**设计模式笔记**

1. **设计模式入门**

OO基础:抽象、封装、多态、继承

OO原则:

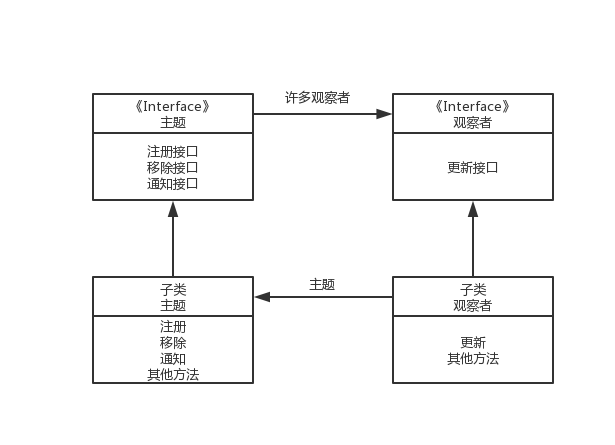
1. 封装变化
2. 多用组合,少用继承
3. 针对接口编程而非实现

OO模式之策略模式：

定义算法族,分别封装起来,让其之间可以相互替换,次模式让算法的变化独立于使用算法的客户

1. **观察者模式**

**性质:**观察者模式的本质就是出版者+订阅者。或者说,主题(出版者)和观察者(订阅者)。

**OO模式之观察者模式：**

定义了对象之间的一对多依赖，这样一来，当一个对象改变状态时，它的所有依赖都会收到通知且自动更新。

**观察者模式的类图：**

**松耦合：**

当两个对象之间松耦合，他们依然可以交互，但是不太清楚彼此的细节

**设计原则：**

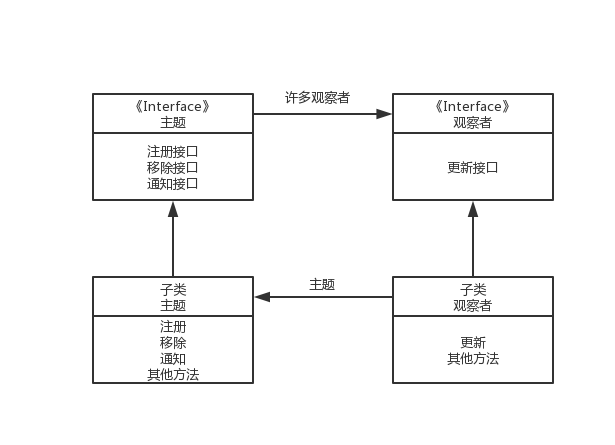
为了交互对象间的松耦合而努力

松耦合的设计之所以能够建立富有弹性的OO系统以应对变化，是因为对象之间的依赖降到了最低

1. **装饰者模式**

**设计原则:**

类应当对扩展开放，对修改关闭

**OO模式之装饰者模式：**

动态地将责任附加到对象上,若要扩展功能，装饰者提供比继承更有弹性的替代方案

**装饰者模式的类图:**

实际上装饰者模式利用了组合来替代继承

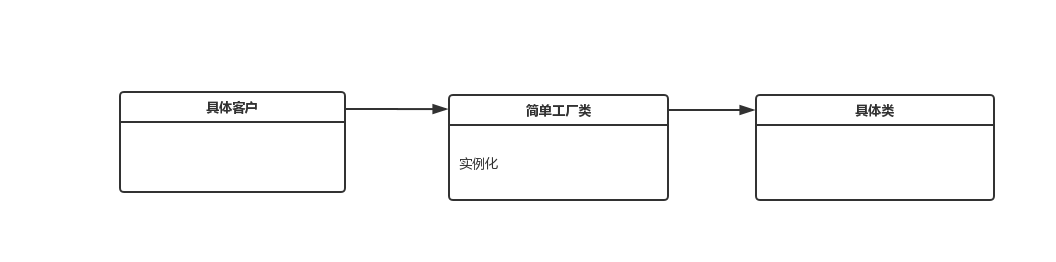
**OO原则(扩展)：**

1. 封装变化
2. 多用组合、少用继承
3. 针对接口编程而非实现
4. 为交互对象之间松耦合而努力
5. 对扩展开放、对修改关闭

**四、工厂模式**

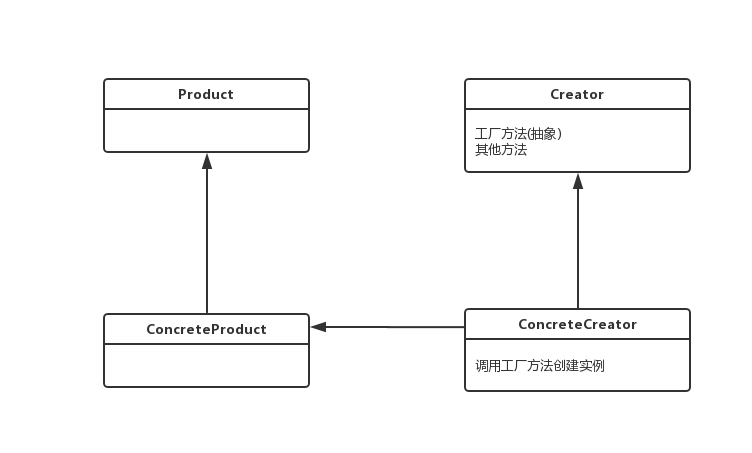
**1.简单工厂**

简单工厂并非一个具体的设计模式，反而更像一种编程习惯。



其本质是将实例化的过程用一个单独的类封装起来,这个类也被称作简单工厂类,如下图所示

**2.OO模式之工厂方法模式**

工厂方法模式定义了一个创建对象的接口,但由子类决定要实例化的类到底是哪一个,工厂方法让类把实例化推迟到子类

如上图所示,Creator是一个类,其实现了所有操纵产品的方法,但不实现工厂方法

ConcreteCreator实现了工厂方法,以创建实际的产品

Product是一个抽象产品类,作为所有产品实现的共同接口

ConcreteProduct代表具体的产品

**3.OO原则之依赖倒置原则**

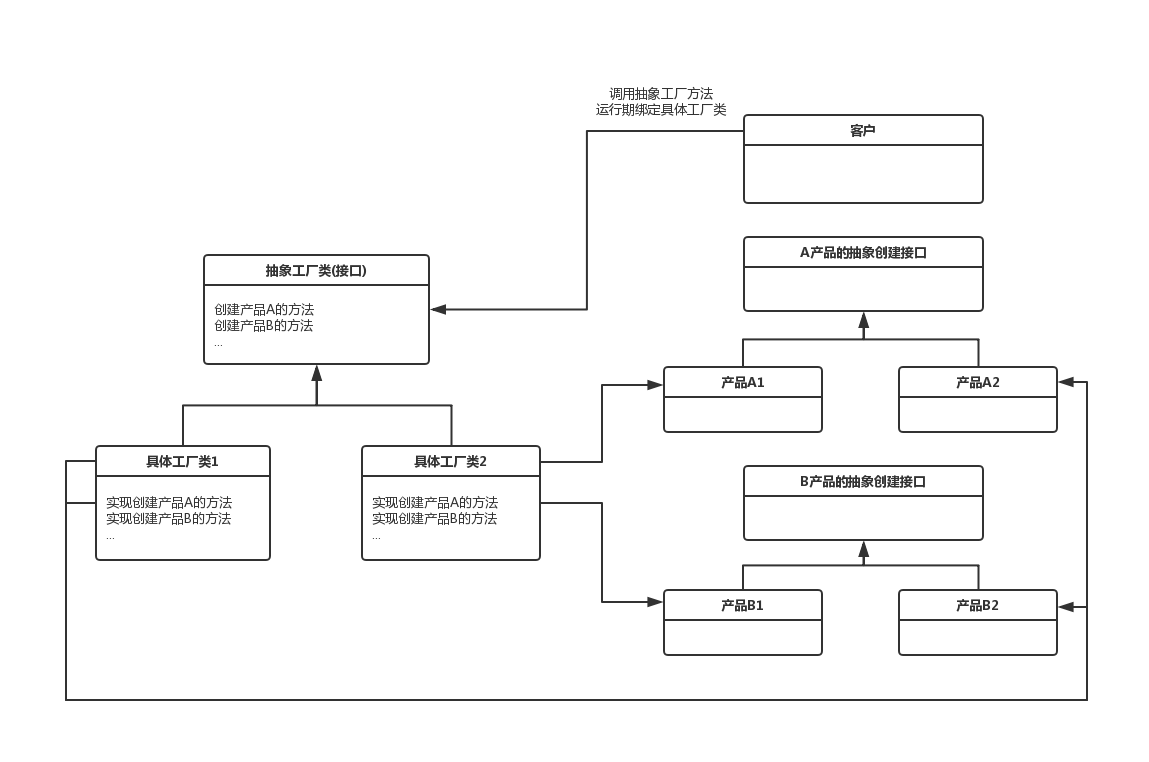
要依赖抽象而不要依赖具体类

在应用工厂方法后,高层组件和低层组件都依赖于抽象方法,符合依赖倒置原则.想要遵循该原则,工厂方法并非唯一的技巧,却是最有威力的技巧之一！

**4.OO模式之抽象工厂模式**

抽象工厂模式提供一个接口,用于创建相关或者依赖对象的家族,而不用明确指定具体类

抽象工厂允许客户使用抽象的接口来创建一组相关的产品,而不需要知道或关心实际产出的产品是什么。这样一来客户就从具体的产品中被解耦。



产品等级结构:即产品的继承结构(比如某一品牌汽车下属的所有汽车类型)

产品族:在抽象工厂模式中,产品族指的是由一个工厂生产的位于不同产品等级结构中的一组产品

**工厂方法和抽象工厂方法的共同点与区别：**

1. 所有工厂都是用来封装对象的创建
2. 工厂方法使用继承(is - a),把对象的创建委托给子类,子类实现工厂方法来创建对象
3. 抽象工厂使用组合(has - a),对象的创建被实现在工厂接口所暴露出来的方法中
4. 工厂方法允许将实例化推迟到子类
5. 遵循依赖倒置原则
6. 抽象工厂创建对象家族(有多个抽象产品类),而工厂方法(只有唯一抽象产品类)仅仅创建单一对象
7. 实际上抽象工厂中每一类产品的创建方法都与工厂方法相同(即抽象类只提供创建接口,子类负责具体创建实例)
8. 抽象工厂方法只是纯粹使用具体工厂方法来创建产品
9. 工厂方法中抽象创建者所实现的代码通常会用到子类创建的具体类型
10. 当增加产品等级结构时,抽象工厂方法不符合开闭原则
11. 工厂方法针对一个产品等级结构,抽象工厂方法增对多个产品等级结构