[UML类图画法全程解析](http://blog.csdn.net/zhq56030207/article/details/6258983)

学习设计模式，UML类图画法是基础，通过UML类图，能更好地和大家交流，也能很容易就表达出自己的设计想法，它就好比普通话，是一种标准语言。

本节向大家介绍一下UML类图画法，主要包括UML类图元素和关系画法，希望通过本文的介绍，你对UML类图画法有一定的认识。

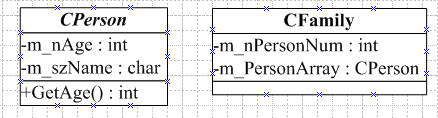
软件设计起步：UML类图画法

学习设计模式，画UML类图是基础，通过UML类图，能更好地和大家交流，也能很容易就表达出自己的设计想法，它就好比普通话，是一种标准语言。  
现在流行的主要工具有两种：RationalRose和MicrosoftVisio，这两种工具都比较易用，选择哪种工具就看个人的喜好了。本人对Microsoft的软件比较有好感，所以自然MicrosoftVisio2003是我的首选。

## **UML类图常用元素。**

**类：**

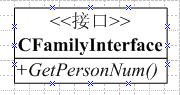
类是一种复杂的数据类型，它是将不同类型的数据和与这些数据相关的操作封装在一起的集合体。

[](http://images.51cto.com/files/uploadimg/20100628/1812300.jpg)

CPerson是一个抽象类，它是不能被实例化的，而CFamily可以被实例化。

**接口：**

接口是被调用者调用的一组操作方法。

[](http://images.51cto.com/files/uploadimg/20100628/1812301.jpg)

其实CPerson也可以作为接口。

## **UML类图常用关系。**

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 is-a



### **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。

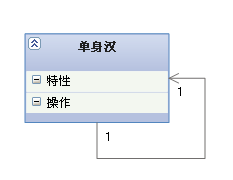
        【代码体现】：成员变量

        【箭头及指向】：带普通箭头的实心线，指向被拥有者 has-a



        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 use-a



### **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：

