


继承 > 实现 > 组合 > 聚合 > 关联 > 依赖

原创张紫娃已于 2023-08-08 12:49:46 修改阅读量1k收藏1点赞数1

版权

分类专栏：框架

文章标签：javadevtools开发语言

 框架 专栏收录该内容

3 订阅93 篇文章

订阅专栏

文章目录

- 类与类之间的关系（即事物关系）有如下6种
- 设计 类间关系 遵循的原则
 - 高内聚低耦合
 - 针对接口编程优于针对实现编程
 - 组合优于继承
- A is-a B 泛化（继承，实现）
- A has-a B 包含（紧密程度：组合 > 聚合 > 关联）
- A need-a B 依赖（依赖）

类与类之间的关系（即事物关系）有如下6种

- 继承关系（Generalization），又叫泛化
- 实现关系（Realization）
- 组合关系（Composition）
- 聚合关系（Aggregate）
- 关联关系（Association）
- 依赖关系（Dependency）

设计 类间关系 遵循的原则

高内聚低耦合

耦合度：继承 > 实现 > 组合 > 聚合 > 关联 > 依赖

针对接口编程优于针对实现编程

组合优于继承

A is-a B 泛化（继承，实现）

在 继承关系 中，子类继承父类的所有功能，父类所具有的属性、方法，子类都应该有。除了与父类一致的信息，子类中还包括额外的信息。例如，公交车、出租车都是车

接口（包括抽象类）是方法的集合

在 实现关系 中，类实现了接口，类中的方法实现了接口声明的所有方法。例如，人、动物都吃饭都睡觉

A has-a B 包含（紧密程度：组合 > 聚合 > 关联）

关联关系 是类与类之间 最常用 的一种关系，表示一类对象与另一类对象之间有联系。比如说，人有汽车，人有电脑；整体和部分的关系，可以分割，后来形成在一起

```
1 public class Person {
2     private Car car;
3 }
```

```
3 | }
4 |
5 | 对于赋值：
6 | 1. 可以在构造方法里赋值
7 | 2. 可以通过set方法赋值
```

Association is a relation between two separate classes which establishes through their Objects.

Association can be

- 1、one-to-one,
- 2、one-to-many, i.e. Bank can have many employees
- 3、many-to-one,
- 4、many-to-many.

In Object-Oriented programming, an Object communicates to another object to use functionality and services provided by that object.

Composition and Aggregation are the two forms of association.

组合、聚合也属于关联关系，他们3个代码结构一样，只是关联关系的类间关系比其他两种关系要弱。

聚合关系 也表示类之间整体与部分的关系，成员对象是整体对象的一部分，但是成员对象可以脱离整体对象独立存在。

例如，工人和工作服是整体与部分的关系，但是可以分开，没有共同的生命周期。工作服可以穿在别的司机身上，工人也可以换别人的工作服。再比如说，汽车和车轮，电脑和主板。

```
1 | public class Car {
2 |     private Wheel wheel;
3 | }
4 |
5 | 对于赋值：
6 | 1. 可以在构造方法里赋值
7 | 2. 可以通过set方法赋值
```

It is a special form of Association

where:

- 1、It represents Has-A's relationship.
- 2、It is a **unidirectional** association i.e. a one-way relationship.

For example, a department can have students but **vice versa** is not possible and **thus** unidirectional in nature.

- 3、In Aggregation, both the entries can survive **individually** which means ending one entity will not affect the other entity.

When do we use Aggregation ?

Code reuse is best achieved by aggregation.

组合关系 表示类之间整体与部分的关系，整体与部分有一致的生存期。一旦整体对象不存在，部分对象也将不存在，整体和部分是同生共死的关系。 **皮之不存，毛将焉附**

例如，人和脑袋，人和大脑，人和心脏

```
1 | public class Person {
2 |     private Head head;
3 |     private Heart heart;
4 | }
5 |
6 | 对于赋值：
7 | 1. 可以在构造方法里赋值
8 | 2. 可以通过set方法赋值
```

Composition is a **restricted** form of Aggregation in which two entities are highly dependent on each other.

- 1、It represents part-of relationship.
- 2、In composition, both entities are dependent on each other.
- 3、When there is a composition between two entities, the composed object cannot exist without the other entity.

Aggregation vs Composition

1. Dependency: Aggregation implies a relationship where the child can exist independently of the parent. whereas Composition implies a relationship where the child cannot exist independent of the parent.
2. Type of Relationship: Aggregation relation is “has-a” and composition is “part-of” relation.
3. Type of association: Composition is a strong Association whereas Aggregation is a weak Association.

A need-a B 依赖（依赖）

依赖关系 是一种“使用”关系，当需要表示一个事物使用另一个事物时，使用依赖关系。在大多数情况下，依赖关系体现在某个类的方法使用另一个类的对象作为**参数**。

例如，汽车依赖汽油，如果没有汽油，则汽车将无法行驶。

```
1 public class Car {
2     public void driveDistance(Oil oil){
3         ...
4     }
5 }
```

- 7 对于赋值：
- 8 可以在方法中自己创建
- 9 可以在方法中传递进来

-----读书笔记摘自 书名：《设计模式就该这样学：基于经典框架源码和真实业务场景》（谭勇德）


Java类与类之间的关系UML图表示

原创 山风wind 于 2022-05-28 21:26:57 发布 阅读量1.8k 收藏 10 点赞数

版权

分类专栏： JAVA基础

文章标签： java uml 开发语言

 JAVA基础 专栏收录该内容

2 订阅 28 篇文章 订阅专栏

Java类与类之间的关系（关联、组合、聚合、依赖、继承、实现）

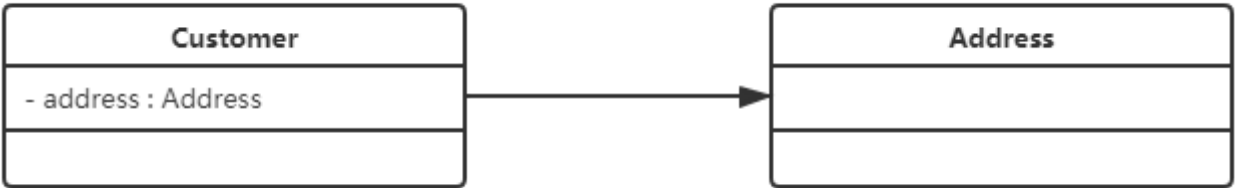
- 1.关联关系
 - 1.1 单向关联
 - 1.2 双向关联
 - 1.3 自关联
- 2.聚合关系
- 3.组合关系
- 4.依赖关系
- 5.继承关系
- 6.实现关系

1.关联关系

关联关系是对象之间的一种引用关系，用于表示一类对象与另一类对象之间的联系，如老师和学生、师傅和徒弟、丈夫和妻子等。关联关系是类与类之间最常用的一种关系，分为一般关联关系、聚合关系和组合关系。

关联又可以分为**单向关联**，**双向关联**，**自关联**。

1.1 单向关联



在 **UML类图** 中单向关联用一个带箭头的实线表示。上图表示每个顾客都有一个地址，这通过让Customer类持有一个类型为Address的成员变量类实现。

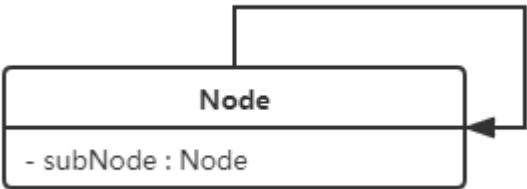
1.2 双向关联



从上图中我们很容易看出，所谓的双向关联就是双方各自持有对方类型的成员变量。

在UML类图中，双向关联用一个不带箭头的直线表示。上图中在Customer类中维护一个List<Product>，表示一个顾客可以购买多个商品；在Product类中维护一个Customer类型的成员变量表示这个产品被哪个顾客所购买。

1.3 自关联



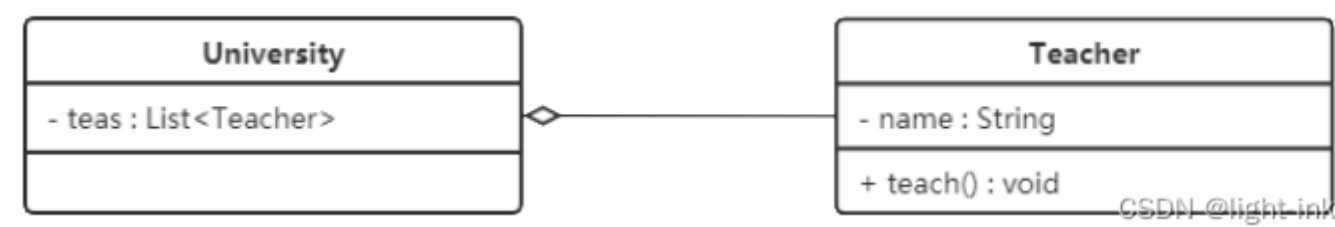
自关联在UML类图中用一个带有箭头且指向自身的线表示。上图的意思就是Node类包含类型为Node的成员变量，也就是“自己包含自己”。链表的实现就是自关联。

2.聚合关系

聚合关系是关联关系的一种，是**强关联关系**，是整体和部分之间的关系。

聚合关系也是通过成员对象来实现的，其中**成员对象是整体对象的一部分**，但是**成员对象可以脱离整体对象而独立存在(组合与聚合的区别)**。例如，学校与老师的关系，学校包含老师，但如果学校停办了，老师依然存在。

在 UML 类图中，聚合关系可以用带空心菱形的实线来表示，菱形指向整体。下图所示是大学和教师的关系图：

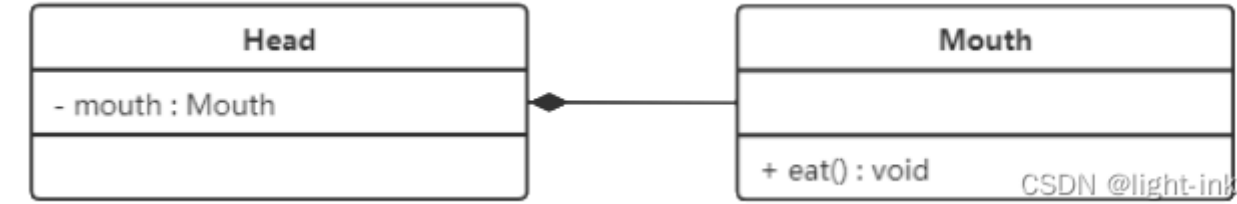


3.组合关系

组合表示类之间的整体与部分的关系，但它是一种**更强烈的聚合关系**。

在组合关系中，**整体对象可以控制部分对象的生命周期**，**一旦整体对象不存在，部分对象也将不存在**，**部分对象不能脱离整体对象而存在**。例如，头和嘴的关系，没有了头，嘴也就不存在了。

在 UML 类图中，组合关系用带实心菱形的实线来表示，菱形指向整体。下图所示是头和嘴的关系图：

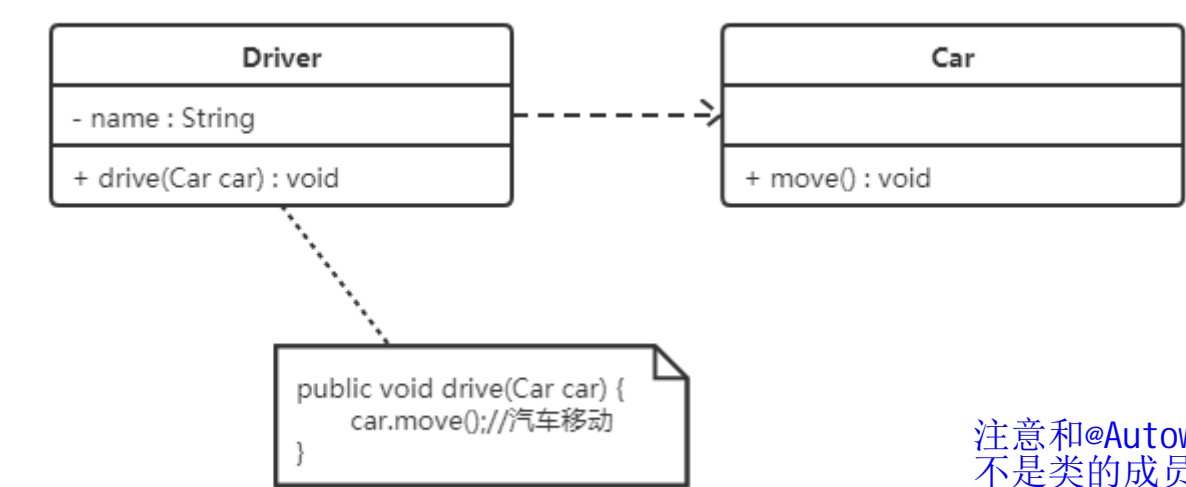


代码表示就是
class Head{
 private Mouth mouth = new Mouth;
}
Head和Mouth生命周期一致了，对比聚合属性mouth是在构造方法里传入的。Head实例好Mouth生命周期不一样。

4.依赖关系

依赖关系是一种使用关系，它是**对象之间耦合度最弱的一种关联方式**，是临时性的关联。在代码中，某个类的方法通过局部变量、方法的参数或者对静态方法的调用来访问另一个类（被依赖类）中的某些方法来完成一些职责。

在 UML 类图中，依赖关系使用带箭头的虚线来表示，箭头从使用类指向被依赖的类。下图所示是司机和汽车的关系图，司机驾驶汽车：

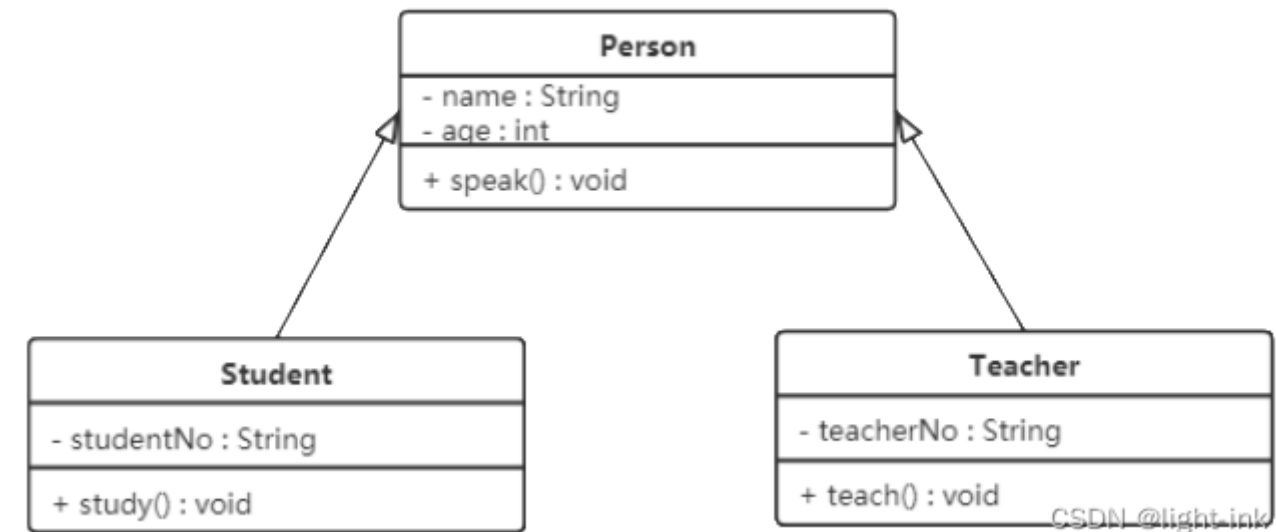


注意和@Autowired依赖注入不同，这里是局部方法里的参数不是类的成员变量。

5.继承关系

继承关系是对象之间耦合度最大的一种关系，表示**一般与特殊的关系**，是父类与子类之间的关系，是一种继承关系。

在 UML 类图中，泛化关系用带空心三角箭头的实线来表示，箭头从子类指向父类。在代码实现时，使用面向对象的继承机制来实现泛化关系。例如，Student 类和 Teacher 类都是 Person 类的子类，其类图如下图所示：



6.实现关系

**实现关系是接口与实现类之间的关系。 **在这种关系中，类实现了接口，类中的操作实现了接口中所声明的所有的抽象操作。

在 UML 类图中，实现关系使用带空心三角箭头的虚线来表示，箭头从实现类指向接口。例如，汽车和船实现了交通工具。

