链滴

设计模式之抽象工厂模式

作者：sologxl

原文链接：https://ld246.com/article/1541676052331

来源网站：链滴

示例：

```
//抽象的人类,为所有的人造人提供一个相同的人造人属性
public abstract class Man {
    public abstract void eat();
}

public class AsiaMan extends Man {
    @Override
    public void eat() {
    System.out.println("亚洲人用筷子吃");
    }
}

public  class AmericanSuperMan extends Man  {
    @Override
    public void eat() {
    System.out.println("美国人用刀叉吃");
    }
}
//抽象工厂,为所有的不同的工厂提供一个制作人造人的相同方法
public abstract class Factory {
    public abstract Man product();
}

public class AsiaManFactory extends Factory {
    @Override
    public Man product() {
        Man anAisaMan = new AsiaMan();
        return anAisaMan;
    }
}

public class AmericanFactory extends Factory {
    @Override
    public Man product() {
        AmericanSuperMan anAmericanSuperMan = new AmericanSuperMan();
        return anAmericanSuperMan;
    }
}
public static void main(String[] args) {

    AsiaManFactory asiaManFactory = new AsiaManFactory();
    AmericanFactory americanFactory = new AmericanFactory();

    Man anAmericanSuperMan = americanFactory.product();
    Man anAsia = asiaManFactory.product();

    anAsia.eat();
    anAmericanSuperMan.eat();

}
```

抽象工厂模式即将工厂抽象出来。从而极大提高开发效率。

比如自行车需要两种产品，轮胎和脚踏板。那这个时候就可以创建一个工厂生产轮胎和脚踏板。这个候，这个就是普通工厂模式。

但是当自行车升级后，需要升级第二代轮胎和第二代脚踏板，如果还是创建一个工厂生产第二代轮胎第二代脚踏板，那高层调用者就需要识别哪个工厂是第一代或者第二代。

这个时候抽象工厂就发挥作用了，如果将工厂抽象出来，那高层调用者就只需要知道工厂类就可以了。

抽象工厂模式的优点：

- 封装性。

每个产品的实现类不是高层模块要关心的，它要关心的是什么？

是接口，是抽象，它不关心对象是如何创建出来，这由谁负责呢？工厂类，只要知道工厂类是谁，我能创建出一个需要的对象，省时省力，优秀设计就应该如此。

- 产品族内的约束为非公开状态。

缺点：

抽象工厂模式的最大缺点就是产品族扩展非常困难，为什么这么说呢？我们以通用代码为例，如果要加一个产品C，也就是说产品家族由原来的2个增加到3个，看看我们的程序有多大改动吧！

抽象类AbstractCreator要增加一个方法createProductC()，然后两个实现类都要修改，想想看，这重违反了开闭原则，而且我们一直说明抽象类和接口是一个契约。

改变契约，所有与契约有关系的代码都要修改，那么这段代码叫什么？叫“有毒代码”，——只要与段代码有关系，就可能产生侵害的危险！