正文
接着上一次的工厂方法模式讲。

假设目前你的程序里面有三个对象IphoneX、IphoneXs、IphoneXR的尺寸,那么你使用工厂模式就已经足够了，因为她们属于同一个品类，都属于苹果，如果在添加一个IPhone2019产品，也只需要把IPhone2019加入到你的苹果工厂里面就够了。

但是如果你程序里面还需要知道华为mate10或者小米8的尺寸。这时候你怎么来创建这些对象呢？这时候工厂模式明显已经不适用了，因为工厂模式是对象都实现了同一个接口，这时候就可以使用抽象工厂模式了。

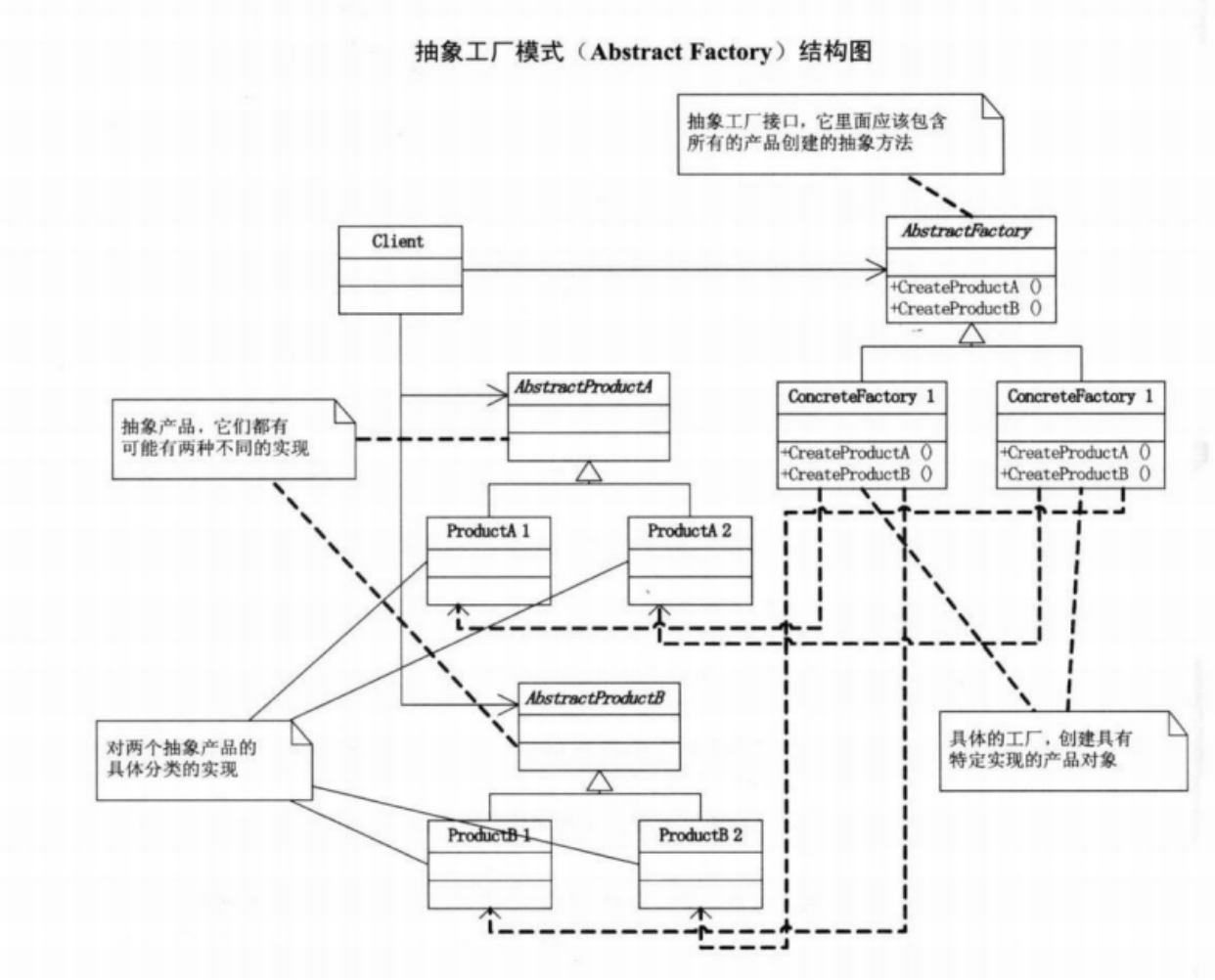
一、什么是抽象工厂模式

为创建一组相关或相互依赖的对象提供一个接口，而且无需指定他们的具体类。

抽象工厂模式是所有形态的工厂模式中最为抽象和最具一般性的一种形态。抽象工厂模式是指当有多个抽象角色时，使用的一种工厂模式。

抽象工厂模式可以向客户端提供一个接口，使客户端在不必指定产品的具体的情况下，创建多个产品族中的产品对象。

根据**里氏替换原则**，任何接受父类型的地方，都应当能够接受子类型。因此，实际上系统所需要的，仅仅是类型与这些抽象产品角色相同的一些实例，而不是这些抽象产品的实例。换言之，也就是这些抽象产品的具体子类的实例。工厂类负责创建抽象产品的具体子类的实例。



二、抽象工厂模式的应用场景

当一个对象都有相同的约束时，可以使用抽象工厂模式。

打个比方说，这个工厂的几个产品都需要经过某些共同的步骤和打上相同的商标，这一组产品可以在一个工厂里面生产，减少很多重复的代码在不同的地方都出现多次。

三、抽象工厂模式和工厂方法模式对比

抽象工厂模式的定义：为创建一组相关或相互依赖的对象提供一个接口，而且无需指定它们的具体类。

工厂方法模式的定义：为某个对象提供一个接口，而且无需指定它们的具体类。

都是子类实现接口的方法，并在子类写具体的代码。

工厂方法模式中也是可以有多个具体工厂，也是可以有多个抽象产品，和多个具体工厂、具体产品类。

区别是在抽象工厂接口类中，能创建几个产品对象。

抽象工厂模式的工厂能创建多个相关的产品对象，而工厂方法模式的工厂只创建一个产品对象。

四、抽象工厂模式的优缺点

优点：

1. 它分离了具体的类
2. 它使得易于交换产品系列
3. 它有利于产品的一致性

缺点：

难以支持新种类的产品。

抽象方法模式的最大缺点就是产品族本身的扩展非常困难。如果在产品族中增加一个新的产品类型，则需要修改多个接口，并影响现已有的工厂类。

上面这句话，有些人不怎么理解，我給大家解释一下，打个比方说，你要在这个工厂创建三个对象，原本只是创建两个对象的，那么你就要在抽象方法中添加一个创建对象的方法，那么所有实现了这个接口的类都是要重新添加这个创建对象的方法，这就是对之前的工厂有影响的原因。

五、抽象工厂模式的实现

首先是抽象产品类

```
// 产品族1
public interface Engine {
    public void produceEngine();
}

// 产品族2
public interface Screen {
    public void produceScreen();
}
```

具体产品类1

```
public class Huawei20Engine implements Engine{
    @Override
    public void produceEngine() {
        System.out.println("生产 Huawei20 引擎");
    }
}
```

```
public class Huawei20Screen implements Screen {
    @Override
    public void produceScreen() {
        System.out.println("生产 Huawei20 屏幕");
    }
}
```

具体产品类2

```
public class IphoneXEngine implements Engine {

    @Override
    public void produceEngine() {
        System.out.println("生产 iphoneX 的引擎");
    }
}


public class IphoneXScreen implements Screen{
    @Override
    public void produceScreen() {
        System.out.println("生产 iphoneX 屏幕");
    }
}
```

抽象工厂类

```
public interface AbstractFactory {
    public Screen createScreen();
    public Engine createEngine();
}
```

具体工厂类1

```
public class Huawei20Factory implements AbstractFactory{
    @Override
    public Engine createEngine() {
        return new Huawei20Engine();
    }

    @Override
    public Screen createScreen() {
        return new Huawei20Screen();
    }
}
```

具体工厂类2

```
public class IphoneXFactory implements AbstractFactory{
    @Override
    public Screen createScreen() {
        return new IphoneXScreen();
    }
```

```
    @Override
    public Engine createEngine() {
        return new IphoneXEngine();
    }
}
```

客户端测试类

```
public class Test {
    public static void main(String[] args) {
        AbstractFactory huawei20Factory = new Huawei20Factory();
        huawei20Factory.createEngine().produceEngine();
        huawei20Factory.createScreen().produceScreen();

        System.out.println();
        AbstractFactory iphoneXFactory = new IphoneXFactory();
        Screen screen = iphoneXFactory.createScreen();
        screen.produceScreen();
        Engine engine = iphoneXFactory.createEngine();
        engine.produceEngine();
    }
}
```

六、总结工厂模式

工厂模式可以分为：简单工厂模式，工厂方法模式，抽象工厂模式。

简单工厂模式就没什么好说的了，无非是所有的东西都写在一个类里面，要什么就调用什么，如果要添加新的方法也是到这类里面添加，代码很多，看起来也是很乱，就像一个大工厂，什么都在里面。扩展性很低。

而工厂方法模式，把说明的理论和生产的东西就分开一点。抽象工厂模式是工厂方法模式的升级。

总结：工厂模式与抽象工厂模式都属于创建型模式，在工厂模式中弥补了简单工厂模式的缺陷（不符合开闭原则），而在抽象工厂模式中弥补了工厂模式的不足（一个工厂只能生产一种产品）。