一、开闭原则：

这一条放在第一位来理解，它的含义是对扩展开放，对修改关闭。即我们要为未来可预见的修改留有扩展点，在以后修改时增加扩展即可。

如支付方式要留有扩展点，以便未来增加微信或支付宝支付。

如缓存方法要支持扩展，以便未来使用redis缓存。

“可预见的修改”是个头疼的问题，因为未来可能要修改的点太多，但只有一小部分才会被真正修改，其余“可能要修改的点”都会作废

在我众多的项目经历中，这个原则适用于框架中或者公共组件中，而对于其他的代码，我们不妨在需要时再增加扩展点

要记住，“需要时再增加扩展点”比“现在增加了但用不到”要好很多

二、里氏替换选择：

此原则的含义是子类可以在任何地方替换它的父类。

如B继承A，那么任何一个需要A的地方，我传给它B代码也要能够正确执行

里氏替换原则的潜台词是：尽量使用精准的抽象类或者接口。

三、单一职责原则：

单一职责的含义是：类的职责单一，引起类变化的原因单一。如果一个类做的事情太多，在组合的时候，必然会产生不必要的方法出现，这实际上是一种污染。

单一职责的潜台词是：拆分到最小单位，解决复用和组合问题。

四、接口隔离原则：

接口隔离原则可以说是单一职责的必要手段，它的含义是尽量使用职能单一的接口，而不使用职能复杂、全面的接口。

接口隔离原则的潜台词是：拆分，从接口开始。

示例，客户端1使用类A的一部分接口，客户端B使用类A的另一部分接口，那么A应该继承2个interface，如比较常见的一个类A实现了IReadXXX和IWrithXXX接口

五、依赖倒置原则：

想要理解依赖倒置原则，必须先理解传统的解决方案。面相对象的初期的程序，被调用者依赖于调用者。也就是调用者决定被调用者有什么方法，有什么样的实现方式，这种结构在需求变更的时候，会付出很大的代价，甚至推翻重写。

依赖倒置原则就是要求调用者和被调用者都依赖抽象，这样两者没有直接的关联和接触，在变动的时候，一方的变动不会影响另一方的变动。

依赖倒置的潜台词是：面向抽象编程，解耦调用和被调用者。

六、迪米特原则：

迪米特原则要求尽量的封装，尽量的独立，尽量的使用低级别的访问修饰符。这是封装特性的典型体现。

一个类如果暴露太多私用的方法和字段，会让调用者很茫然。并且会给类造成不必要的判断代码。所以，我们使用尽量低的访问修饰符，让外界不知道我们的内部。

另外，迪米特原则要求类之间的直接联系尽量的少，两个类的访问，通过第三个中介类来实现。

迪米特原则的潜台词是：不和陌生人说话，有事去中介。

七、组合/聚合复用原则：

此原则的含义是，如果只是达到代码复用的目的，尽量使用组合与聚合，而不是继承。

继承的耦合性更大，比如一个父类后来添加实现一个接口或者去掉一个接口，那子类可能会遭到毁灭性的编译错误，但如果只是组合聚合，只是引用类的方法，就不会有这种巨大的风险，同时也实现了复用。

组合聚合复用原则的潜台词是：我只是用你的方法，我们不一定是同类。