Java 程序员应该了解的 10 个面向对象设计原则

作者：cat

原文链接：https://ld246.com/article/1358839334586

来源网站：链滴

<p>面向对象设计原则是 OOPS（Object-Oriented Programming System，面向对象的程序设计统）编程的核心，但大多数 Java 程序员追逐像
Singleton、
Decorator、
Observer这样的设计模式，而不重视面对象的分析和设计。甚至还有经验丰富的 Java 程序员没有听说过 OOPS 和
SOLID设计原则，他们根本不知道设计原则的好处也不知道如何依照这些原则来进行编程。

 众所周知，Java 编程最基本的原则就是要追求高内聚和低耦合的解决方案和代码模块设计。查 Apache 和 Sun 的开放源代码能帮助你发现其他 Java 设计原则在这些代码中的实际运用。Java Deveopment Kit 则遵循以下模式：BorderFactory 类中的
工厂模式、Runtime 类中的
单件模式。你可以通过 Joshua Bloch
《Effective Java》一书来了解更多信息。我个人偏向的一种面向对象的设计模式是 Kathy Sierra 的
Head First Design Pattern以及
Head First Object Oriented Analysis and Design。&nsp;

 虽然实际案例是学习设计原则或模式的最佳途径，但通过本文的介绍，没有接触过这些原则或在学习阶段的 Java 程序员也能够了解这 10 个面向对象的设计原则。其实每条原则都需要大量的篇才能讲清楚，但我会尽力做到言简意赅。

 原则 1：DRY（Don't repeat yourself）

 即不要写重复的代码，而是用“abstraction”类来抽象公有的东西。如果你需要多次用到一个编码值，那么可以设为公共常量；如果你要在两个以上的地方使用一个代码块，那么可以将它设为一独立的方法。SOLID 设计原则的优点是易于维护，但要注意，不要滥用，duplicate 不是针对代码，是针对功能。这意味着，即使用公共代码来验证 OrderID 和 SSN，二者也不会是相同的。使用公共码来实现两个不同的功能，其实就是近似地把这两个功能永远捆绑到了一起，如果 OrderID 改变了格式，SSN 验证代码也会中断。因此要慎用这种组合，不要随意捆绑类似但不相关的功能。 <r>

 原则 2：封装变化

 在软件领域中唯一不变的就是“Change”，因此封装你认为或猜测未来将发生变化的代码。OPS 设计模式的优点在于易于测试和维护封装的代码。如果你使用 Java 编码，可以默认私有化变量和

法，并逐步增加访问权限，比如从 private 到 protected 和 not public。有几种 Java 设计模式也使封装，比如 Factory 设计模式是封装“对象创建”，其灵活性使得之后引进新代码不会对现有的代码成影响。

 原则 3：开闭原则

 即对扩展开放，对修改关闭。这是另一种非常棒的设计原则，可以防止其他人更改已经测试好代码。理论上，可以在不修改原有的模块的基础上，扩展功能。这也是
开闭原则的宗旨。 <br

 原则 4：单一职责原则

 类被修改的几率很大，因此应该专注于单一的功能。如果你把多个功能放在同一个类中，功能间就形成了关联，改变其中一个功能，有可能中止另一个功能，这时就需要新一轮的测试来避免可能现的问题。

 原则 5：依赖注入或倒置原则

 这个设计原则的亮点在于任何被 DI 框架注入的类很容易用 mock 对象进行测试和维护，因为象创建代码集中在框架中，客户端代码也不混乱。有很多方式可以实现依赖倒置，比如像 AspectJ 等 AOP（Aspect Oriented programming）框架使用的字节码技术，或 Spring 框架使用的代理等。&bsp;

 原则 6：优先利用组合而非继承

 如果可能的话，优先利用组合而不是继承。一些人可能会质疑，但我发现，组合比继承灵活得。组合允许在运行期间通过设置类的属性来改变类的行为，也可以通过使用接口来组合一个类，它提了更高的灵活性，并可以随时实现。
《Effective Java》也推荐此原则。

 原则 7：里氏代换原则（LSP）

 根据该原则，子类必须能够替换掉它们的基类，也就是说使用基类的方法或函数能够顺利地引子类对象。LSP 原则与单一职责原则和接口分离原则密切相关，如果一个类比子类具备更多功能，很可能某些功能会失效，这就违反了 LSP 原则。为了遵循该设计原则，派生类或子类必须增强功能。&bsp;

 原则 8：接口分离原则

 采用多个与特定客户类有关的接口比采用一个通用的涵盖多个业务方法的接口要好。设计接口棘手，因为一旦释放接口，你就无法在不中断执行的情况下改变它。在 Java 中，该原则的另一个优在于，在任何类使用接口之前，接口不利于实现所有的方法，所以单一的功能意味着更少的实现方法

 原则 9：针对接口编程，而不是针对实现编程

 该原则可以使代码更加灵活，以便可以在任何接口实现中使用。因此，在 Java 中最好使用变接口类型、方法返回类型、方法参数类型等。《Effective Java》 和《head first design pattern》中也有提到。

 原则 10：委托原则

 该原则最典型的例子是 Java 中的 equals() 和 hashCode() 方法。为了平等地比较两个对象，们用类本身而不是客户端类来做比较。这个设计原则的好处是没有重复的代码，而且很容易对其进行改。

 总之，希望这些面向对象的设计原则能帮助你写出更灵活更好的代码。理论是第一步，更重要是需要开发者在实践中去运用和体会。

 英文原文：
<a href="https://ld246.com/forward?goto=http%3A%2F%2Fjavarevisited.blogspot.com%2F212%2F03%2F10-object-oriented-design-principles.html" target="_blank" rel="nofollow ugc"10 Object Oriented Design principles Java programmer should know</p>