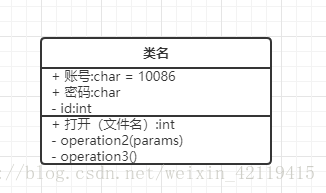
**UML类图详解**



1）类属性语法：[可见性] 属性名 [：类型] [= 初值]

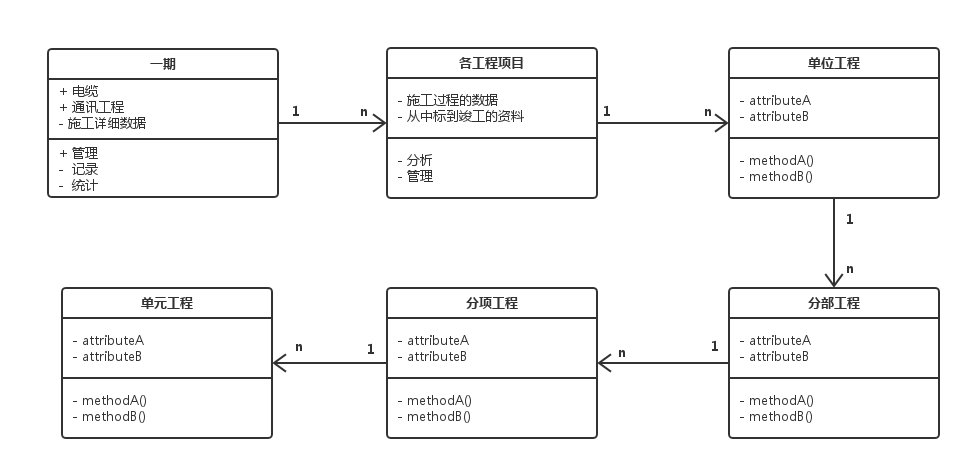
       可见性：

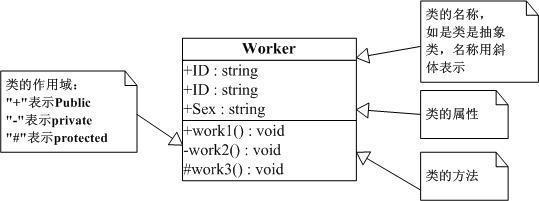
            公有（+）：外部对象访问；

            私有（-）：本类对象使用；

            保护（#）：本类对象和子类对象访问；

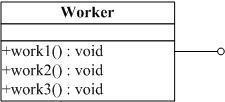
       类操作语法：[可见性] 操作名 [（参数列表）] [：返回类型]





**2.2 接口（Interface）**

　　接口是一种特殊的类，具有类的结构但不可被实例化，只可以被实现（继承）。在UML中，接口使用一个带有名称的小圆圈来进行表示。



**2.3、类图中关系（relation）**

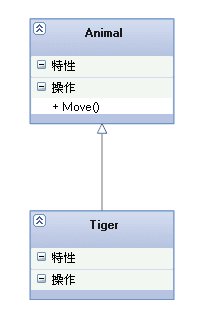
在UML类图中，常见的有以下6种关系: 泛化（Generalization）, 实现（Realization），关联（Association)，聚合（Aggregation），组合(Composition)，依赖(Dependency)。

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。

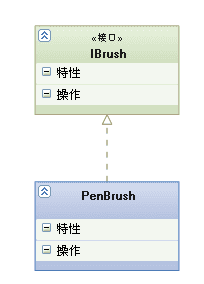
【箭头指向】：带三角箭头的实线，箭头指向父类



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



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

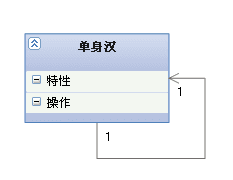
【箭头及指向】：带普通箭头的实心线，指向被拥有者



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:



4. 聚合（Aggregation）

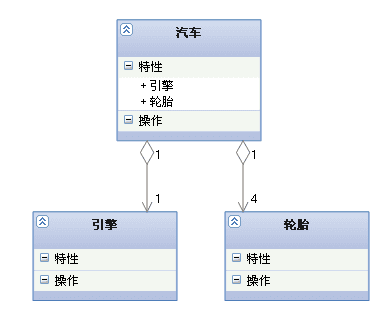
【聚合关系】：是整体与部分的关系，且部分可以离开整体而单独存在。

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



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。

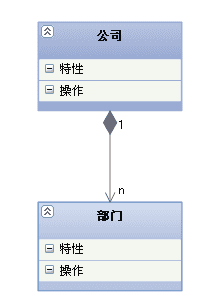
如公司和部门是整体和部分的关系，没有公司就不存在部门。

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



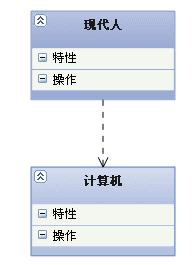
6. 依赖(Dependency)

【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

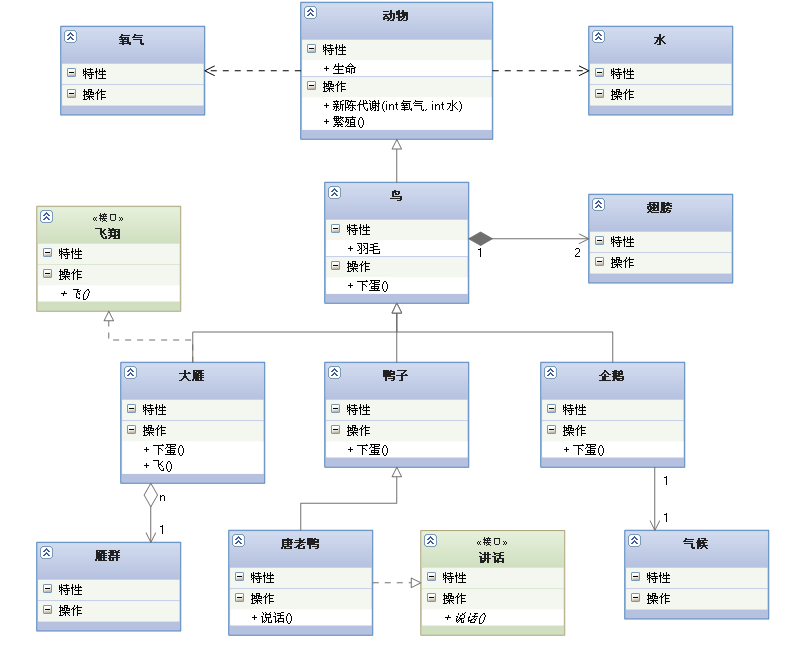
【箭头及指向】：带箭头的虚线，指向被使用者

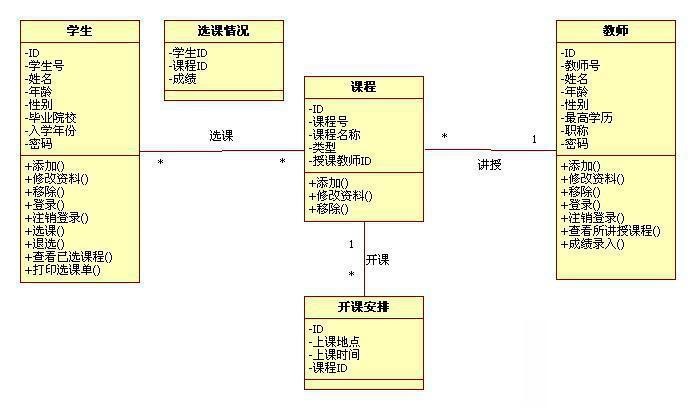


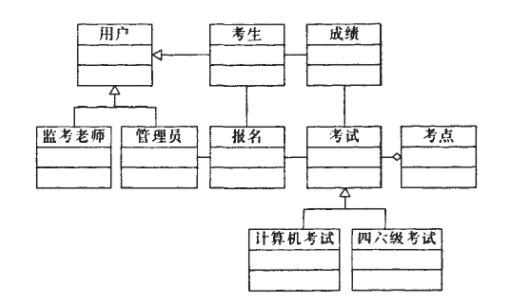
各种关系的强弱顺序：

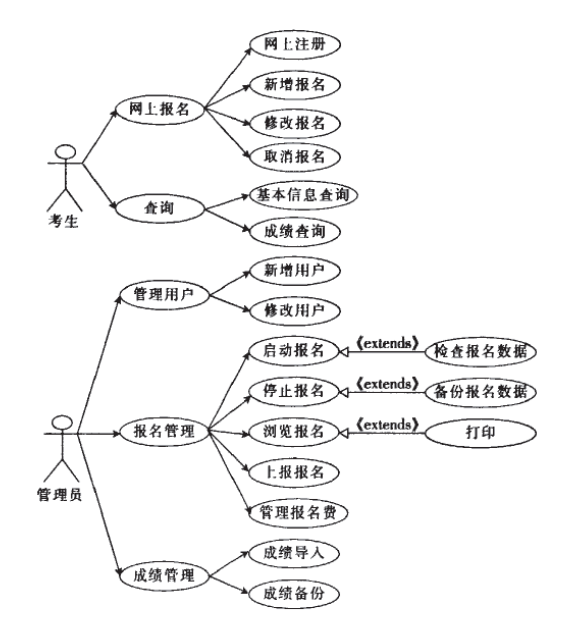
**泛化 = 实现** > **组合** > **聚合** > **关联** > **依赖**

下面这张UML图，比较形象地展示了各种类图关系：

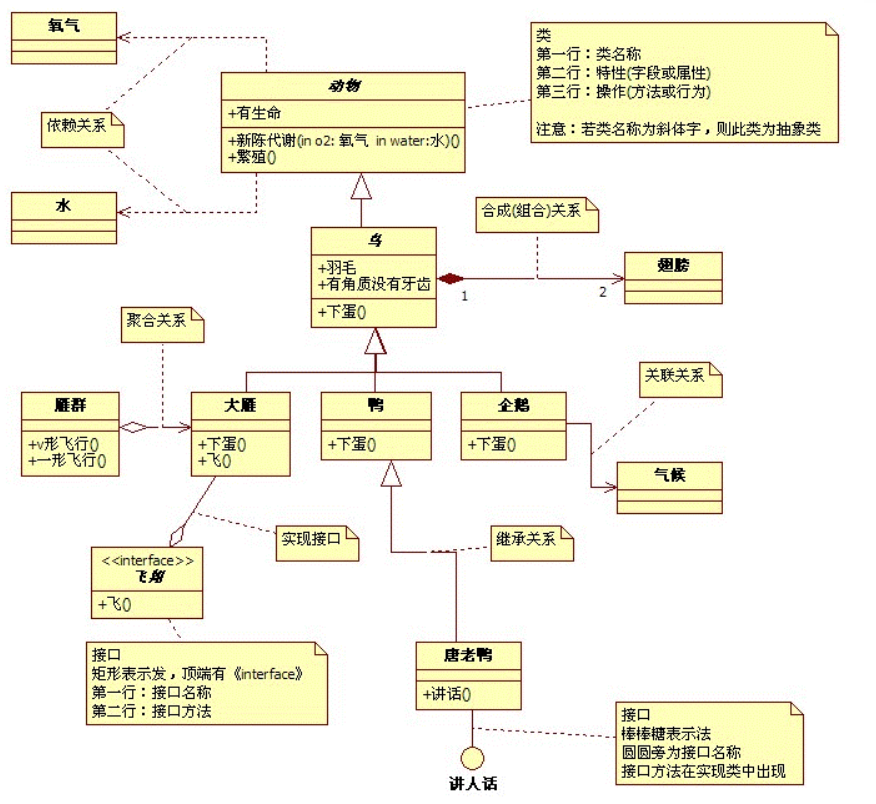








下面是类图的实例（好像大话设计中有）：



**UML中类图实例**

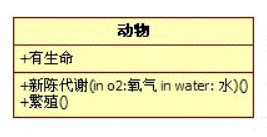
接口：空心圆+直线（唐老鸭类实现了‘讲人话’）；  
依赖：虚线+箭头（动物和空气的关系）；  
关联：实线+箭头（企鹅需要知道气候才迁移）；  
聚合：空心四边形+实线+箭头（雁群和大雁的关系）；  
合成/组合：实心四边形+实线+箭头（鸟和翅膀的关系）；  
泛化/继承：空心三角形+实线（动物和鸟的继承关系）；  
实现：空心三角形+虚线（实现大雁飞翔的接口）；

**UML类图**

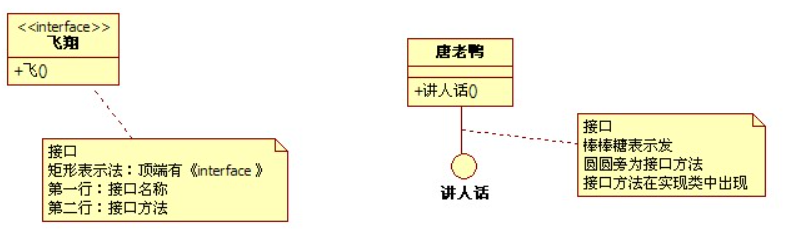
解释UML类图：

1.      首先看“动物”矩形框，它代表一个类。该类图分为三层，第一层显示类的名称，如果是抽象类就要用斜体显示。第二层是类的特性，通常就是字段和属性。第三层是类的操作，通常是方法和行为。

   注意前面的符号，‘+’表示public, ‘—’ 表示private, ‘#’表示protected.

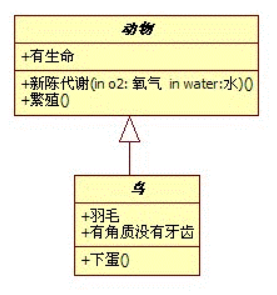


2.      “飞翔”矩形框表示一个接口图，它与类图的区别主要是顶端有《interface》显示，第一行是接口名称，第二行是接口方法。接口还有另一种表示方法，俗称棒棒糖表示法，就是唐老鸭类实现了“讲人话”的接口。

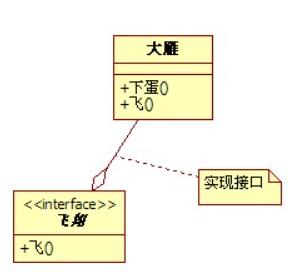


interfaceIFly                             interfaceIlanguage                                
{                                             {  
   voidFly();                                   void Speak();  
}                                            }

3.      动物，鸟，鸭，唐老鸭他们之间都是继承的关系，继承关系用空心三角形+实现来表示。

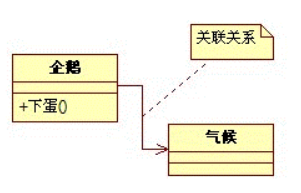


4.“大雁”实现了“飞翔”接口。实现接口用空心三角形+虚线来表示。（注：下面的图中应为空心三角形）



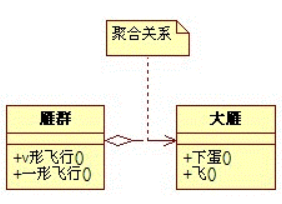
classBird:Animal                      class WideGoose:IFly  
{                                       {  
   //继承动物类                                 //实现飞翔接口  
}                                       }

5.      企鹅与气候有很大的关系，企鹅需要“知道”气候的变化，需要“了解”气候规律。当一个类“知道”另一个类时，可以用关联(association)关系。关联关系用实线箭头来表示。



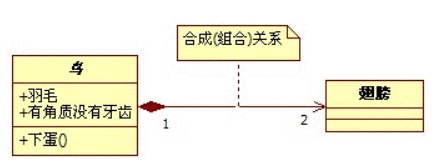
class Penguin :Bird  
{  
   private Climateclimate;//在企鹅Penguin中，引用到气候Climate对象  
}

6.      “大雁”和“雁群”这两个类。大雁是群居动物，每只大雁都属于一个雁群，一个雁群可以有多只大雁。所以它们之间就满足聚合(Aggregation)关系。聚合表示一种弱的“拥有”关系，体现的是A对象可以包含B对象，但B对象不是A对象的一部分。聚合关系用空心的菱形+ 实线箭头表示。



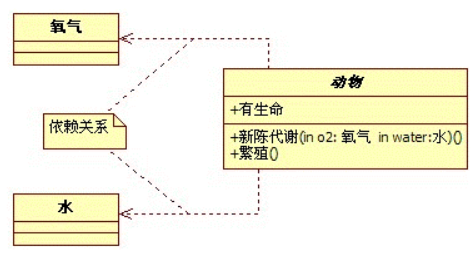
classWideGooseAggregate  
{  
   private WideGoose[]arrayWideGoose;  
   //在雁群WideGooseAggregate类中，有大雁数组对象arrayWideGoose  
}

7.      “鸟”和“翅膀”这两个类。鸟和翅膀似整体和部分的关系，并且翅膀和鸟的生命周期是相同的，在这里鸟和其翅膀就是合成关系。合成(composition)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样。合成关系用实心的的菱形+实线箭头来表示。另外，合成关系的连线两端还有一个数字“1”和数字“2”，，这被称为基数。表明这一端的类可以有几个实例，很显然，一个鸟应该有两支翅膀。如果一个类可能有无数个实例，则就用“n”来表示。关联关系，聚合关系也可以有基数的。



class Bird  
{  
  private Wing wing;  
  public Bird()  
   {  
      wing=new Wing();  
    //在鸟Bird类中，初始化时，实例化翅膀Wing,它们之间同时生成  
   }  
}

8.      “动物”、“氧气”与“水”之间。动物有几大特征，比如有新陈代谢，能繁殖。而动物要有生命，需要氧气，水以及食物等。也就是说动物依赖于氧气和水。它们之间是依赖关系(Dependency),用虚线箭头来表示。



abstract class Animal  
{  
   public bolism(Oxygenoxygen,Water water)  
    {  
    }  
}