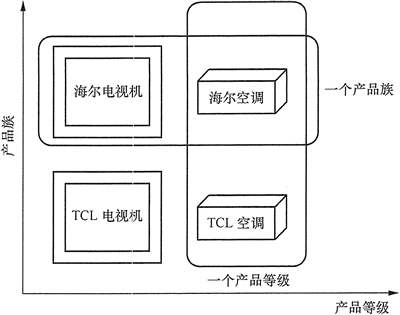
# 抽象工厂模式

## 抽象工厂模式与工厂方法模式

现实生活中存在两种工厂，一种工厂规模较小，只生产一类产品，一种工厂规模较大，不仅限于生产一种产品。这里所说的工厂不限于传统的工厂，这里说的产品也不限于传统的产品。比如说，格力的产品有冰箱、空调、洗衣机……，阿里巴巴的产品有支付宝、淘宝、菜鸟……。

我们称同类产品为同等级产品，一个工厂生产的产品属于同一产品族。

在工厂方法模式中，工厂属于小厂，只生产同等级产品。在抽象工厂模式中，工厂属于大长，生产不同等级产品。



## 定义

抽象工厂模式的定义为：抽象工厂模式是一种，为客户端提供一个创建一组相关或相互依赖对象的接口，且客户端无须指定所要对象的具体类就能得到同族的不同等级的产品的模式结构。

## talk is cheap,show me the code

|  |  |
| --- | --- |
|  |  |



## 特点

抽象工厂模式是工厂方法模式的升级版本，工厂方法模式只生产一个等级的产品，而抽象工厂模式可生产多个等级的产品。

使用抽象工厂模式一般要满足以下条件：

* 系统中有多个产品族，每个具体工厂创建同一族不同等级的产品。
* 系统一次只可能消费其中某一族产品，即同族的产品一起使用。

抽象工厂模式除了具有工厂方法模式的优点外，其他主要优点如下：

* 可以在类的内部对产品族中相关联的多等级产品共同管理，而不必专门引入多个新的类来进行管理。
* 当增加一个新的产品族时不需要修改原代码，满足开闭原则。

其缺点是：当产品族中需要增加一个新的产品时，所有的工厂类都需要进行修改。

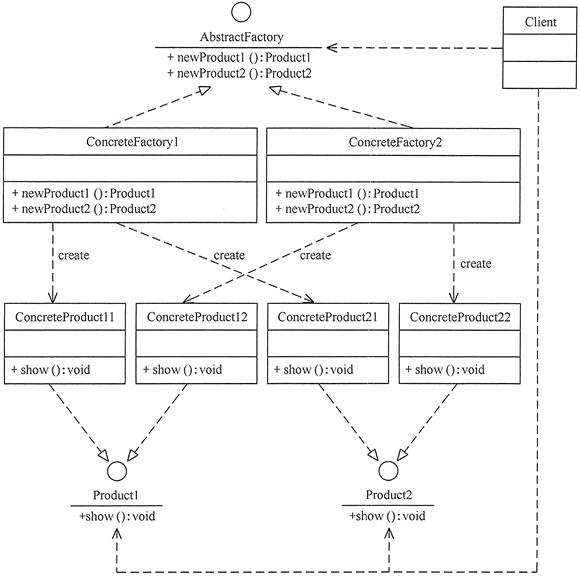
## 结构

抽象工厂模式同工厂方法模式一样，也是由抽象工厂、具体工厂、抽象产品和具体产品等 4 个要素构成，但抽象工厂中方法个数不同，抽象产品的个数也不同。现在我们来分析其基本结构和实现方法。

抽象工厂模式的主要角色如下：

1. 抽象工厂（Abstract Factory）：提供了创建产品的接口，它包含多个创建产品的方法，可以创建不同等级的产品。
2. 具体工厂（Concrete Factory）：主要是实现抽象工厂中的多个抽象方法，完成具体产品的创建。
3. 抽象产品（Product）：定义了产品的规范，描述了产品的主要特性和功能，抽象工厂模式有多个抽象产品。
4. 具体产品（ConcreteProduct）：实现了抽象产品角色所定义的接口，由具体工厂来创建，它 同具体工厂之间是多对一的关系。

抽象工厂模式的结构图如图所示：



## 应用场景

抽象工厂模式最早的应用是用于创建属于不同操作系统的视窗构件。如 java 的 AWT 中的 Button 和 Text 等构件在 Windows 和 UNIX 中的本地实现是不同的。  
  
抽象工厂模式通常适用于以下场景：

1. 当需要创建的对象是一系列相互关联或相互依赖的产品族时，如电器工厂中的电视机、洗衣机、空调等。
2. 系统中有多个产品族，但每次只使用其中的某一族产品。如有人只喜欢穿某一个品牌的衣服和鞋。
3. 系统中提供了产品的类库，且所有产品的接口相同，客户端不依赖产品实例的创建细节和内部结构。

## 拓展

抽象工厂模式的扩展有一定的“开闭原则”倾斜性：

1. 当增加一个新的产品族时只需增加一个新的具体工厂，不需要修改原代码，满足开闭原则。
2. 当产品族中需要增加一个新种类的产品时，则所有的工厂类都需要进行修改，不满足开闭原则。

另一方面，当系统中只存在一个等级结构的产品时，抽象工厂模式将退化到工厂方法模式。