**1.单一原则**（Single Responsibility Principle）：一个类或者一个方法只负责一项职责，尽量做到类的只有一个行为原因引起变化；

　　a、业务对象（BO business object）、业务逻辑（BL business logic）拆分；

**2.里氏替换原则**（LSP liskov substitution principle）：子类可以扩展父类的功能，但不能改变原有父类的功能；（本质其实就是c++的多态）

　　（目的：增强程序的健壮性）实际项目中，每个子类对应不同的业务含义，使父类作为参数，传递不同的子类完成不同的业务逻辑。

**3.依赖倒置原则（dependence inversion principle）**：面向接口编程；（通过接口作为参数实现应用场景）

　　抽象就是接口或者抽象类，细节就是实现类

　　含义：

　　　　上层模块不应该依赖下层模块，两者应依赖其抽象；

　　　　抽象不应该依赖细节，细节应该依赖抽象；

通俗点就是说变量或者传参数，尽量使用抽象类，或者接口；

【接口负责定义public属性和方法，并且申明与其他对象依赖关系，抽象类负责公共构造部分的实现，实现类准确的实现业务逻辑】

**4.接口隔离（interface segregation principle）**：建立单一接口；（扩展为类也是一种接口，一切皆接口）

　　　定义：

　　　　a.客户端不应该依赖它不需要的接口；

　　　　b.类之间依赖关系应该建立在最小的接口上；

简单理解：复杂的接口，根据业务拆分成多个简单接口；（对于有些业务的拆分多看看适配器的应用）

　【接口的设计粒度越小，系统越灵活，但是灵活的同时结构复杂性提高，开发难度也会变大，维护性降低】

**5.迪米特原则（law of demeter LOD）**：最少知道原则，尽量降低类与类之间的耦合；

一个对象应该对其他对象有最少的了解

**6.开闭原则（open closed principle）：**用抽象构建架构，用实现扩展原则；