工厂模式之工厂方法

作者：leap
原文链接：https://ld246.com/article/1501597139024
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

工厂模式之工厂方法

前言

在工厂模式之简单工厂中介绍了什么是工厂模式以及为什么要使用工厂模式，最后给出了简单工厂的个示例，解决了在面向对象设计中对象创建与对象使用职责耦合的问题，但是新的问题又来了:**当系统需要引进新的实现时，由于静态工厂方法通过传入不同参数的方式来返回不同的实现，这必定要修改厂类的代码，违反了开闭原则。如何在不修改现有代码的前提下增加新的实现？工厂方法模式应运而生。**

工厂方法模式

既然一个工厂满足不了需求，那么就来多个工厂吧，这样一来就不用修改工厂类代码了。但是这些工应该具有一致的行为:创建对象，所以给它们一个共同的父类或者接口，这里我们定义一个接口。看下的示例：

```
// 产品接口
interface Product {
    ...
}

//抽象工厂接口
interface Factory {
    Product getProduct();
}
```

定义一个产品接口用于表示产品的抽象层次。它的实现分别为ProductA，ProductB...,在上篇文章中用A，B，C来表示类不是很好，不利于理解。对于工厂来说首先想到的是它能生产产品。

在简单工厂中把类的创建逻辑都放在一个方法中，现在我们来把它分散在不同的工厂实现中，就像这：

```
class FactoryA implements Factory {
    Product getProduct(){
        return new ProductA();
    }
}

class FactoryB implements Factory {
    Product getProduct(){
        return new ProductB();
    }
}
```

然后使用它

```
class Main {
    public static void main(String args[]){
        Factory factoryA = new FactoryA();
        Product productA = FactoryA.getProduct();
    }
}
```

这个时候如果我们要增加新的具体产品，只需要实现好新的产品，在新的工厂类中生产它就行了。不要修改原有代码，符合开闭原则。

思考

新的问题又来了，如果新产品不断增多，工厂类的数量也随之增加。导致系统内激增大量的类，增加系统的复杂性和维护难度。有没有更好的办法来解决呢？