 me.breath();  
             }  
         }  
     }  
       
     class Air  
     {  
         public void releasePower()  
         {  
             //do sth.  
         }  
     }  
      
   * 释义：一个人自创生就需要不停的呼吸，而人的呼吸功能之所以能维持生命就在于吸进来的气体发挥了作用，所以说空气只不过是人类的一个工具，而人并不持有对它的引用。
2. **关联**：
   * UML表示法：实线 + 箭头   
     http://images.cnblogs.com/cnblogs_com/floodpeak/2.gif
   * 关系：" ... has a ..."
   * 所谓关联就是某个对象会长期的持有另一个对象的引用，而二者的关联往往也是相互的。关联的两个对象彼此间没有任何强制性的约束，只要二者同意，可以随时解除关系或是进行关联，它们在生命期问题上没有任何约定。被关联的对象还可以再被别的对象关联，所以关联是可以共享的。
   * 典型的例子很多，比如：  
     class Human  
     {  
         ArrayList friends = new ArrayList();  
         public void makeFriend(Human human)  
         {  
             friends.add(human);  
         }  
         public static void main()  
         {  
             Human me = new Human();  
             while(true)  
             {  
                 me.makeFriend(mySchool.getStudent());  
             }  
         }  
     }   
     
   * 释义：人从生至死都在不断的交朋友，然而没有理由认为朋友的生死与我的生死有必然的联系，故他们的生命期没有关联，我的朋友又可以是别人的朋友，所以朋友可以共享。
3. **聚合**：
   * UML表示法：空心菱形 + 实线 + 箭头   
     http://images.cnblogs.com/cnblogs_com/floodpeak/3.gif
   * 关系：" ... owns a ..."
   * 聚合是强版本的关联。它暗含着一种所属关系以及生命期关系。被聚合的对象还可以再被别的对象关联，所以被聚合对象是可以共享的。虽然是共享的，聚合代表的是一种更亲密的关系。
   * 典型的例子很多，比如：  
     class Human  
     {  
         Home myHome;  
         public void goHome()  
         {  
             //在回家的路上  
             myHome.openDoor();  
             //看电视  
         }  
         public static void main()  
         {  
             Human me = new Human();  
             while(true)  
             {  
                 //上学  
                 //吃饭  
                 me.goHome();  
             }  
         }  
     }  
      
   * 释义：我的家和我之间具有着一种强烈的所属关系，我的家是可以分享的，而这里的分享又可以有两种。其一是聚合间的分享，这正如你和你媳妇儿都对这个家有着同样的强烈关联；其二是聚合与关联的分享，如果你的朋友来家里吃个便饭，估计你不会给他配一把钥匙。
4. **组合**：
   * UML表示法：实心菱形 + 实线 + 箭头   
     http://images.cnblogs.com/cnblogs_com/floodpeak/4.gif
   * 关系：" ... is a part of  ..."
   * 组合是关系当中的最强版本，它直接要求包含对象对被包含对象的拥有以及包含对象与被包含对象生命期的关系。被包含的对象还可以再被别的对象关联，所以被包含对象是可以共享的，然而绝不存在两个包含对象对同一个被包含对象的共享。
   * 典型的例子很多，比如：  
     class Human  
     {  
         Heart myHeart = new Heart();  
         public static void main()  
         {  
             Human me = new Human();  
             while(true)  
             {  
                 myHeart.beat();  
             }  
         }  
     }  
     
   * 释义：组合关系就是整体与部分的关系，部分属于整体，整体不存在，部分一定不存在，然而部分不存在整体是可以存在的，说的更明确一些就是部分必须创生于整体创生之后，而销毁于整体销毁之前。部分在这个生命期内可以被其它对象关联甚至聚合，但有一点必须注意，一旦部分所属于的整体销毁了，那么与之关联的对象中的引用就会成为空引用，这一点可以利用程序来保障。心脏的生命期与人的生命期是一致的，如果换个部分就不那么一定，比如阑尾，很多人在创生后的某个时间对其厌倦便提前销毁了它，可它和人类的关系不可辩驳的属于组合。  
     在UML中存在一种特例，就是允许被包含对象在包含对象销毁前转移给新的对象，这虽然不自然，但它给需要心脏移植的患者带来了福音。