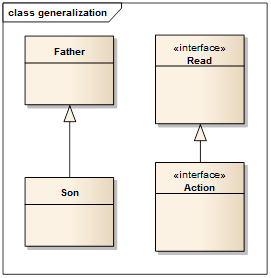
UML类图中常见关系详解（泛化、实现、依赖、关联、组合、聚合）

UML中类与类，已经类与接口，接口与接口的关系有：泛化（generalization），关联（association），依赖（dependency），实现（realization）这几种。

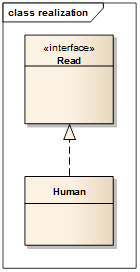
## **泛化（generalization）**(实线表示强)

关系时指一个类（子类、子接口）继承另外一个类（称为父类、父接口）的功能，并可以增加它自己新功能的能力，继承是类与类或者接口与接口最常见的关系，在Java中通过关键字extends来表示。



## **实现（realization）**(虚线表示弱)

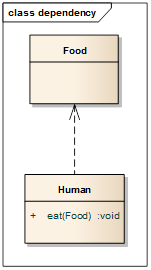
是指一个class实现interface接口（一个或者多个），表示类具备了某种能力，实现是类与接口中最常见的关系，在Java中通过implements关键字来表示。



## 依赖-弱关系（dependency）(虚线表示弱)

关系也是表示类与类之间的连接，表示一个类依赖于另外一个类的定义，依赖关系时是单向的。简单理解就是类A使用到了类B，这种依赖具有偶然性、临时性，是非常弱的关系。但是类B的变化会影响到类A。举个例子，如某人要过河，则人与船的关系就是依赖，人过河之后，与船的关系就解除了，因此是一种弱的连接。在代码层面，为类B作为参数被类A在某个方法中使用。

在java中，依赖表现为：局部变量，方法中的参数和对静态方法的调用。



## 关联-强关系（association）(实线表示强)

关系表示类与类之间的连接，它使得一个类知道另外一个类的属性和方法。

关联可以使用单箭头表示单向关联，使用双箭头或者不适用箭头表示双向关联，不建议使用双向关联，关联有两个端点，每个端点可以有一个基数，表示这个关联的类可以有几个实例。

0..1 表示可以有0个或者1个实例

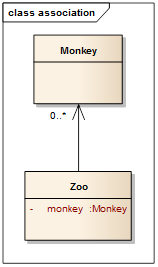
0..\* 表示对实例的数目没有限制

1     表示只能有一个实例

1..\* 表示至少有一个实例

关联关系体现的是两个类，或者类与接口之间的强依赖关系，这种关系很强烈，比依赖更强，不是偶然性的，也不是临时性的，而是一种长期性，相对平等的关系，表现在代码层面，为被关联的类B以类属性的形式出现在类A中，也可能是关联类A引用了被关联类B的全局变量。

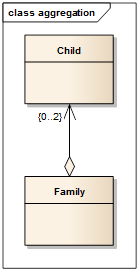
在Java中，关联关系是使用实例变量来实现的



## 聚合-弱关系（aggregation）(空心菱形)

是关联关系的特例，是强的关联关系，聚合是整个与个体的关系，即has-a关系，此时整体和部分是可以分离的，他们具有各自的生命周期，部分可以属于多个对象，也可以被多个对象共享；比如计算机和CPU，公司与员工的关系；在代码层面聚合与关联是一致的，只能从语义上来区分。

聚合关系也是使用实例变量来实现的，在java语法上区分不出关联和聚合，关联关系中类出于一个层次，而聚合则明显的在两个不同的层次。

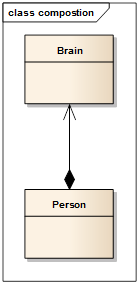


## 组合-强关系（compostion）(实心菱形)

也是关联关系的一种特例，体现的是一种contain-a关系，比聚合更强，是一种强聚合关系。它同样体现整体与部分的关系，但此时整体与部分是不可分的，整体生命周期的结束也意味着部分生命周期的结束，反之亦然。如大脑和人类。

体现在代码层面与关联时一致的，只能从语义来区分。

组合与聚合几乎完全相同，唯一区别就是对于组合，“部分”不同脱离“整体”单独存在，其生命周期应该是一致的。



总结：

主要是关联关系的细化需要注意强弱，由若到强分别是

**组合>聚合>关联>依赖**

参考：

<http://justsee.iteye.com/blog/808799>

UML的构造快包含3种：

(1) 事物（4种）：结构事物，行为事物，分组事物，注释事物

(2) 关系（4种）：泛化关系，实现关系，依赖关系，关联关系

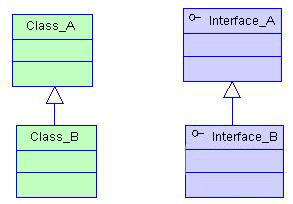
(3) 图（10种）：用例图，类图，对象图，包图，组件图，部署图，状态图，活动图，序列图，协作图

事物是对模型中最具代表性的成分的抽象；关系把事物结合在一起；图聚集了相关的事物。

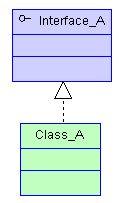
 (2) 关系（4种）

UML 中类与类, 类与接口, 接口与接口这间的关系有: 泛化(generalization) 关系, 关联(association)关系( 关联, 聚合, 合成), 依赖(dependency)关系，实现(realization)关系.

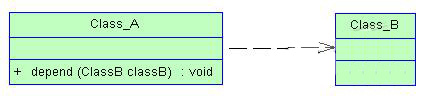
泛化(generalization)关系是一个类（称为子类、子接口）继承另外的一个类（称为父类、父接口）的功能，并可以增加它自己的新功能的能力，继承是类与类或者接口与接口之间最常见的关系；在Java中此类关系通过关键字extends明确标识，在设计时一般没有争议性。



实现(realization)关系指的是一个class类实现interface接口（可以是多个）的功能；实现是类与接口之间最常见的关系；在Java中此类关系通过关键字implements明确标识，在设计时一般没有争议性；

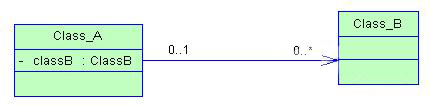


依赖(dependency)关系: 也是类与类之间的连接. 表示一个类依赖于另一个类的定义. 依赖关系总是单向的 。可以简单的理解，就是一个类A使用到了另一个类B，而这种使用关系是具有偶然性的、、临时性的、非常弱的，但是B类的变化会影响到A；比如某人要过河，需要借用一条船，此时人与船之间的关系就是依赖；表现在代码层面，为类B作为参数被类A在某个method方法中使用。  
在java 中. 依赖关系体现为: 局部变量, 方法中的参数, 和对静态方法的调用.



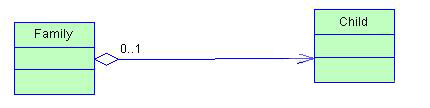
关联(association)关系: 表示类与类之间的联接, 它使一个类知道另一个类的属性和方法.   
关联可以使用单箭头表示单向关联, 使用双箭头或不使用箭头表示双向关联, 不建议使用双向关联. 关联有两个端点, 在每个端点可以有一个基数, 表示这个关联的类可以有几个实例.   
常见的基数及含义:   
0..1:0 或1 个实例.   
0..\*: 对实例的数目没有限制.   
1: 只能有一个实例.   
1..\*: 至少有一个实例.

他体现的是两个类、或者类与接口之间语义级别的一种强依赖关系，比如我和我的朋友；这种关系比依赖更强、不存在依赖关系的偶然性、关系也不是临时性的，一般是长期性的，而且双方的关系一般是平等的,表现在代码层面，为被关联类B以类属性的形式出现在关联类A中，也可能是关联类A引用了一个类型为被关联类B的全局变量；在java 语言中关联关系是使用实例变量实现的.

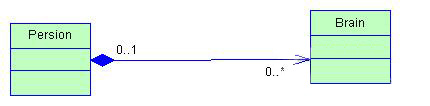
  
   
聚合(aggregation)关系: 关联关系的一种特例, 是强的关联关系. 聚合是整体和个体之间的关系,即has-a的关系，此时整体与部分之间是可分离的，他们可以具有各自的生命周期，部分可以属于多个整体对象，也可以为多个整体对象共享；比如计算机与CPU、公司与员工的关系等；表现在代码层面，和关联关系是一致的，只能从语义级别来区分；

聚合关系也是使用实例变量实现的. 从java 语法上是分不出关联和聚合的.

关联关系中两个类是处于相同的层次, 而聚合关系中两不类是处于不平等的层次, 一个表示整体, 一个表示部分.

  
   
组合(合成)关系(composition): 也是关联关系的一种特例，他体现的是一种contains-a的关系，这种关系比聚合更强，也称为强聚合；他同样体现整体与部分间的关系，但此时整体与部分是不可分的，整体的生命周期结束也就意味着部分的生命周期结束；比如你和你的大脑；合成关系不能共享. 。表现在代码层面，和关联关系是一致的，只能从语义级别来区分。

组合跟聚合几乎相同，唯一的区别就是“部分”不能脱离“整体”单独存在，就是说， “部分”的生命期不能比“整体”还要长。



总结：

对于继承、实现这两种关系没多少疑问，他们体现的是一种类与类、或者类与接口间的纵向关系；其他的四者关系则体现的是类与类、或者类与接口间的引用、横向关系，是比较难区分的，有很多事物间的关系要想准备定位是很难的，前面也提到，这几种关系都是语义级别的，所以从代码层面并不能完全区分各种关系；但总的来说，后几种关系所表现的强弱程度依次为：组合>聚合>关联>依赖。