第二章：面向对象设计原则

H1

> 概述：在面向对象软件系统设计时， 要充分考虑软件系统的可维护性和可复用性。本章将介绍7 个重要的面向对象设计原则：单一职责原则、开闭原则、里氏代换原则、依赖倒转原则、接口隔离原则、合成复用原则和迪米特法则。

一、单一职责原则

定义: 一个对象应该只包含单一的职责， 并且该职责被完整地封装在一个类中。

H2

在软件系统中， 一个类承担的职责越多， 它被复用的可能性就越小， 而且一个类承担的职责过多， 相当于将这些职责耦合在一起， 当其中一个职责变化时可能会影响其他职责的运作， 因此要将这些职责进行分离， 将不同的职责封装在不同的类中，即将不同的变化原因封装在不同的类中， 如果多个职责总是同时发生改变则可将它们封装在同一类中。
单一职责原则是实现高内聚、低耦合的指导方针， 它是最简单但又最难运用的原则， 需
要设计人员发现类的不同职责并将其分离， 而发现类的多重职责需要设计人员具有较强的
分析设计能力和相关实践经验。

> 注：对于这样的一个原则简单理解起来就是我们在设计软件系统时需要提前给各个类规定好其在在系统中的主要功能，不能将所有功能杂糅在一个类中，这也就是解耦，降低各个类的依赖性。

二、开闭原则

定义: 对拓展开放，对修改关闭

H2

> 注：当我们完成对一个系统的开发，整个系统都是井然有序的，类与类之间都有很强的依赖性，如果后期进行修改，那么将会是一件很麻烦的事。所以需要限制关闭修改。
> 那在进行修改的话， Java 中会采用的方法是：前期提供接口、抽象类等机制，定义系统的抽象层，修改的话只要添加具体的实现类，拓展只需要对接口、抽象类进行实现就可以了。

三、里氏代换原则

定义： 所有引用基类的地方必须能透明地使用其子类的对象。

H2 书上是这么解释的：

> 使用子类对象来替换父类对象。将父类设计为抽象类或者接口， 让子类继承父类或实现父接口， 并实现在父类中声明法， 在运行时子类实例替换父类实例， 可以很方便地扩展系统的功能， 无须修改原有子类的代码， 增加新的功能可以通过增加一个新的子类来实现。

四、依赖倒转原则

定义： 高层模块不应该依赖低层模块， 它们都应该依赖抽象。抽象不应该依赖于细节， 细节应该依赖于抽象。

H2

书上解释为：

> 依赖倒转原则要求在程序代码中传递参数时或在关联关系中尽量引用层次高的抽象层类，即**使用接口和抽象类进行变量类型声明、参数类型声明、方法返回类型声明， 以及数据类型的转换等**， 而不要用具体类来做这些事情。为了确保该原则的应用， 一个具体类应当只实现接口或抽象类中声明过的方法， 而不要给出多余的方法， 否则将无法调用到在子类中增加的新方法。

理解为：

> 该原则的**使用场景**上面已经介绍了：变量类型声明、参数类型声明、方法返回类型声明， 以及数据类型的转换等。
> 简单来说就是当函数传参、返回值、变量声明（new之前的变量名）……都尽可能使用父类

五、接口隔离原则

定义：客户端不应该隔离那些它不需要的接口

H2 理解为：

> 我们知道，但类去实现接口时，若接口太大，即子类需要实现的函数也将越多，所以该原则是将大接口分为若干个小接口，然后让要具体实现类去实现其中的一个小接口，这样避免了实现类要实现很多接口的尴尬。

六、合成复用原则

定义： 优先使用对象组合， 而不是通过继承来达到复用的目的。

H2 理解为：

> 该原则出现的场景为：我们想要使用某个类时，一般的方式可以分为这几种：
>
> ```
> 1. 创建类的对象，然后调用方法
> 2. 继承该类，然后调用方法
> ```
>
> 书上这样解释继承的不好之处：滥用继承反而会增加系统构建和维护的难度以及系统的复杂度
>
> 所以不能随便使用继承，一般使用的创建对象，调用方法这种方式。

七、迪米特法则

定义： 每一个软件单位对其他单位都只有最少的知识， 而且局限于那些与本单位密切相关的软件单位。

H2 解释：

> 要求一个软件实体应当尽可能少地与其他实体发生相互作用。如果一个系统符合迪米特法则， 那么**当某一个模块发生修改时就会尽量少地影响其他模块， 扩展会相对容易**， 这是对软件实体之间通信的限制， 迪米特法则要求限制软件实体之间通信的宽度和深度。应用迪米特法则可**降低系统的耦合度， 使类与类之间保持松散的耦合关系。**

理解：

> 该法则的使用主要时希望模块与模块之间尽可能地减少交互。如果要交互地话，可以使用中介者模式，添加中间类在不同模块之间传递信息。