一、首先来罗列一下这六种关系都有什么

**1.泛化（Generalization）**

**2. 实现（Realization）**

**3. 关联（Association)**

**4. 聚合（Aggregation）**

**5. 组合/合成(Composition)**

**6. 依赖(Dependency)**

各种关系的强弱顺序：

**泛化 = 实现 > 组合/合成 > 聚合 > 关联 > 依赖**

有些版本将泛化和实现统称为**一般化关系**

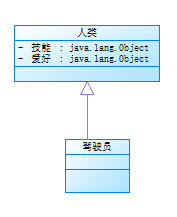
二、详细说一下这六种关系

**1.泛化（Generalization）**

　　【泛化关系】： 是一种继承关系，表现在类与类的继承关系；接口与接口的继承关系。驾驶员继承人类

　　【代码体现】：extends

　　【箭头指向】：带三角箭头的实线，箭头指向父类或者父接口

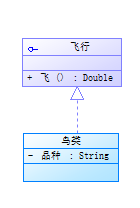


**2. 实现（Realization）**

【实现关系】： 是实现类对接口实现的体现。鸟实现飞行的接口

　　【代码体现】：implements

　　【箭头指向】：带三角箭头的虚线线，箭头指向接口

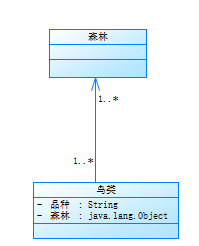


**3. 关联（Association)**

　　【关联关系】： 是类与类之间的链接，它使一个类知道另一个类的属性和方法。关联可以是双向的，也可以是单向的。鸟栖息在某片森林

　　【代码体现】：成员变量

　　【箭头指向】：带普通箭头的实线，双向的关联可以有两个箭头或者没有箭头，单向的只有一个箭头，表示关联的方向。



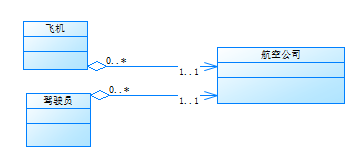
**4. 聚合（Aggregation）**

【聚合关系】： 是关联关系的一种，是强的关联关系。聚合是整体和部分之间的关系，且部分可以离开整体单独存活。

　　关联和聚合从Java语法上分辨不出，要考虑逻辑关系，如果不确定，可设置为关联关系。下面，飞机和驾驶员都属于航空公司的一部分

　　【代码体现】：成员变量

　　【箭头指向】：带空心菱形的实线，菱形指向整体。



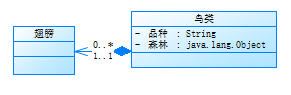
**5. 组合/合成(Composition)**

　　【合成关系】： 是关联关系的一种，是比聚合强的关系。他要求普遍的聚合关系中代表整体的对象负责部分的对象的生命周期，合成关系不能共享。

　　就是说：代表部分的对象每时每刻都只能与一个对象发生合成关系。如果不能确定，可以设置为聚合关系。如鸟的翅膀，和鸟的关系就是合成关系

　　【代码体现】：成员变量

　　【箭头指向】：带实心菱形的实线，菱形指向整体。



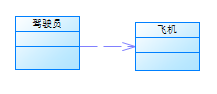
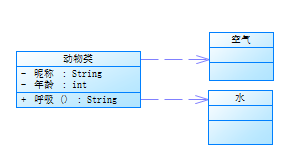
**6. 依赖(Dependency)**

【依赖关系】：是类与类之间的连接，依赖总是单项的。依赖表示一个类的定义依赖于另一个类的定义。如飞机驾驶员驾驶飞机，就是一种依赖

　　与关联关系不同的是，飞机并不是驾驶员的一个成员变量，毕竟并非所有的飞架驾驶员都拥有飞机。动物依赖水和空气也是个不错的例子

　　【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

三、下面用一个整合的图来展示这六种关系

