# 1、类之间关系

## 实现关系(realize)

类的继承结构表现在UML中为：实现（realize）和泛化（generalize），继承关系为is-a的关系（判断两个对象是否为继承关系通过判断是否为is-a关系即可，比如程序猿是哺乳动物）。

实现关系用一条空心箭头的虚线表示。

在代码中，实现关系表现为继承抽象类。

例如：车是一个抽象的概念（在C++中用抽象类表示，在Java中用接口表示），在现实中无法直接用来定义对象，只有具体指明了子类（具体什么车，是卡车、汽车还是自行车），才可以用来定义对象。

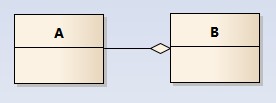
## 泛化关系(generalization)

泛化关系用一条空心箭头的实线表示。

在代码中，泛化关系表现为继承非抽象类。

## 聚合关系(aggregation)

聚合关系用一条带空心菱形箭头的直线表示。

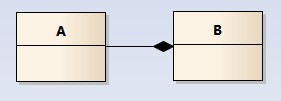


聚合关系用于表示实体对象之间的关系，表示整体由部分构成的语义。

与组合关系不同的是，整体和部分不是强依赖的，即使整体不存在了，部分仍然存在。

## 组合关系(composition)

组合关系用一条带实心菱形箭头直线表示。



与聚合关系一样，组合关系同样表示整体由部分构成的语义。

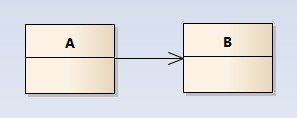
但组合关系是一种强依赖的特殊聚合关系，如果整体不存在了，则部分也不存在了。

## 关联关系(association)

关联关系是用一条直线表示的。

它描述不同类的对象之间的结构关系，它是一种静态关系，通常与运行状态无关，一般由常识等因素决定的，它一般用来定义对象之间静态的、天然的结构，所以，关联关系是一种“强关联”的关系。

关联关系默认不强调方向，表示对象间相互知道；如果特别强调方向，如下图，表示A知道B，但 B不知道A：

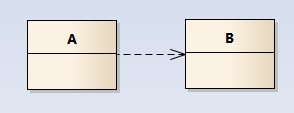


在最终代码中，关联对象通常是以成员变量的形式实现的。

## 依赖关系(dependency)

依赖关系是用一套带箭头的虚线表示的。

如下图表示A依赖于B，它描述一个对象在运行期间会用到另一个对象的关系：



与关联关系不同的是，它是一种临时性的关系，通常在运行期间产生，并且随着运行时的变化，依赖关系也可能发生变化。

显然，依赖也有方向，双向依赖是一种非常糟糕的结构，我们总是应该保持单向依赖，杜绝双向依赖的产生；

注：在最终代码中，依赖关系体现为类构造方法及类方法的传入参数，箭头的指向为调用关系；依赖关系除了临时知道对方外，还是“使用”对方的方法和属性。

# 2、时序图