设计模式总结

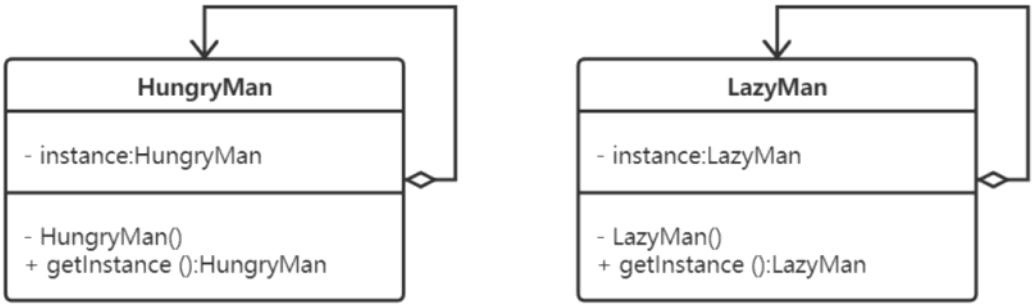
2023年11月30日 11:34

一、单例模式

单例模式是指在内存中只会创建且仅创建一次对象的设计模式。在程序中多次使用同一个对象且作用相同时，为了防止频繁地创建对象使得内存飙升，单例模式可以让程序仅在内存中创建一个对象，让所有需要调用的地方都共享这一单例对象。

>

设计模型之单例模式(含UML完整实例) 单例模式画图-CSDN博客



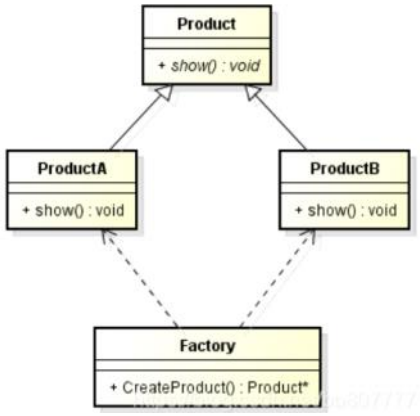
应用场景：如计算机要打开任务管理器，只需打开一个。

二、工厂模式

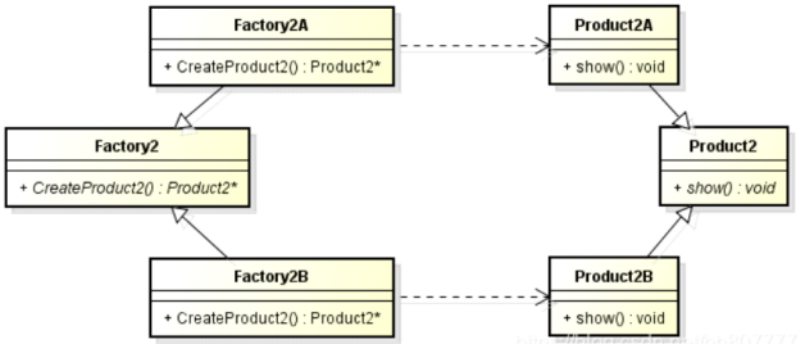
创建对象的类是工厂，创建的对象就是产品，使用工厂生产产品，只需用工厂提供的接口生产产品即可，不用关心具体的生产过程。

工厂模式分为三种：①简单工厂模式 ②工厂方法模式 ③抽象工厂模式

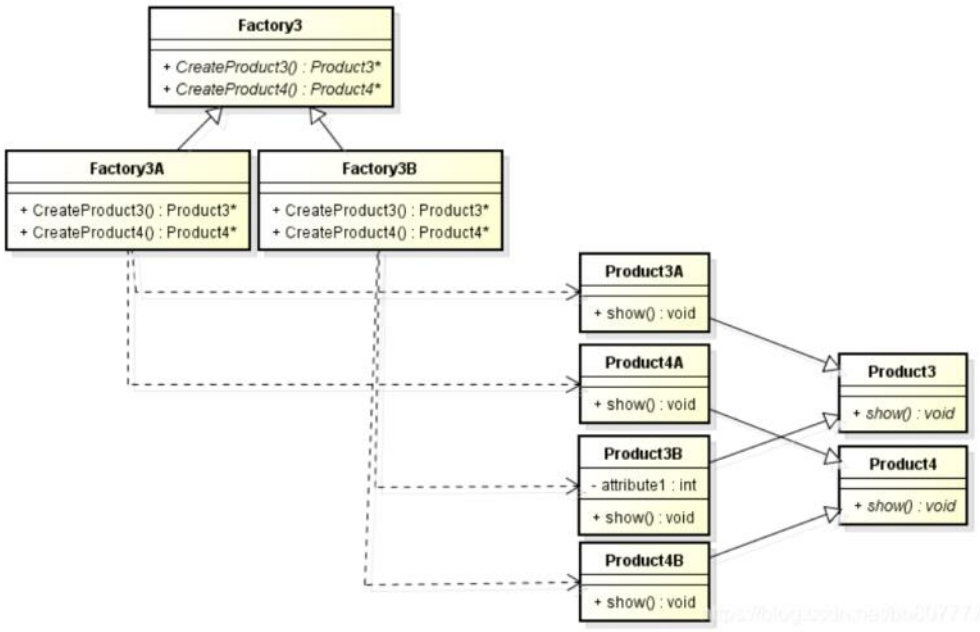
①简单工厂模式：一个工厂生产多种产品。



②工厂方法模式：多个工厂，每个工厂只能生产一种产品。工厂、产品都是抽象类，用子类工厂生产。



③抽象工厂模式：多个工厂，每个工厂可以生产多种产品。工厂、产品都是抽象类，用子类工厂生产，每个子类工厂可以生产多种产品，形成多产品的组合。



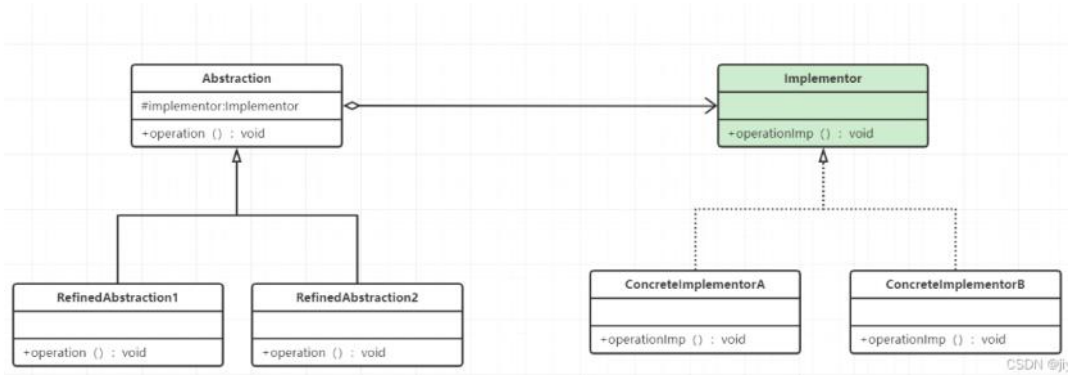
应用场景：

- 编程中，在一个 A 类中通过 new 的方式实例化了类 B，那么 A 类和 B 类之间就存在关联（耦合）
- 后期因为需要修改了 B 类的代码和使用方式，比如构造函数中传入参数，那么 A 类也要跟着修改，一个类的依赖可能影响不大，但若有多个类依赖了 B 类，那么这个工作量将会相当的大，容易出现修改错误，也会产生很多的重复代码，这无疑是件非常痛苦的事；
- 这种情况下，就需要将创建实例的工作从调用方（A类）中分离，与调用方解耦，也就是使用工厂方法创建实例的工作封装起来（减少代码重复），由工厂管理对象的创建逻辑，调用方不需要知道具体的创建过程，只管使用，而降低调用者因为创建逻辑导致的错误；

三、桥接模式

桥接模式就是把事物和其具体实现分开，使他们可以各自独立的变化。

桥接的用意是：将抽象化与实现化解耦，使得二者可以独立变化，像我们常用的JDBC桥DriverManager一样，JDBC进行连接数据库的时候，在各个数据库之间进行切换，基本不需要动太多的代码，甚至丝毫不用动，原因就是JDBC提供统一接口，每个数据库提供各自的实现，用一个叫做数据库驱动的程序来桥接就行了。



应用场景：

如用曲线、直线、虚线画圆形、方形、三角形；有9种组合，用桥接将这两大类进行组合。

四、策略模式

策略模式（Strategy Pattern），指的是定义一系列算法，并将这些算法封装到具有公共接口的一系列策略类中，使得它们可以动态自由切换。策略模式的本质是:**分离算法，选择实现**。

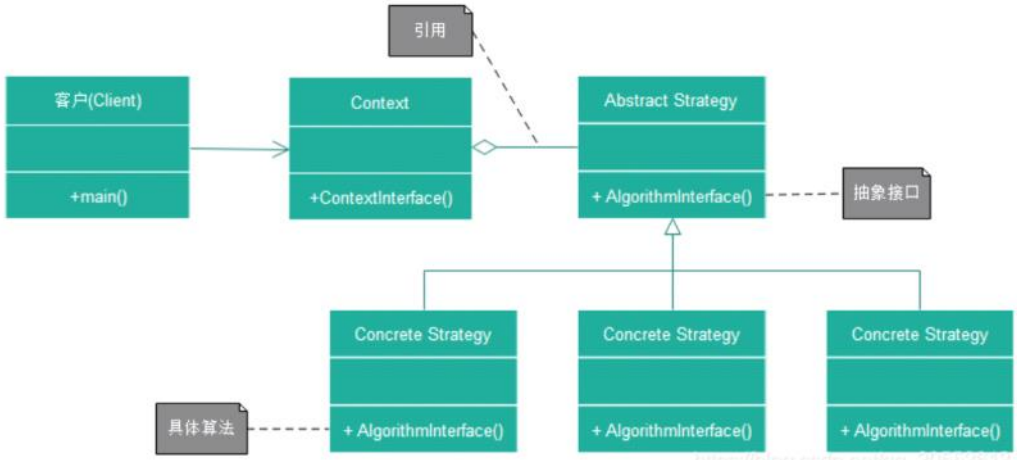
策略模式由环境角色（Context）、抽象策略（Abstract Strategy）、具体策略（Concrete Strategy）、客户（Client）i四个要素组成。

环境角色（Context）， 持有一个对Abstract Strategy的引用，最终由客户端调用

抽象策略（Abstract Strategy）,声明一个公共抽象接口，由不同算法类实现该接口；通过该接口，Context可以调用不同的算法

具体策略（Concrete Strategy）， 继承Abstract Strategy类，并实现抽象接口，提供具体算法

客户（Client），客户通过调用Context调用策略算法，严格来说客户不属于策略模式的一部分



应用场景：

作业做的用客户端选取不同支付方式。