设计模式之美总结

面向对象

- 封装、抽象、继承、多态
- 面向对象编程 VS 面向过程编程
- 面向对象分析、设计、编程
- 接口 VS 抽象类
- 基于接口而非实现编程
- 多用组合少用继承
- 贫血模型和充血模型

设计原则

- SOLID原则–SRP单一职责原则
- SOLID原则–OCP开闭原则
- SOLID原则–LSP里式替换原则
- SOLID原则–DIP依赖倒置原则

- DRY原则、KISS原则、YAGNI原则、LOD法则

单一职责原则

单一职责原则是类职责划分的重要参考依据，是保证代码”高内聚“的有效手段，是面向对象设计前两步（划分职责并识别出有哪些类、定义类及其属性和方法）的主要指导原则。单一职责原则的难点在于，对代码职责是否足够单一的判定。这要根据具体的场景来具体分析。同一个类的设计，在不同的场景下，对职责是否单一的判定，可能是不同的。

开闭原则

开闭原则是保证代码可扩展性的重要指导原则，是对代码扩展性的具体解读。很多设计模式诞生的初衷都是为了提高代码的扩展性，都是以满足开闭原则为设计目的的。实际上，尽管开闭原则描述为对扩展开放、对修改关闭，但也并不是说杜绝一切代码修改，正确的理解是以最小化修改代价来完成新功能的添加。实际上，在平时的开发中，我们要时刻思考，目前的设计在以后应对新功能扩展的时候，是否能做到不需要大的代码修改（比如调整代码结构）就能完成。

依赖倒置原则

依赖倒置原则主要用来指导框架层面的设计。高层模块不依赖低层模块，它们共同依赖同一个抽象。深挖一下的话，我们要把它跟控制反转、依赖注入、依赖注入框架做区分。实际上，比依赖倒置原则更加常用的是依赖注入。它用来指导如何编写可测试性代码，换句话说，编写可测试代码的诀窍就是应用依赖注入。

YAGNI 原则

YAGNI 原则表示暂时不需要的就不要做，KISS 原则表示要做就要尽量保持简单。

DRY 原则

DRY 原则主要是提醒你不要写重复的代码。

LOD 原则

LOD 原则又叫最小知道原则，不该有直接依赖关系的类之间，不要有依赖；有依赖关系的类之间，尽量只依赖必要的接口。

代码重构

- 目的、对象、时机、方法
- 单元测试和代码可测试式
- 大重构（大规模高层次）
- 小重构（小规模低层次）

设计模式

经典的设计模式有 23 种，分三种类型：创建型、结构型和行为型。其中，创建型设计模式主要解决“对象的创建”问题，结构型设计模式主要解决“类或对象的组合”问题，行为型设计模式主要解决“类或对象之间的交互”问题。

常用的并不多，主要有：单例、工厂、建造者、代理、装饰器、适配器、观察者、模板、策略、职责链、迭代器这 11 种。

创建型

主要解决“对象的创建”问题

常用	不常用
单例模式	原型模式
工厂模式（工厂方法和抽象工厂）	
建造者模式	

结构型

主要解决“类或对象的组合”问题

常用	不常用

代理模式	门面模式
桥接模式	组合模式
装饰者模式	享元模式
适配器模式	

行为型

类或对象之间的交互

常用	**不常用**
观察者模式	访问者模式
模板模式	备忘录模式
策略模式	命令模式
职责链模式	解释器模式
迭代器模式	中介模式
状态模式	