设计模式

一、设计模式七大原则

设计模式原则，其实就是程序员在编程时，应当遵守的原则，也是各种设计模式的基础(即：设计模式为什么这样设计的依据)

设计模式常用的七大原则有:

1) 单一职责原则

2) 接口隔离原则

3) 依赖倒转(倒置)原则

4) 里氏替换原则

5) 开闭原则

6) 迪米特法则

7) 合成复用原则

1. **单一职责原则**

1.1基本介绍

对类来说的，即一个类应该只负责一项职责。如类A负责两个不同职责：职责1，职责2。

当职责1需求变更而改变A时，可能造成职责2执行错误，所以需要将类A的粒度分解为

A1，A2

1.2 案例见singleResponsibility。

1.3 注意事项和细节

1) 降低类的复杂度，一个类只负责一项职责。

2) 提高类的可读性，可维护性

3) 降低变更引起的风险

4) 通常情况下，**我们应当遵守单一职责原则**，只有逻辑足够简单，才可以在代码级违

反单一职责原则；只有类中方法数量足够少，可以在方法级别保持单一职责原则

1. **接口隔离原则**
   1. 基本介绍

客户端不应该依赖它不需要的接口，即一个类对另一个类的依赖应该建立在最小的接口上。

* 1. 案例见 segregation

1.3 使用接口隔离原则改进

1) 类A通过接口Interface1依赖类B，类C通过接口Interface1依赖类D，如果接口

Interface1对于类A和类C来说不是最小接口，那么类B和类D必须去实现他们不

需要的方法

2) 将接口Interface1拆分为独立的几个接口，类A和类C分别与他们需要的接口建立

依赖关系。也就是采用接口隔离原则

3) 接口Interface1中出现的方法，根据实际情况拆分为三个接口

**3 依赖倒转(倒置)原则**

**1.1基本介绍**

1) 高层模块不应该依赖低层模块，二者都应该依赖其抽象

2) 抽象不应该依赖细节，细节应该依赖抽象

3) 依赖倒转(倒置)的中心思想是面向接口编程

4) 依赖倒转原则是基于这样的设计理念：相对于细节的多变性，抽象的东西要稳定的

多。以抽象为基础搭建的架构比以细节为基础的架构要稳定的多。在java中，抽象

指的是接口或抽象类，细节就是具体的实现类

5) 使用接口或抽象类的目的是制定好规范，而不涉及任何具体的操作，把展现细节的

任务交给他们的实现类去完成

1.2案例 inversion

* 1. 注意事项和细节

1) 低层模块尽量都要有抽象类或接口，或者两者都有，程序稳定性更好.

2) 变量的声明类型尽量是抽象类或接口, 这样我们的变量引用和实际对象间，就存在

一个缓冲层，利于程序扩展和优化

1. 继承时遵循里氏替换原则

4 里氏替换原则

* 1. 基本介绍

1) 里氏替换原则(**Liskov Substitution Principle**)在1988年，由麻省理工学院的姓里的女士提出的。

2) 如果对每个类型为T1的对象o1，都有类型为T2的对象o2，使得以T1定义的所有程序

P在所有的对象o1都代换成o2时，程序P的行为没有发生变化，那么类型T2是类型T1

的子类型。换句话说，所有引用基类的地方必须能透明地使用其子类的对象。

3) 在使用继承时，遵循里氏替换原则，在子类中**尽量**不要重写父类的方法

4) 里氏替换原则告诉我们，继承实际上让两个类耦合性增强了，在适当的情况下，可

以通过聚合，组合，依赖 来解决问题。

1.2 案例 liskov

5 开闭原则

* 1. 基本介绍

1) 开闭原则（Open Closed Principle）是编程中最基础、最重要的设计原则

2) 一个软件实体如类，模块和函数应该对扩展开放(对提供方)，对修改关闭(对使用 方)。用抽象构建框架，用实现扩展细节。

3) 当软件需要变化时，尽量**通过扩展**软件实体的行为来实现变化，而**不是通过修改**已有的代码来实现变化。

4) 编程中遵循其它原则，以及使用设计模式的目的就是遵循**开闭原则**。

1.2 案例 openAndClose

6) 迪米特法则

1.1 基本介绍

1) 一个对象应该对其他对象保持最少的了解

2) 类与类关系越密切，耦合度越大

3) 迪米特法则(**Demeter Principle**)又叫**最少知道原则**，即一个类对自己依赖的类知道的越少越好。也就是说，对于被依赖的类不管多么复杂，都尽量将逻辑封装在类的内部。对外除了提供的public 方法，不对外泄露任何信息

4) 迪米特法则还有个更简单的定义：只与直接的朋友通信

5) 直接的朋友：每个对象都会与其他对象有耦合关系，只要两个对象之间有耦合关系，我们就说这两个对象之间是朋友关系。耦合的方式很多，依赖，关联，组合，聚合等。其中我们称出现成员变量，方法参数，方法返回值中的类为直接的朋友，而出现在局部变量中的类不是直接的朋友。也就是说，陌生的类最好不要以局部变量

的形式出现在类的内部。

* 1. 案例 demeterDemo
  2. 迪米特法则注意事项和细节

1) 迪米特法则的核心是降低类之间的耦合

2) 但是注意：由于每个类都减少了不必要的依赖，因此迪米特法则只是要求降低

类间(对象间)耦合关系， 并不是要求完全没有依赖关系

7) 合成复用原则

1.1 基本介绍

原则是尽量使用合成/聚合的方式，而不是使用继承