# **设计模式的六大原则**

## **1、\*单一职责原则**

一个类只负责一项职责，如若不然，就应该把类拆分。

## **2、里氏替换原则（克服继承的缺点）**

里氏替换原则中，子类可以扩展父类的功能，但不要改变父类原有的功能，对父类的方法尽量不要重写和重载。因为父类代表了定义好的结构，通过这个规范的接口与外界交互，子类不应该随便破坏它。

如果对每一个类型为S的对象o1，都有类型为T的对象o2，使得以T定义的所有程序P在所有的对象o1都代替成o2时，程序P的行为没有发生变化，那么类型S是类型T的子类型。

只要父类能出现的地方子类就可以出现，而且替换为子类也不会产生任何错误或异常，使用者可能根本就不需要知道是父类还是子类。但是，反过来就不行了，有子类出现的地方，父类未必就能适应。

## **3、\*依赖倒转原则（Dependence Inversion Principle）**

这个是开闭原则的基础，具体内容：面向接口编程。

(1)、模块间的依赖通过抽象发生，实现类之间不发生直接的依赖关系，其依赖关系是通过接口或抽象类产生的

(2)、接口或抽象类不依赖于实现类

(3)、实现类依赖于接口或抽象类

## **4、接口隔离原则（Interface Segregation Principle）**

这个原则的意思是：每个接口中不存在子类用不到却必须实现的方法，如果不然，就要将接口拆分。使用多个隔离的接口，比使用单个接口（多个接口方法集合到一个的接口）要好。

## **迪米特法则（Law of Demeter）**

一个类应该对自己需要耦合或调用的类知道最少，也就是对于被依赖的类，向外公开的方法应该尽可能的少。

(1)、假如A要实现的功能，需要调用B的多种能力时，请将B的多种能力封装成一个能力供A调用。

(2)、假如A要调用B持有的C的某个能力时，请在B 中封装好C的能力再供A调用，不要让A与C产生依赖。

## **6：\*开闭原则（Open Close Principle）**

开闭原则是一个非常基础的原则，其他的五个原则都是开闭原则的具体，也就是说其他的五个原则是指导设计的工具和方法，而开闭原则才是它们的精神领袖。只要我们遵守好其他的五大原则，那么我们设计的软件自然就遵守了开闭原则。

简单总结上面的五大原则就是：单一职责原则告诉我们实现类要职责单一；里式替换原则告诉我们不要破坏继承体系；依赖倒置原则告诉我们要面向抽象编程；接口隔离原则告诉我们设计接口要精简单一；迪米特法则告诉我们要降低耦合。而开闭原则告诉我们：要对修改关闭，对扩展开放。其实前面的五大原则一直反复强调的，几乎每一个原则都在强调的宗旨就是：解耦，单一，高内聚。

开闭原则解读：

(1)、通过接口或抽象类约束扩展，对扩展进行边界限定，不允许出现在接口或抽象类中不存在的public方法

(2)、参数类型、引用对象尽量使用接口或者抽象类，而不是实现类

(3)、抽象层尽量保持稳定，一旦确定即不允许修改

**PS：设计模式是解决问题的思想，重要的是思想，而不是刻意在写代码时去追求设计模式**