一句话设计模式——快速掌握23种设计模式的核心思想(上)

2017-09-26 微辣鸡米饭 [架构之家](https://mp.weixin.qq.com/s?__biz=MzIyNjE4NDcyMA==&mid=2247484205&idx=1&sn=738851cffda7cf6da9d51e5487bf43c8&chksm=e8751852df0291442a03d5ec75d128c39006beee65e51b5e687732db1a987d9d1f507fdfadcd&mpshare=1&scene=1&srcid=0926eTHmWKXplSBL0KCEFkb7&key=a0a520779039cf2b983343f839cb6433692a6621c1aee3aa9d8c5f121918d871343d980dc23d21000e7e504d0f0da89a1c2167334db7ed921e8ea85f11f9d4b0f2b864c6c2f3ad859438a6eda52baa7a&ascene=1&uin=Mjg1Mzg5MzU%3D&devicetype=Windows+7&version=6205051a&pass_ticket=m7jrJ6n3B3wZ4RMxYevQWriAFvfGtvAcIen84ySwVng%3D&winzoom=1##)

《一句话设计模式》的想法起源于前段时间为一个培训机构上课的时候，每讲完一个设计模式我都会用一句话总结。总结往往不是“将抽象部分与它的实现部分分离，使它们都可以独立地变化”这种晦涩抽象的表达，而是通俗的表述最核心的部分。

这篇文章不适合深入学习设计模式，因为没有代码，没有模式扩展。

正确的使用方式：1. 初学者在深入学习之前先留下一个粗略的印象。2. 已经系统学习了设计模式的人快速回顾。

简单工厂模式

核心思想：**将类的创建过程与对象的使用过程分离。**

简单工厂模式

具体实现过程：定义一个工厂，根据参数（通常是字符串）的不同返回不同类的实例。因为我们希望将工厂生产的对象可以使用相同的方式处理，所以产品类类通常会实现相同的接口。

工厂方法模式

核心思想：**简单工厂基础上，通过实例化不同的工厂，来创建不同的类。**

工厂方法模式

具体实现过程：在简单工厂的基础上，对工厂进行抽象，将类的创建过程延迟到工厂的子类中完成。这样做的好处是当需要新增加一个新的产品类的时候，只需要新增一个“具体的工厂”就可以，而不需要像简单工厂一样，修改工厂类的具体实现。符合“开闭原则”。

抽象工厂模式

核心思想：**工厂方法基础上，让一个工厂生产一类对象。**

抽象工厂模式

具体实现过程：在工厂方法的基础上，在一个具体工厂中实现生产不同产品的方法。可以达到方便生产一个“产品簇”的目的。

建造者模式

核心思想：**定义一个建造过程，具体的建造放在子类当中，这样可以使用相同的建造过程构造不同的实例。**

建造者模式

原型模式

核心思想：**使用Object类的clone方法，快速复制对象。**

原型模式

单例模式

核心思想：**让一个类自身持有其实例的成员变量，声明为static和private，来达到系统中只有一个实例的目的。**

单例模式

具体实现过程：为了确保全系统只有一个实例，所以不可以让用户自己创建，所以这个类需要一个private的构造函数。还需要一个getInstance()方法用来获得这个唯一的实例，因为需要在系统的每一处都能获得，所以这个方法应该是public的。因为调用的getInstance()的目的是获取实例，只能在类上面调用，所以getInstance()应该是static的。

适配器模式

核心思想：**创建一个适配器类可以同时具备有现有类的行为，和期望的接口。**

对象适配器模式

具体实现过程：分为两种情况

1. 类适配器：适配器类实现期望的接口，继承现有的类已获得他的行为（方法）。
2. 对象适配器：适配器类实现期望的接口，并持有现有类的实例，通过实例调用方法。  
   目的都是相同的。

桥接模式

核心思想：**使用组合或者聚合的方式处理抽象和实现之间的联系，让具体的实现在不同维度上变化**

桥接模式

组合模式

核心思想：**让具体类和这些类的容器实现同一个接口，用户调用到容器类时使用递归调用，不需要区分这两者的区别。**

组合模式

外观模式

核心思想：**在外观类内实现具体的操作，用户只需要和外观类打交道，屏蔽方法调用细节。**

外观模式

享元模式

核心思想：**建一个对象池，复用需要在系统中多次出现的对象。**

装饰模式

核心思想：**，装饰器和被装饰的对象有相同的接口，装饰器在使用时获得被装饰的对象，为其增加新的操作。**

装饰模式

具体实现过程：不论是具体的组件，和装饰过后的组件，都希望可以用相同过的方式使用，所以实现同一个接口。装饰类需要持有一个具体组件的实例，才能调用他的方法，所以需要一个聚合关联。

代理模式

核心思想：**代理类持有具体实现类的实例，将在代理类上的操作转化为实例上方法的调用。**

代理模式