# 域对象之间的关系

【https://www.2cto.com/database/201507/423654.html】

【https://blog.csdn.net/qq\_31655965/article/details/54645220】

## 一般化（泛化）：

泛化，包括我们常说的继承和实现

## 依赖：

依赖指的是类之间的访问关系，如果类A访问类B的属性和方法，或者类A负责类实例化类B，那么可以说A依赖B。和关联关系不同，无需把类B定义为类A的属性。

依赖关系有如下三种情况：

1. A类是B类中的（某个方法的）局部变量；
2. A类是B类方法当中的一个参数；
3. A类向B类发送消息，从而影响B类发生变化；



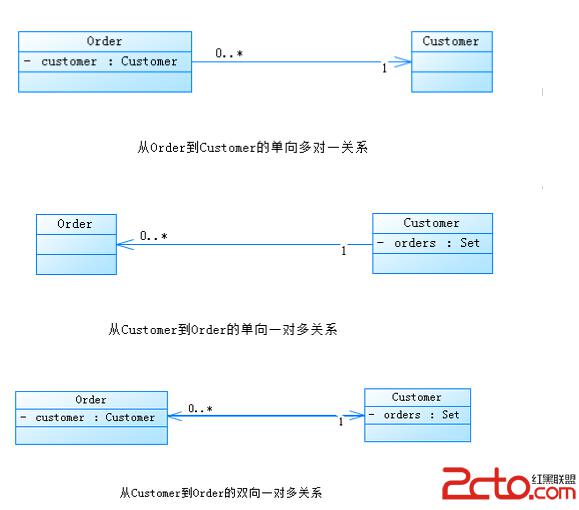
Driver（B类）依赖于Car（A类）

## 关联：

类之间的应用关系，可以有一对一、一对多和多对多，例如customer和order之间就是一对多。

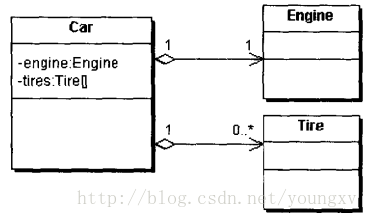
【注】管理关系可以有单向关联和双向关联

|  |
| --- |
| public class Order{  private Customer customer;  } |



## 聚合：

聚合关系是关联关系的一种特例，是强关联关系，聚合指的是整体与部分之间的关系，部分即可依赖于整体，也可脱离整体。



在代码层面，聚合和关联是一致的，只能从语义级别来区分。

普通的关联关系中，A类和B类没有必然的联系，而聚合关系中，需要B类是A类的一部分，是一种“has-a”的关系，即A has a B，比如家庭有孩子，屋子里有空调。

但是has不是must has，A可以有B，也可以没有。A是整体，B是部分，整体与部分之间是可分离的，他们具有各自的生命周期，部分可以属于一个整体对象，也可以为多个整体对象共享。

**不同于关联关系的平等地位，聚合关系中两个类的地位是不平等的。**

## 组合关系：

组合也是关联关系的一种特例，它体现的是一种contains-a的关系，这种关系比聚合更强，也称为强聚合。

先看一段代码：

|  |
| --- |
| public class Person{  //一个人有鼻子有眼睛,鼻子和眼睛是人的一部分组成,所以人和鼻子、人和眼睛就是组合关系  private Eye eye = new Eye();  private Nose nose = new Nose();  } |

组合关系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。

就像你有鼻子有眼睛，如果你一不小心失去生命周期，鼻子和眼睛的生命周期也会结束，而且鼻子和眼睛不能脱离你单独存在。

只看代码，是无法区分关联、聚合和组合的，具体是哪一种关系，只能从语义级别来区分：

**同样，组合关系中，两个类的关系也是不平等的。**