UML图中类之间的关系:依赖,泛化,关联,聚合,组合,实现

分类： 设计模式 Java 2012-06-07 18:34 17113人阅读 评论(23) 收藏 举报

uml class button string interface java

目录(?) [+]

类与类图

1) 类(Class)封装了数据和行为，是面向对象的重要组成部分，它是具有相同属性、操作、关系的对象集合的总称。

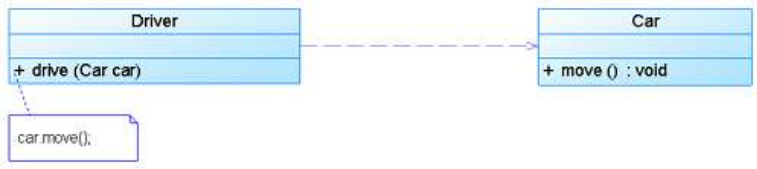
2) 在系统中，每个类具有一定的职责，职责指的是类所担任的任务，即类要完成什么样的功能，要承担什么样的义务。一个类可以有多种职责，设计得好的类一般只有一种职责，在定义类的时候，将类的职责分解成为类的属性和操作（即方法）。

3) 类的属性即类的数据职责，类的操作即类的行为职责

一、依赖关系(Dependence)

依赖关系（**Dependence**）：假设A类的变化引起了B类的变化，则说名B类依赖于A类。

- 依赖关系(Dependency) 是一种使用关系，特定事物的改变有可能会影响到使用该事物的其他事物，在需要表示一个事物使用另一个事物时使用依赖关系。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。
- 在UML中，依赖关系用带箭头的虚线表示，由依赖的一方指向被依赖的一方。




```
public class Driver
{
    public void drive(Car car)
    {
        car.move();
    }
    ……
}
public class Car
{
    public void move()
    {
        ......
    }
    ……
}
```

依赖关系有如下三种情况：

1、A类是B类中的（某中方法的）局部变量；

2、A类是B类方法当中的一个参数；

NOSQL Mongo (0)
分布式 (1)
数据结构与算法 xi (0)
协议 (1)
信息论的熵 (0)
关于php的libevent扩展的应用 (0)
libevent简单介绍 (0)
SOA (0)

文章存档
2015年05月 (2)
2015年04月 (1)
2015年01月 (2)
2014年10月 (1)
2014年09月 (1)
展开

阅读排行
八大排序算法 (149047)
Mysql 多表联合查询效率 (115534)
socket阻塞与非阻塞，同 (76928)
hbase安装配置（整合到 (74219)
Nginx工作原理和优化、i (66084)
深入理解java异常处理机 (58399)
设计模式 (十八) 策略模 (55133)
Hadoop集群配置（最全 (54576)
Java输入输出流 (50884)
PageRank算法 (46558)

评论排行
八大排序算法 (54)
深入理解java异常处理机 (48)
socket阻塞与非阻塞，同 (46)
硬盘的读写原理 (35)
设计模式 (十八) 策略模 (35)
海量数据处理算法—Bit-M (25)
设计模式（一）工厂模式 (24)
UML图中类之间的关系:依 (23)
PHP SOCKET编程 (22)
HDFS写入和读取流程 (20)

推荐文章
* Android HandlerThread 源码分析
*从零学编程：写一封情书
*小胖学PHP总结：PHP的循环语句
*Spring基于注解@AspectJ的AOP
*HTML5梦幻之旅：仿Qt示例Drag and Drop Robot（换装机器人）

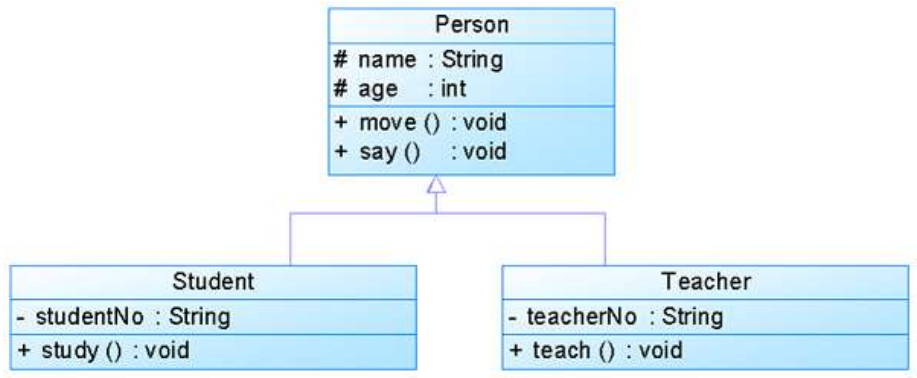
最新评论
Hadoop Hive sql语法详解
lifangzhen0: hiveSQL入门，写

3、A类向B类发送消息，从而影响B类发生变化；

二、泛化关系（Generalization）

泛化关系（Generalization）：A是B和C的父类，B,C具有公共类（父类）A，说明A是B,C的一般化（概括，也称泛化）

- 泛化关系(Generalization)也就是继承关系，也称为“is-a-kind-of”关系，泛化关系用于描述父类与子类之间的关系，父类又称作基类或超类，子类又称作派生类。在UML中，泛　化关系用带空心三角形的直线来表示。
- 在代码实现时，使用面向对象的继承机制来实现泛化关系，如在Java语言中使用extends关键字、在C++/C#中使用冒号“：”来实现。




```
public class Person
{
    protected String name;
    protected int age;
    public void move()
    {
        ……
    }
    public void say()
   {
        ……
    }
}
public class Student extends Person
{
    private String studentNo;
    public void study()
    {
        ……
    }
}
```

在UML当中，对泛化关系有三个要求：

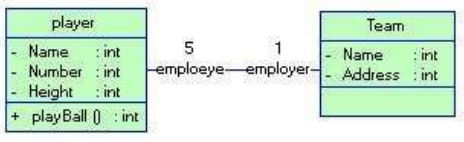
1、子类与父类应该完全一致，父类所具有的属性、操作，子类应该都有；

2、子类中除了与父类一致的信息以外，还包括额外的信息；

3、可以使用父类的实例的地方，也可以使用子类的实例；

三、关联关系（Association）

关联关系（Association）：类之间的联系，如客户和订单，每个订单对应特定的客户，每个客户对应一些特定的订单，再如篮球队员与球队之间的关联（下图所示）。



其中，关联两边的"employee"和“employer”标示了两者之间的关系，而数字表示两者的关系的限制，是关联两者之间的多重性。通常有“*”（表示所有，不限），“1”（表示有且仅有一个），“0...”（表示0个或者多个），“0，1”（

表示0个或者一个），“n...m”(表示n到m个都可以),“m...*”（表示至少m个）。

- 关联关系(Association) 是类与类之间最常用的一种关系，它是一种结构化关系，用于表示一类对象与另一类对象之间有联系。
- 在UML类图中，用实线连接有关联的对象所对应的类，在使用Java、C#和C++等编程语言实现关联关系时，通常将一个类的对象作为另一个类的属性。
- 在使用类图表示关联关系时可以在关联线上标注角色名。

1) 双向关联：默认情况下，关联是双向的。

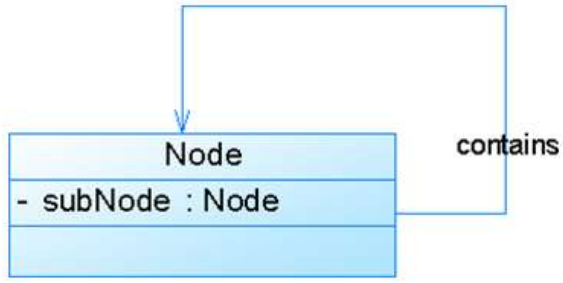

```
public class Customer
{
    private Product[] products;
    ……
}
public class Product
{
    private Customer customer;
    ……
}
```

2) 单向关联:类的关联关系也可以是单向的，单向关联用带箭头的实线表示.


```
public class Customer
{
    private Address address;
    ……
}

public class Address
{
    ……
}
```

3) 自关联：在系统中可能会存在一些类的属性对象类型为该类本身，这种特殊的关联关系称为自关联。




```
public class Node
{
    private Node subNode;
    ……
}
```

4) 重数性关联：重数性关联关系又称为多重性关联关系(Multiplicity)，表示一个类的对象与另一个类的对象连接的个数。在UML中多重性关系可以直接在关联直线上增加一个数字表示与之对应的另一个类的对象的个数。

表示方式	多重性说明
1..1	表示另一个类的一个对象只与一个该类对象有关系
0..*	表示另一个类的一个对象与零个或多个该类对象有关系

1..*	表示另一个类的一个对象与一个或多个该类对象有关系
0..1	表示另一个类的一个对象没有或只与一个该类对象有关系
m..n	表示另一个类的一个对象与最少m、最多n个该类对象有关系(m<=n)


```
public class Form
{
    private Button buttons[];
    ……
}
public class Button
{
    …
}
```

四、聚合关系（Aggregation）

聚合关系（**Aggregation**）:表示的是整体和部分的关系，整体与部分 可以分开.

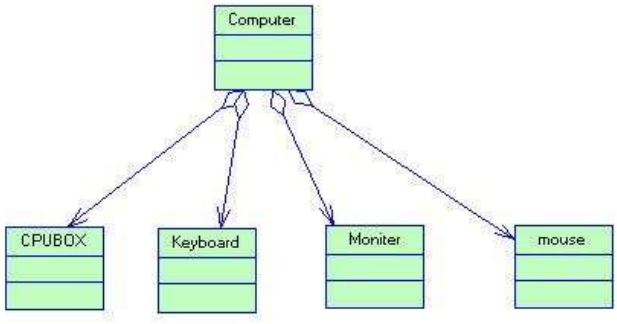
- 聚合关系(Aggregation) 表示一个整体与部分的关系。通常在定义一个整体类后，再去分析这个整体类的组成结构，从而找出一些成员类，该整体类和成员类之间就形成了聚合　关系。
- 在聚合关系中，成员类是整体类的一部分，即成员对象是整体对象的一部分，但是成员对象可以脱离整体对象独立存在。在UML中，聚合关系用带空心菱形的直线表示。


```
public class Car
{
    private Engine engine;
    public Car(Engine engine)
   {
        this.engine = engine;
    }

    public void setEngine(Engine engine)
    {
        this.engine = engine;
    }
    ……
}
public class Engine
{
    ……
}
```

如：电话机包括一个话筒

电脑包括键盘、显示器，一台电脑可以和多个键盘、多个显示器搭配，确定键盘和显示器是可以和主机分开的，主机可以选择其他的键盘、显示器组成电脑；



五、组合关系（Composition）

组合关系（Composition）：也是整体与部分的关系，但是整体与部分不可以分开.

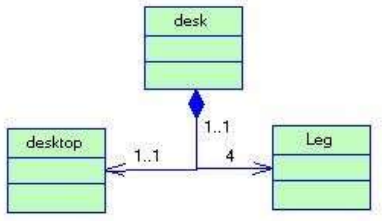
• 组合关系(Composition)也表示类之间整体和部分的关系，但是组合关系中部分和整体具有统一的生存期。一旦整体对象不存在，部分对象也将不存在，部分对象与整体对象之　间具有同生共死的关系。

• 在组合关系中，成员类是整体类的一部分，而且整体类可以控制成员类的生命周期，即成员类的存在依赖于整体类。在UML中，组合关系用带实心菱形的直线表示。


```
public class Head
{
    private Mouth mouth;
    public Head()
    {
    mouth = new Mouth();
    }
    ……
}

public class Mouth
{
    ……
}
```

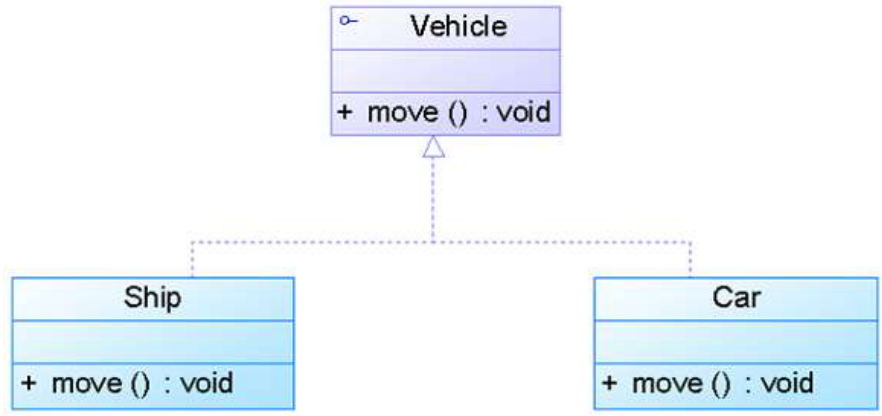
如：公司和部门，部门是部分，公司是整体，公司A的财务部不可能和公司B的财务部对换，就是说，公司A不能和自己的财务部分开； 人与人的心脏.



六、实现关系（Implementation)

实现关系（Implementation）：是用来规定接口和实线接口的类或者构建结构的关系，接口是操作的集合，而这些操作就用于规定类或者构建的一种服务。

• 接口之间也可以有与类之间关系类似的继承关系和依赖关系，但是接口和类之间还存在一种实现关系(Realization)，在这种关系中，类实现了接口，类中的操作实现了接口中所　声明的操作。在UML中，类与接口之间的实现关系用带空心三角形的虚线来表示。




```
public interface Vehicle
{
    public void move();
}
public class Ship implements Vehicle
{
    public void move()
    {
    ……
    }
}
public class Car implements Vehicle
{
    public void move()
    {
    ……
    }
}
```


上一篇 重构全面总结
下一篇 深入理解java嵌套类和内部类

查看评论

23楼 mayfla 2015-05-16 14:42发表

讲的很清楚，也找到了要找的问题。博主赞

22楼 matmat437 2015-04-27 13:43发表

呵呵

21楼 matmat437 2015-04-27 13:42发表

10楼说的对，他的问题是介绍组合和聚合的概念是对的，但是例子举的不对，组合是contain,用构造方法初始化成员类的实例，聚合是has, 不能用构造方法初始化。

20楼 nihaobukeqi 2015-01-13 14:13发表

真心觉得写得很好！我也收藏了！

19楼 fengkuang 2014-12-10 14:46发表

总结的很好啊，对于聚合和组合关系，我有个疑问：聚合中，如果B类作为A类成员变量，那么A类实例化的时候是怎么给B对象分配的内存空间，是在A类对象的空间里还是只是存了一个B类对象的地址？

18楼 eewj 2014-10-08 21:09发表

很好，很清晰

17楼 gisysj 2014-08-08 10:39发表

写得真好，赞一个，找了好久，这是好文章，思路很清晰，有代码来说明，非常不错

16楼 码鬼 2014-07-26 19:40发表

全面透彻，待我慢慢掌握

15楼 王静娜 2014-05-16 21:04发表

挺清晰的，学习了

14楼 追随我心ing 2014-05-15 11:00发表

写的非常好，思路清新，帮助我理清了这些概念，谢谢。

13楼 萨穆尔骑 2014-04-26 21:01发表

真心赞！

12楼 perock 2014-03-13 23:35发表

不错不错，楼主很用心

11楼 yaochunxu 2014-01-02 10:03发表

10楼说的对!

10楼 LoveAnnuoa 2014-01-01 10:19发表

有点问题吧？聚合应该是has，组合应该是contains

9楼 emilyhg 2013-11-26 15:09发表

为什么看不到图片

8楼 woruixu 2013-11-08 18:00发表

找了好久 看了这篇 很清晰了 感谢楼主

7楼 whyisyoung 2013-10-13 15:19发表

刚学UML，nice啊~

6楼 猿先生 2013-10-11 19:41发表

感谢博主！

5楼 yw610879290 2013-08-14 09:45发表

写的很不错

4楼 ironxue 2013-04-02 20:14发表

对菜鸟帮助很大啊。楼主强大。

3楼 zddeng_scir 2012-12-20 10:45发表

写的很清晰 谢谢楼主

2楼 avalon 2012-07-02 20:48发表

写得很不错，收留了！

1楼 jdluojing 2012-06-07 22:36发表

挺好的

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack
FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
Angular Cloud Foundry Redis Scala Django Bootstrap