# 设计模式七大原则

【韩顺平Java设计模式】

## 单一职责原则：

### 基本介绍：

对类来说，即一个类应该只负责一项职责。如果类A负责两个职责：职责1，职责2。当职责1需求变更而改变A时，可能造成职责2执行错误，所以需要将类A的粒度分解为A1、A2。

## 二、接口隔离原则：

### 1、基本介绍：

客户端不应该依赖它不需要的接口，即一个类对另一个类的依赖，应该建立在最小的接口上。

## 三、依赖倒转（倒置）原则：

### 基本介绍：

1. 高层模块不应该依赖低层模块，二者都应该依赖其抽象；
2. 抽象不应该依赖细节，细节应该依赖抽象；
3. 依赖倒转（倒置）的中心思想是面向接口编程；
4. 依赖倒转原则是基于这样的设计理念：相对于细节的多变性，抽象的东西要稳定的多。以抽象为基础搭建的架构要比以细节为基础的架构要稳定的多。在Java中，抽象指的是接口或抽象类，细节就是具体的实现类；
5. 使用接口或抽象类的目的是制定好规范，而不涉及任何具体的操作，把展现细节的任务交给他们的实现类去完成。

### 注意事项和细节：

1. 低层模块尽量要有抽象类或接口，或者两者都有，程序稳定性更好；
2. 变量的声明类型尽量是抽象类或接口，这样我们的变量引用和实际对象间，就存在一个缓冲层，利于程序扩展和优化；
3. 继承时遵循里氏替换原则；

## 里氏替换原则：

### OO中的继承性的思考和说明：

1. 继承包含了一样一层含义：父类中凡是已经实现好的方法，实际上是在设定规范和契约，虽然它不强制要求所有的子类必须遵循这些契约，但是如果子类对这些已经实现的方法任意修改，就会对整个继承体系造成破坏；
2. 继承在给程序设计带来便利的同时，也带来了弊端。比如使用继承会给程序带来侵入性，程序的可移植性降低，增加对象间的耦合，如果一个类被其他类所继承，则当这个类需要修改时，必须考虑到所有的子类，并且父类修改后，所有涉及的功能都有可能产生故障；
3. 在编程中，如何正确的使用继承？===> 使用里氏替换原则

### 基本介绍：

1. 所有引用基类的地方必须能透明的使用其子类的对象；
2. 在使用继承时，遵循里氏替换原则，在子类中尽量不要重写父类的方法；

3）里氏替换原则告诉我们，继承实际上让两个类耦合性增强了，在适当的情况下，可以通过聚合、组合、依赖来解决问题；

## 开闭原则：

### 基本介绍：

1. 开闭原则是编程中最基础、最重要的设计原则；
2. 一个软件实体如类，模块和函数应该对扩展开放，对修改关闭。用抽象构建框架，用实现扩展细节；
3. 当软件需要变化时，尽量通过扩展软件实体的行为来实现变化，而不是通过修改已有的代码来实现变化。
4. 编程之中遵循其它原则，以及使用设计模式的目的就是遵循开闭原则；

## 迪米特原则：

### 基本介绍：

1. 一个对象应当对其它对象保持最少的了解；
2. 类与类之间关系越密切，耦合度越大；
3. 迪米特原则又叫“最少知道原则”，即一个类对自己依赖的类知道的越少越好。也就是说，对于被依赖的类不管多么复杂，都尽量将逻辑封装在类的内部。对外除了提供的public方法，不对外泄露任何信息；
4. 迪米特原则还有个更简单的定义：只与直接的朋友通信；
5. 直接的朋友：每个对象都会与其它对象有耦合关系，只要两个对象之间有耦合关系，我们就说这两个对象之间是朋友关系。耦合的方式很多，依赖、关联、组合、聚合等。其中，我们称出现成员变量、方法参数、方法返回值中的类为直接的朋友，而出现在局部变量中的类不是直接的朋友。也就是说，**陌生的类最好不要以局部变量的形式出现在类的内部**。

## 合成复用原则：

### 基本介绍：

尽量使用合成/聚合的方式，而不是使用继承。