

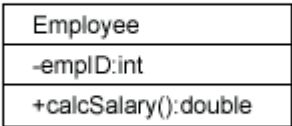
# 详解八大UML类图符号的表示法 - Colin、 - CSDN博客

## 类 (Class)

类 (**图A**) 是对象的蓝图，其中包含3个组成部分。第一个是Java中定义类名。第二个是属性 (attributes)。第三个是该类提供的方法。

属性和操作之前可附加一个可见性修饰符。加号 (+) 表示具有公共可见性。减号 (-) 表示私有可见性。#号表示受保护的可见性。省略这些修饰符表示具有package (包) 级别的可见性。如果属性或操作具有下划线，表明它是静态的。在操作中，可同时列出它接受的参数，以及返回类型，如图A的“Java”区域所示。

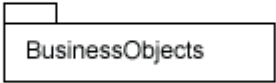
图A

Java	UML
<pre>public class Employee {     private int empID;      public double calcSalary() {         ...     } }</pre>	 <pre>classDiagram     class Employee {         -empID:int         +calcSalary():double     }</pre>

## 包 (Package)

包 (**图B**) 是一种常规用途的组合机制。UML中的一个包直接对应于Java中的一个包。在Java中，一个包可能含有其他包、类或者同时含有这两者。进行建模时，你通常拥有逻辑性的包，它主要用于对你的模型进行组织。你还会拥有物理性的包，它直接转换成系统中的Java包。每个包的名称对这个包进行了惟一性的标识。

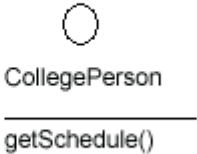
图B

Java	UML
<pre>package BusinessObjects;  public class Employee {  }</pre>	

## 接口（Interface）

接口（图C）是一系列操作的集合，它指定了一个类所提供的服务。它直接对应于Java中的一个接口类型。接口既可用图C的那个图标来表示，也可由附加了<<interface>>的一个标准类来表示。通常，根据接口在类图上的样子，就能知道与其他类的关系。

图C

Java	UML
<pre>public interface CollegePerson {     public Schedule getSchedule(); }</pre>	

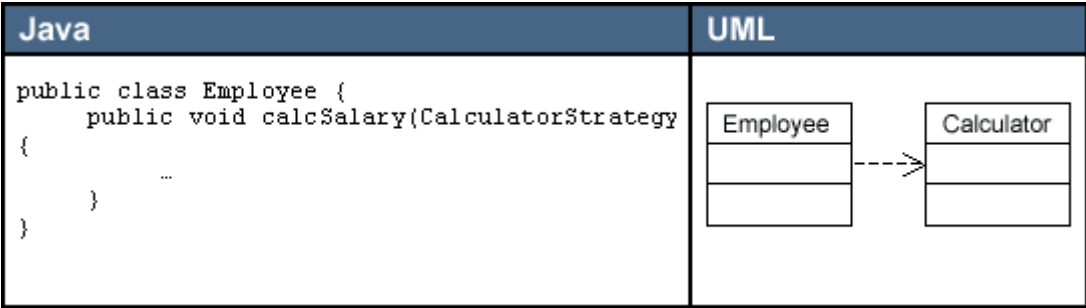
## 关系

后面的例子将针对某个具体目的来独立地展示各种关系。虽然语法无误，但这些例子可进一步精炼，在它们的有效范围内包括更多的语义。

## 依赖（Dependency）

实体之间一个“使用”关系暗示一个实体的规范发生变化后，可能影响依赖于它的其他实例（图D）。更具体地说，它可转换为对不在实例作用域内的一个类或对象的任何类型的引用。其中包括一个局部变量，对通过方法调用而获得的一个对象的引用（如下例所示），或者对一个类的静态方法的引用（同时不存在那个类的一个实例）。也可利用“依赖”来表示包和包之间的关系。由于包中含有类，所以你可根据那些包中的各个类之间的关系，表示出包和包的关系。

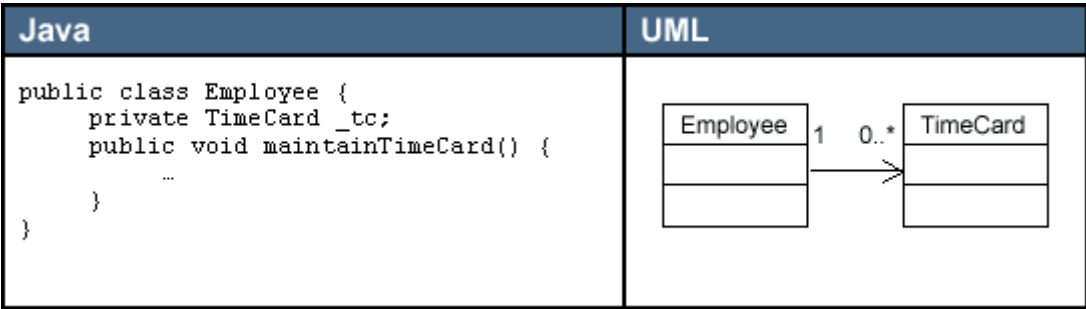
图D



关联（Association）

实体之间的一个结构化关系表明对象是相互连接的。箭头是可选的，它用于指定导航能力。如果没有箭头，暗示是一种双向的导航能力。在Java中，关联（图E）转换为一个实例作用域的变量，就像图E的“Java”区域所展示的代码那样。可为一个关联附加其他修饰符。多重性（Multiplicity）修饰符暗示着实例之间的关系。在示范代码中，Employee可以有0个或多个的TimeCard对象。但是，每个TimeCard只从属于单独一个Employee。

图E

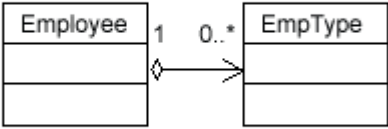


## 聚合（Aggregation）

聚合（图F）是关联的一种形式，代表两个类之间的整体/局部关系。聚合暗示着整体在概念上处于比局部更高的一个级别，而关联暗示两个类在概念上位于相同的级别。聚合也转换成Java中的一个实例作用域变量。

关联和聚合的区别纯粹是概念上的，而且严格反映在语义上。聚合还暗示着实例图中不存在回路。换言之，只能是一种单向关系。

图F

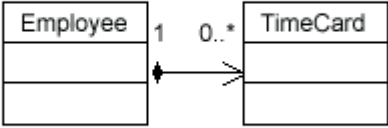
Java	UML
<pre>public class Employee {     private EmpType et;     public EmpType getEmpType() {         ...     } }</pre>	

## 合成（Composition）

合成（图G）是聚合的一种特殊形式，暗示“局部”在“整体”内部的生存期职责。合成也是非共享的。所以，虽然局部不一定要随整体的销毁而被销毁，但整体要么负责保持局部的存活状态，要么负责将其销毁。局部不可与其他整体共享。但是，整体可将所有权转交给另一个对象，后者随即将承担生存期职责。

Employee和TimeCard的关系或许更适合表示成“合成”，而不是表示成“关联”。

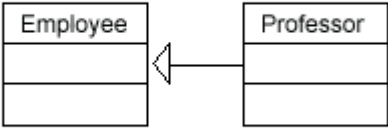
图G

Java	UML
<pre>public class Employee {     private TimeCard tc;     public void maintainTimeCard() {         ...     } }</pre>	

## 泛化（Generalization）

泛化（图H）表示一个更泛化的元素和一个更具体的元素之间的关系。泛化是用于对继承进行建模的UML元素。在Java中，用*extends*关键字来直接表示这种关系。

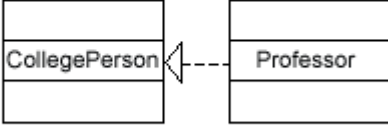
图H

Java	UML
<pre>public abstract class Employee {  } public class Professor extends Employee {  }</pre>	

## 实现（Realization）

实例（图I）关系指定两个实体之间的一个合同。换言之，一个实体定义一个合同，而另一个实体保证履行该合同。对Java应用程序进行建模时，实现关系可直接用*implements*关键字来表示。

图I

Java	UML
<pre>public interface CollegePerson {  } public class Professor implements CollegePers  }</pre>	

- 1.类（Class）：使用三层矩形框表示。  
第一层显示类的名称，如果是抽象类，则就用斜体显示。  
第二层是字段和属性。  
第三层是类的方法。  
注意前面的符号， '+' 表示public， '-' 表示private， '#' 表示protected。
- 2.UML类图符号之接口：使用两层矩形框表示，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。
- 3.UML类图符号之继承类（extends）：用空心三角形+实线来表示。
- 4.UML类图符号之实现接口（implements）：用空心三角形+虚线来表示
- 5.UML类图符号之关联（Association）：用实线箭头来表示，例如：燕子与气候
- 6.UML类图符号之聚合（Aggregation）：用空心的菱形+实线箭头来表示

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，例如：公司和员工

组合（Composition）：用实心的菱形+实线箭头来表示

组合：部分和整体的关系，并且生命周期是相同的。例如：人与手

7.UML类图符号之依赖（Dependency）：用虚线箭头来表示，例如：动物与氧气

8.UML类图符号之基数：连线两端的数字表明这一端的类可以有几个实例，比如：一个鸟应该有两只翅膀。如果一个类可能有无数个实例，则就用‘n’来表示。关联、聚合、组合是有基数的。

---