**UML类图关系**

在UML类图中，常见的有以下几种关系: 泛化（Generalization）,  实现（Realization），关联（Association)，聚合（Aggregation），组合(Composition)，依赖(Dependency)

**1. 泛化（Generalization）**

　　【泛化关系】：是一种继承关系，表示一般与特殊的关系，它指定了子类如何特化父类的所有特征和行为。例如：老虎是动物的一种，即有老虎的特性也有动物的共性。

　　【箭头指向】：带三角箭头的实线，箭头指向父类



**2. 实现（Realization）**

　　【实现关系】：是一种类与接口的关系，表示类是接口所有特征和行为的实现.

　　【箭头指向】：带三角箭头的虚线，箭头指向接口

[](http://www.jfox.info/url.php?_src=http://www.jfox.info/uml-lei-tu-guan-xi-fan-hua-ji-cheng-shi-xian-yi-lai-guan-lian-ju-he-zu-he&isencode=1&content=dGltZT0xNDQxMDk2NzAyJnVybD1odHRwJTNBJTJGJTJGd3d3Lmpmb3guaW5mbyUyRndwLWNvbnRlbnQlMkZ1cGxvYWRzJTJGMjAxMyUyRjEwJTJGbF9lZjczNWEyNi04YTJiLTRiN2YtYjg5ZC02NDA0ODM5OTg2ZjIuZ2lm)

**3. 关联（Association)**

　　【关联关系】：是一种拥有的关系，它使一个类知道另一个类的属性和方法；如：老师与学生，丈夫与妻子关联可以是双向的，也可以是单向的。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头。

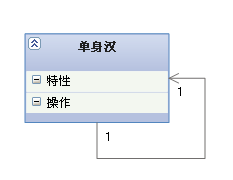
　　【代码体现】：成员变量

　　【箭头及指向】：带普通箭头的实心线，指向被拥有者

[](http://www.jfox.info/url.php?_src=http://www.jfox.info/uml-lei-tu-guan-xi-fan-hua-ji-cheng-shi-xian-yi-lai-guan-lian-ju-he-zu-he&isencode=1&content=dGltZT0xNDQxMDk2NzAyJnVybD1odHRwJTNBJTJGJTJGd3d3Lmpmb3guaW5mbyUyRndwLWNvbnRlbnQlMkZ1cGxvYWRzJTJGMjAxMyUyRjEwJTJGbF8wNmM1MzVkOC1lMTExLTQ0MDItODgzYi1lZWVlODJhNWJjNWEuZ2lm)

　　上图中，老师与学生是双向关联，老师有多名学生，学生也可能有多名老师。但学生与某课程间的关系为单向关联，一名学生可能要上多门课程，课程是个抽象的东西他不拥有学生。

　　下图为自身关联：

[](http://www.jfox.info/url.php?_src=http://www.jfox.info/uml-lei-tu-guan-xi-fan-hua-ji-cheng-shi-xian-yi-lai-guan-lian-ju-he-zu-he&isencode=1&content=dGltZT0xNDQxMDk2NzAyJnVybD1odHRwJTNBJTJGJTJGd3d3Lmpmb3guaW5mbyUyRndwLWNvbnRlbnQlMkZ1cGxvYWRzJTJGMjAxMyUyRjEwJTJGbF84ODI4NDJiYy0yZjkyLTQ3OTEtYjg3YS02MjMwZWY1MmQ2NTAuZ2lm)

**4. 聚合（Aggregation）**

　　【聚合关系】：是整体与部分的关系，且部分可以离开整体而单独存在。如车和轮胎是整体和部分的关系，轮胎离开车仍然可以存在。

　　聚合关系是关联关系的一种，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系。

　　【代码体现】：成员变量

　　【箭头及指向】：带空心菱形的实心线，菱形指向整体

[](http://www.jfox.info/url.php?_src=http://www.jfox.info/uml-lei-tu-guan-xi-fan-hua-ji-cheng-shi-xian-yi-lai-guan-lian-ju-he-zu-he&isencode=1&content=dGltZT0xNDQxMDk2NzAyJnVybD1odHRwJTNBJTJGJTJGd3d3Lmpmb3guaW5mbyUyRndwLWNvbnRlbnQlMkZ1cGxvYWRzJTJGMjAxMyUyRjEwJTJGbF9kZTNmN2UxZi0wY2MxLTQzOGUtYjllYS02N2Q1OTM1ZDIyMTQuZ2lm)

**5. 组合(Composition)**

　　【组合关系】：是整体与部分的关系，但部分不能离开整体而单独存在。如公司和部门是整体和部分的关系，没有公司就不存在部门。

       组合关系是关联关系的一种，是比聚合关系还要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

【代码体现】：成员变量

【箭头及指向】：带实心菱形的实线，菱形指向整体

[](http://www.jfox.info/url.php?_src=http://www.jfox.info/uml-lei-tu-guan-xi-fan-hua-ji-cheng-shi-xian-yi-lai-guan-lian-ju-he-zu-he&isencode=1&content=dGltZT0xNDQxMDk2NzAyJnVybD1odHRwJTNBJTJGJTJGd3d3Lmpmb3guaW5mbyUyRndwLWNvbnRlbnQlMkZ1cGxvYWRzJTJGMjAxMyUyRjEwJTJGbF82NDNjNGRhZC03ODE4LTQyMjQtYTcxMy03MDlmMjkzNjliY2MuZ2lm)

**6. 依赖(Dependency)**

　　【依赖关系】：是一种使用的关系，即一个类的实现需要另一个类的协助，所以要尽量不使用双向的互相依赖.

　　【代码表现】：局部变量、方法的参数或者对静态方法的调用

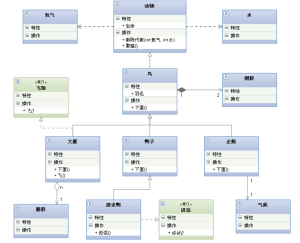
　　【箭头及指向】：带箭头的虚线，指向被使用者

[](http://www.jfox.info/url.php?_src=http://www.jfox.info/uml-lei-tu-guan-xi-fan-hua-ji-cheng-shi-xian-yi-lai-guan-lian-ju-he-zu-he&isencode=1&content=dGltZT0xNDQxMDk2NzAyJnVybD1odHRwJTNBJTJGJTJGd3d3Lmpmb3guaW5mbyUyRndwLWNvbnRlbnQlMkZ1cGxvYWRzJTJGMjAxMyUyRjEwJTJGbF9lNmYxMTJjYS05M2QxLTQzMzgtODhhMy04OThjNjZmMGUyMDMuZ2lm)

　　各种关系的强弱顺序：

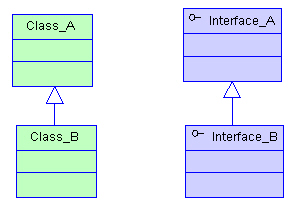
**泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖**

　　下面这张UML图，比较形象地展示了各种类图关系：

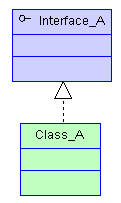
[](http://www.jfox.info/url.php?_src=http://www.jfox.info/uml-lei-tu-guan-xi-fan-hua-ji-cheng-shi-xian-yi-lai-guan-lian-ju-he-zu-he&isencode=1&content=dGltZT0xNDQxMDk2NzAyJnVybD1odHRwJTNBJTJGJTJGd3d3Lmpmb3guaW5mbyUyRndwLWNvbnRlbnQlMkZ1cGxvYWRzJTJGMjAxMyUyRjEwJTJGbF9lNDk1ODlmMS0xZGRhLTQ2MjItODVkMC04ODYzOGY3MDg3MWYucG5n)

这是一堂关于UML基础知识的补习课；现在我们做项目时间都太紧了，基本上都没有做过真正的class级别的详细设计，更别提使用UML来实现规范建模了；本篇主要就以前自己一直感觉很迷糊的几种class之间的关系进行整理，让我们在真正用UML进行比如类图设计时能够更加清晰明了；以下就分别介绍这几种关系：

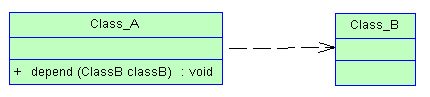
### 继承

指的是一个类（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；  


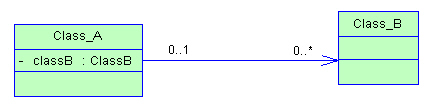
### 实现

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；  


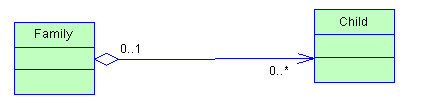
### 依赖

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；  


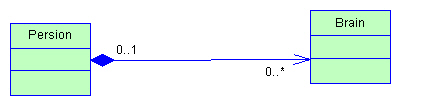
### 关联

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、关联可以是单向、双向的；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；  


### 聚合

聚合是关联关系的一种特例，他体现的是整体与部分、拥有的关系，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；  


### 组合

组合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；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；  


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；