## OO中的设计原则

SOLID原则：

S 单一职责原则（Single Responsibility Principle）

O 开放关闭原则（Open Closed Principle）

L Liskov替换原则（Liskov Substitution Principle）

I 接口分离原则（Interface Separation Principle）

D 依赖倒置原则（Dependency Inversion Principle）

通过应用这些原则，能够提高代码适应变化的能力。要在正确的时机和场合使用，切勿过度使用。

### [单一职责原则](http://www.objectmentor.com/resources/articles/srp.pdf)

方法、类和模块应该只具有一种职责，当需要承担其它职责时，应该把这些职责委托给其它的单职责类。



### [开放关闭原则](http://www.objectmentor.com/resources/articles/ocp.pdf)

对扩展开发：模块的行为是可扩展的。当需求改变时，我们可以对其模块进行扩展，使它能能够满足新需求变更的新行为。

对修改关闭：在对模块行为进行扩展时，不用改动该模块其它的代码。

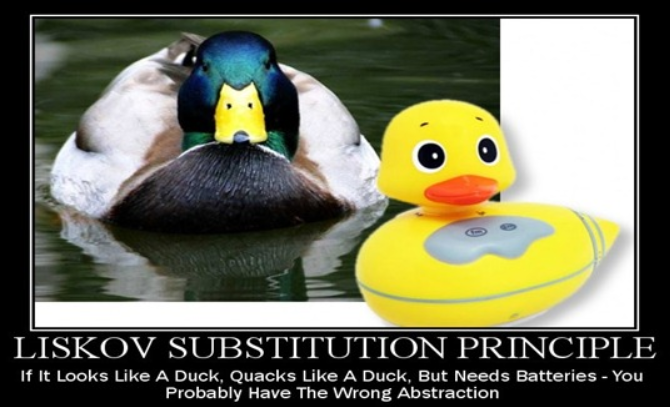
扩展点：提供应对需求改变的入口。

防止变异：识别可预见的变化点并围绕它们创建一个稳定的接口。



### Liskov替换原则

指导创建继承层次的结构，是对SRP和OCP的辅助增强。



### 接口分离原则

### 依赖倒置原则

## OO中的设计模式

## 常见的分层架构

### 三层架构

DAL: 数据访问层，主要与数据库操作相关。

IDAL：数据访问接口层，主要用于抽象数据库操作。

BLL：业务逻辑层，主要用于具体业务功能的实现。

IBLL：业务逻辑接口层，主要用于抽象业务功能。

Model: 数据实体，也可以是ViewModel。

## 引用

1. <https://www.tomdalling.com/blog/software-design/solid-class-design-the-single-responsibility-principle/>
2. <https://www.tomdalling.com/blog/software-design/solid-class-design-the-interface-segregation-principle/>