[UML类图画法及其之间的几种关系](http://blog.csdn.net/wanmeirongyan100/article/details/51601570)

**类图画法**

类之间的几种关系：泛化（Generalization）、实现（Realization）、关联（Association）（又分一般关联、聚合（Aggregation）、组合（Composition））、依赖（Dependency）

**一、类图画法**

**1、 类图的概念**

A、显示出**类、接口**以及它们之间的**静态结构和关系**

B、用于描述系统的结构化设计

**2、 类图的元素**

类、接口、协作、关系，我们只简单介绍一下这四种元素。

同其他的图一样，类图也可以包含**注解和限制**。

类图中也可以包含**包和子系统**，这两者用来**将元素分组**。

有时候你也可以将**类的实例**放到类图中。

**3、 类**

A、 类是对一组具有相同属性、操作、关系和语义的对象的抽象，它是面向对象系统组织结构的核心，包括名称部分（Name）、属性部分（Attribute）和操作部分（Operation），见下图。



B、 类属性的语法为：

[可见性] 属性名 [:类型] [=初始值] [{属性字符串}]

可见性：公有（Public）“+”、私有（Private）“-”、受保护（Protected）“#”

类操作的语法为：

[可见性] 操作名 [(参数表)] [:返回类型] [{属性字符串}]

可见性：公有（Public）“+”、私有（Private）“-”、受保护（Protected）“#”、包内公有（Package）“~”

参数表：

定义方式：“名称：类型”；若存在多个参数，将各个参数用逗号隔开；参数可以具有默认值；

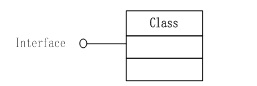
属性字符串：

在操作的定义中加入一些除了预定义元素之外的信息。

**4、 接口**

在没有给出对象的实现和状态的情况下对对象行为的描述。

一个类可以实现一个或多个接口。

使用两层矩形框表示，与类图的区别主要是顶端有<>显示：   
   
也可以用一个空心圆表示：   


**5、 协作**

协作是指一些类、接口和其他的元素一起工作提供一些合作的行为，这些行为不是简单地将元素加能得到的。例如：当你为一个分布式的系统中的事务处理过程建模型时，你不可能只通过一个类来明白事务是怎样进行的，事实上这个过程的执行涉及到一系列的类的协同工作。使用类图来可视化这些类和他们的关系。

**6、 关系**

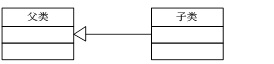
这篇文章的重点，详见第二部分。

**二、类之间的几种关系**

**1、 泛化（Generalization）**

A、 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，描述了一种“is a kind of” 的关系。例如：老虎是动物的一种，即有老虎的特性也有动物的共性。

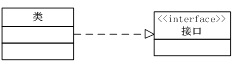
B、 用带**空心箭头的实线**表示，箭头指向父类，如下图：



**2、 实现（Realization）**

A、 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。

B、 用带**空心箭头的虚线**表示，箭头指向接口，如下图：



**3、 关联（Association）**

**A、 一般关联**

a、 关联关系是**类与类之间的联结**，它使一个类知道另一个类的属性和方法，指明了事物的对象之间的联系，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，还有自身关联。

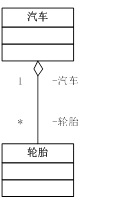
b、 用**带普通箭头的实心线**表示。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头，如下图：



**B、 聚合（Aggregation）**

a、 它是**整体与部分（整体 has a 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。

b、 用带**空心菱形的实线**表示，菱形指向整体，如下图：



**C、 组合（Composition）**

a、 它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。

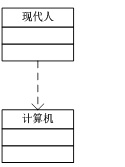
b、 用带**实心菱形的实线**表示，菱形指向整体，如下图：



**4、 依赖（Dependency）**

A、 元素A的变化会影响元素B，那么B和A的关系是依赖关系，B依赖A。要避免双向依赖，一般来说，不应该存在双向依赖。关联、实现、泛化都是依赖关系。

B、 用带箭头的虚线表示，箭头指向被依赖元素。

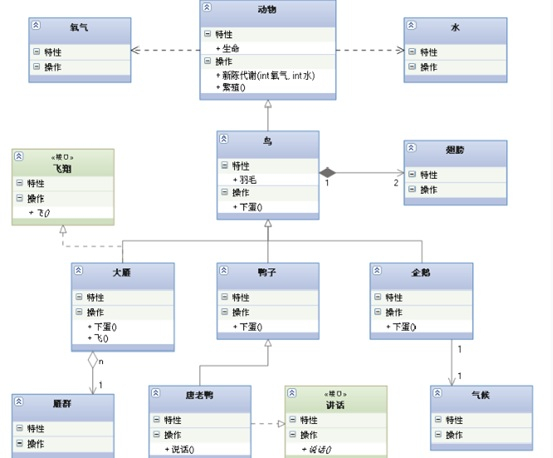


**5、 总结**

各种关系的强弱顺序如下：

**泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖**

下面这张UML图，比较形象地展现了各种类图关系：



转载：<http://www.360doc.com/content/13/0304/13/10504424_269223190.shtml>