Java设计模式

1. 开闭原则：对扩展开放对修改关闭。理解：开发新需求使用其扩展功能，而不是修改其源代码完成。
2. 里氏替换原则: 子类可以扩展父类功能，但不能修改父类原有功能。 如果修改其父类方法可能造成复用性差，运行多态比较频繁时程序抱错几率较大。

修正方法为：取消继承关系，重新设计

1. 依赖倒置原则

是实现开闭原则途径之一,他降低了客户与实现中间的耦合。

作用：

1. 降低系统件的耦合性
2. 提升系统的运行稳定性
3. 减少并发引起的风险
4. 提升系统的可读性和可维护性

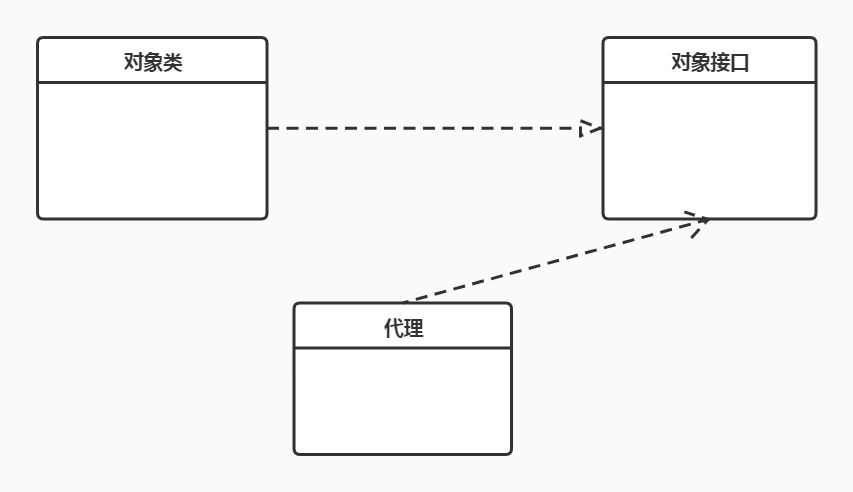
实现方法

1. 每个类提供接口或抽象类，或者二者具备
2. 变量声明尽量是接口或抽象类
3. 任何类不应该从具体类派生
4. 使用里氏替换原则
5. 依赖倒置原则
6. 单一职责
7. 代理模式 通过代理对象访问被代理对象的方法

角色

1. 被代理者
2. 被代理者接口
3. 代理

类图

****

1. 静态代理 代理与被代理实现相同接口或继承父类
2. JDK实现动态代理，机制为被代理对象实现InvocationHandler接口
3. 利用CGlib操作字节码在运行时给被代理对象生成子类，让其产生继承关系便可以调用被代理类方法，不支持final 方法
4. 单例模式
5. 懒汉模式 类加载时没有生成单例，但是在调用初始化方法后会生成。

缺点：在多线程下需要同步，消耗资源，影响性能。

1. 饿汉模式 在类加载时候创建单例
2. 原形模式 用一个已经创建的对象通过复制原对象来创建新的对象，java中Cloneable进行浅拷贝

深拷贝与浅拷贝区别

深拷贝会拷贝所有对象的属性，并拷贝属性指向的内存。速度慢，花销大。浅拷贝复制对象的引用，拷贝的对象和原对象之间共享。

类似指针与内存关系，深拷贝拷贝对账内存到新对象，而浅拷贝拷贝对象的指针引用。

角色

1. 抽象原型类：原型对象必须实现的接口
2. 具体原型类：实现抽象原型类clone方法，他是可复制的
3. 访问类：具体原型类的clone中方法创建新对象。