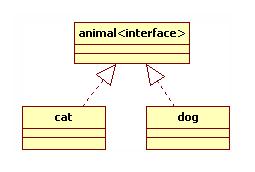
类之间的关系种类：Realization(实现)， Generalization(泛化)，Dependency(依赖)、Association(关联)、Aggregation(聚合)、Composition(合成或组合)。 其中，Aggregation(聚合)、Composition(合成)属于Association(关联)，是特殊的Association关联关系。

实现(Realization)：

    指的是一个类实现接口（可以是多个）的功能；实现是类与接口之间最常见的关系；C 中没有直接的接口而是通过在类中定义纯虚函数来实现的。

    UML图中实现使用一条带有空心三角箭头的虚线指向接口。如下

[](http://blog.chinaunix.net/attachment/201108/19/26111972_1313715398hYp7.jpg)

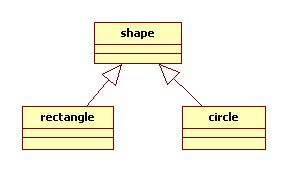
代码示例：

1. Class animal
2. {
3. public:
4. Roar() =0;
5. }
6. Class Cat:public animal
7. {
8. public:
9. Roar(){cout<<”cat”;}
10. }
11. Class dog:public animal
12. {
13. public:
14. Roar(){cout<<”cat”;}
15. }

泛化(Generalization)

   表现为继承或实现关系(is a)。具体形式为类与类之间的继承 关系，接口与接口之间的继承关系，类对接口的实现关系

UML图中实现使用一条带有空心三角箭头的实线指向基类。如下

[](http://blog.chinaunix.net/attachment/201108/19/26111972_1313715617kikS.jpg)

代码示例:

1. Class shape
2. {
3. public:
4. Display(){cout<<”shape”;}
5. }
6. Class rectangle: public shape
7. {
8. public:
9. Display(){cout<<” rectangle”;}
10. }
11. Class circle: public shape
12. {
13. public:
14. Display(){cout<<” circle”;}
15. }

依赖 (Dependency)

表现为函数中的参数(use a)。

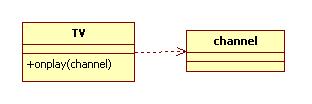
是类与类之间的连接，表示一个类依赖于另一个类的定义，其中一个类的变化将影响另外一个类。例如如果A依赖于B，则B体现为局部变量，方法的参数、或静态方法的调用。如电视(TV)依赖于频道(channel)常见的依赖关系如下：

   （1）类B以参数的形式传入类A的方法。

   （2）类B以局部变量的形式存在于类A的方法中。

   （3）类A调用类B的静态方法。

UML图中实现使用一条带有箭头的虚线指向被依赖的类。如下

[](http://blog.chinaunix.net/attachment/201108/19/26111972_1313715852Lv4j.jpg)

代码示例

1. class chaanle
2. {
3. public:
4. play(){cout<<"play";}
5. }
6. class TV
7. {
8. public:
9. onplay(channel &chan)
10. {
11. chan.play();
12. }
13. }

关联（Association）

表现为变量(has a )。

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。例如如果A依赖于B，则B体现为A的全局变量。如person类和company类

关联关系有双向关联和单向关联。

双向关联：两个类都知道另一个类的公共属性和操作。

单向关联：只有一个类知道另外一个类的公共属性和操作。

大多数关联应该是单向的，单向关系更容易建立和维护，有助于寻找可服用的类。

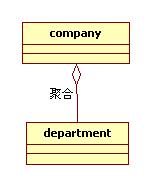
UML图中实现使用一条实线连接相同或不同类

[](http://blog.chinaunix.net/attachment/201108/19/26111972_13137162445n3z.jpg)

聚合(Aggregat ion)

是关联关系的一种，是强的关联关系。聚合关系是整体和个体的关系。普通关联关系的两个类处于同一层次上，而聚合关系的两个类处于不同的层次，一个是整体，一个是部分。同时，是一种弱的“拥有”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；

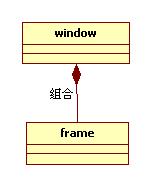
UML图中实现使用一条带有虚心菱形的线来表示

[](http://blog.chinaunix.net/attachment/201108/19/26111972_1313716315x0YS.jpg)

组合(Composition)

是关联关系的一种，是比聚合关系强的关联关系。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。Composition(组合关系)是一种强的“拥有”关系，体现了严格的部分和整体的关系，部分和整体的生命周期一致。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，window窗口和frame，在窗口中创建一个frame时必须把它附加到窗口上，当窗口消失时frame也就消失了；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；

UML图中实现使用一条带有实心菱形的线来表示

[](http://blog.chinaunix.net/attachment/201108/19/26111972_13137163744rZE.jpg)

几种关系所表现的强弱程度依次为：**组合>聚合>关联>依赖**

聚合和组合的区别：

    聚合关系是“has-a”关系，组合关系是“contains-a”关系；聚合关系表示整体与部分的关系比较弱，而组合比较强；聚合关系中代表部分事物的对象与代表聚合事物的对象的生存期无关，一旦删除了聚合对象不一定就删除了代表部分事物的对象。组合中一旦删除了组合对象，同时也就删除了代表部分事物的对象。我们用浅显的例子来说明聚合和组合的区别。“国破家亡”，国灭了，家自然也没有了，“国”和“家”显然也是组合关系。而相反的，计算机和它的外设之间就是聚合关系，因为它们之间的关系相对松散，计算机没了，外设还可以独立存在，还可以接在别的计算机上。在聚合关系中，部分可以独立于聚合而存在，部分的所有权也可以由几个聚合来共享，比如打印机就可以在办公室内被广大同事共用

   聚合和组合的区别则在语义和实现上都有差别，组合的两个对象之间其生命期有很大的关联，被组合的对象是在组合对象创建的同时或者创建之后创建，在组合对象销毁之前销毁。一般来说被组合对象不能脱离组合对象独立存在，而且也只能属于一个组合对象，例如一个文档的版本，必须依赖于文档的存在，也只能属于一个文档。聚合则不一样，被聚合的对象可以属于多个聚合对象，例如一个员工可能可以属于多个公司

关联和聚合的区别:

    关联和聚合的区别主要在语义上，关联的两个对象之间一般是平等的，例如你是我的朋友，聚合则一般不是平等的，例如一个公司包含了很多员工，其实现上是差不多的。