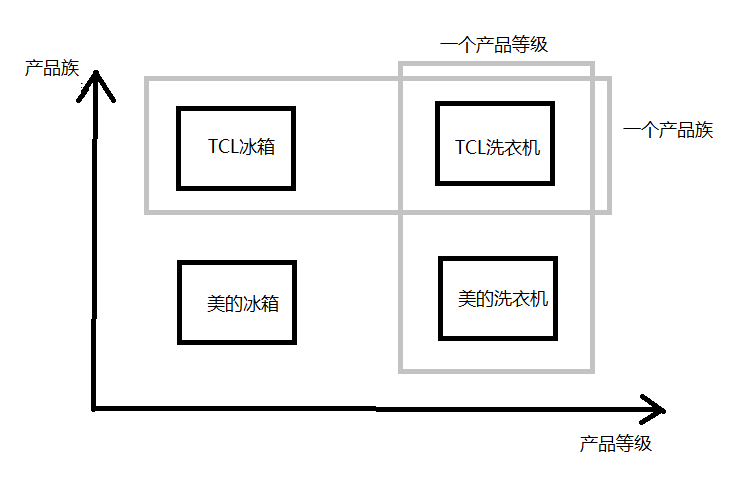
# 定义

为创建一组相关或相互依赖的对象提供一个接口，而且无须指定他们的具体类。

通俗理解就是对一组具有相同主题的工厂进行封装。例如生产一台PC机，如果使用工厂模式的话，就会有主板工厂、显卡工厂、CPU工厂等，而使用抽象工厂的话，只会有一个PC工厂，而PC工厂中又涵盖了主板工厂、显卡工厂、CPU工厂。所以，工厂方法模式针对的是同一类或同等级的产品，而抽象工厂模式针对的是多种类的产品设计。

# 何为产品族



1. 模式特点
2. 系统中有多个产品族，每个具体工厂负责创建同一族但属于不同产品等级（产品种类）的产品。
3. 系统一次只能消费某一族产品，即相同产品族的产品是一起被使用的。当系统需要新增一个产品族时，只需要增加新的工厂类即可，无需修改源代码;
4. 如果需要产品族中增加一个新种类时，则所有的工厂类都需要修改。
5. 抽象工厂模式中的抽象工厂类的职责就是定义每个工厂要实现的功能，即定义多个产品族的产品的创建。同一产品族下有多个产品时，对应的抽象工厂就会有多个抽象方法用来提供创建这些产品的接口。
6. **组成角色**

. 抽象工厂

提供了创建产品的接口，包含多个创建产品的方法，即包含多个类似的new Product()的方法;

. 具体工厂

实现抽象工厂定义的接口，完成某个具体产品的创建；

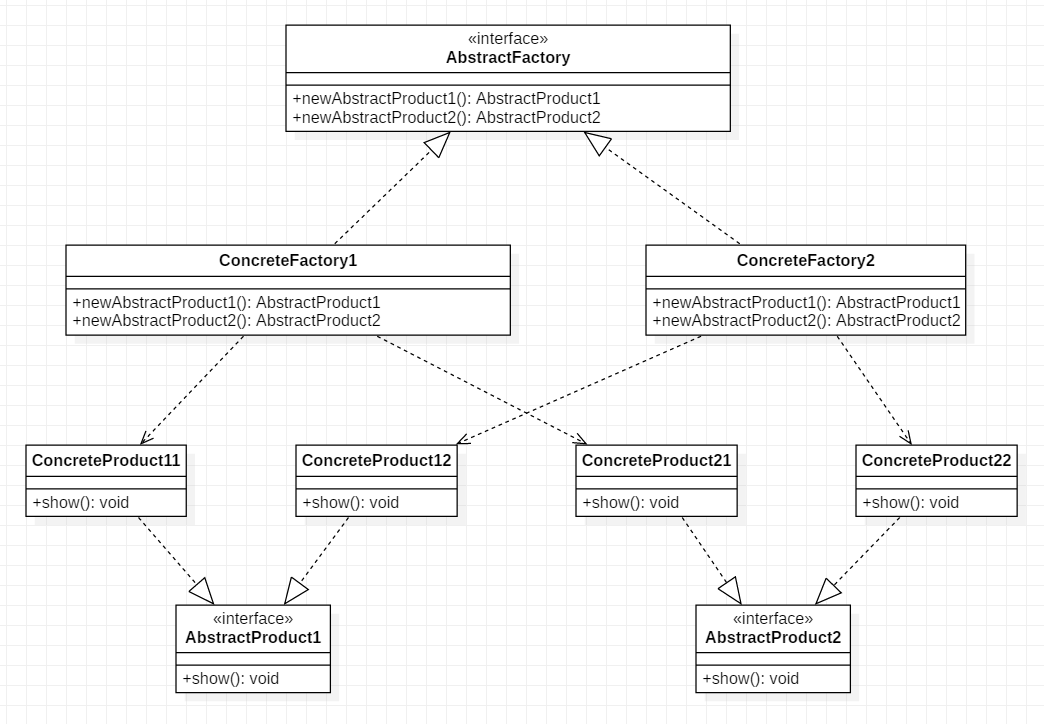
. 抽象产品

抽象产品的定义，一般有多少抽象产品，抽象工厂中就包含多少个创建产品的方法；

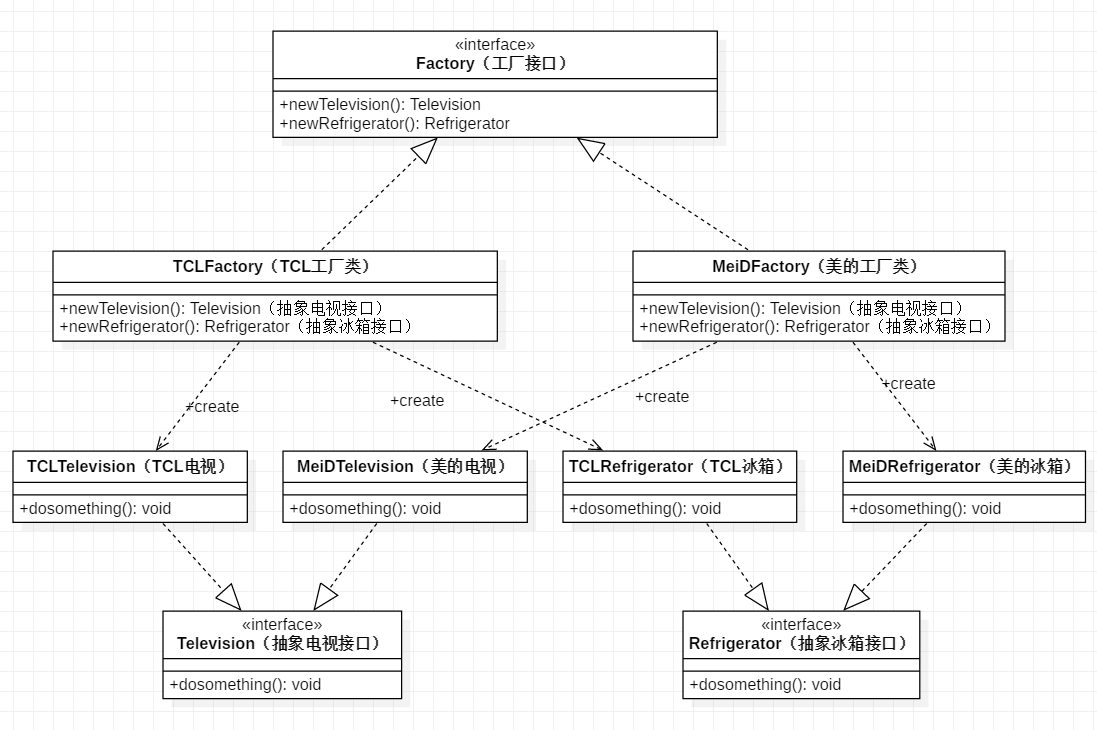
. 具体产品

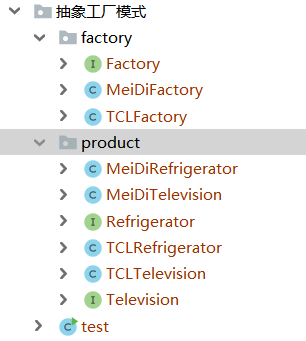
抽象产品的实现类。

实现抽象产品所定义的接口，和具体工厂是一对多的关系

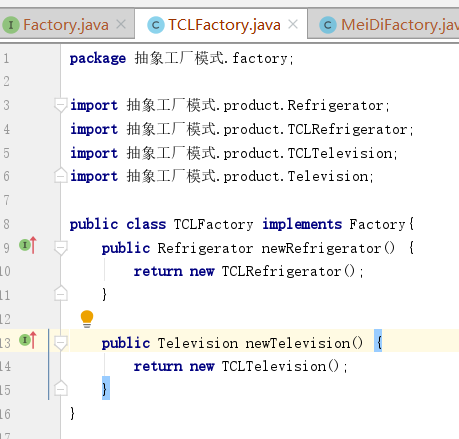


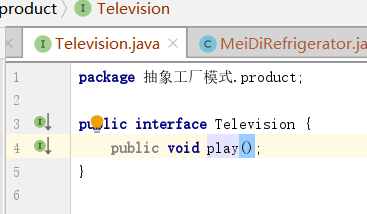
1. **使用实例**



****



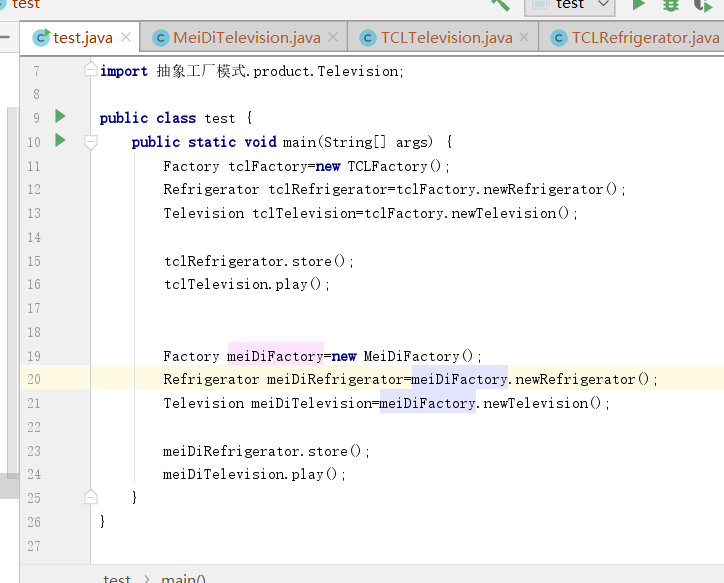


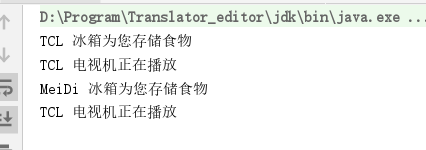




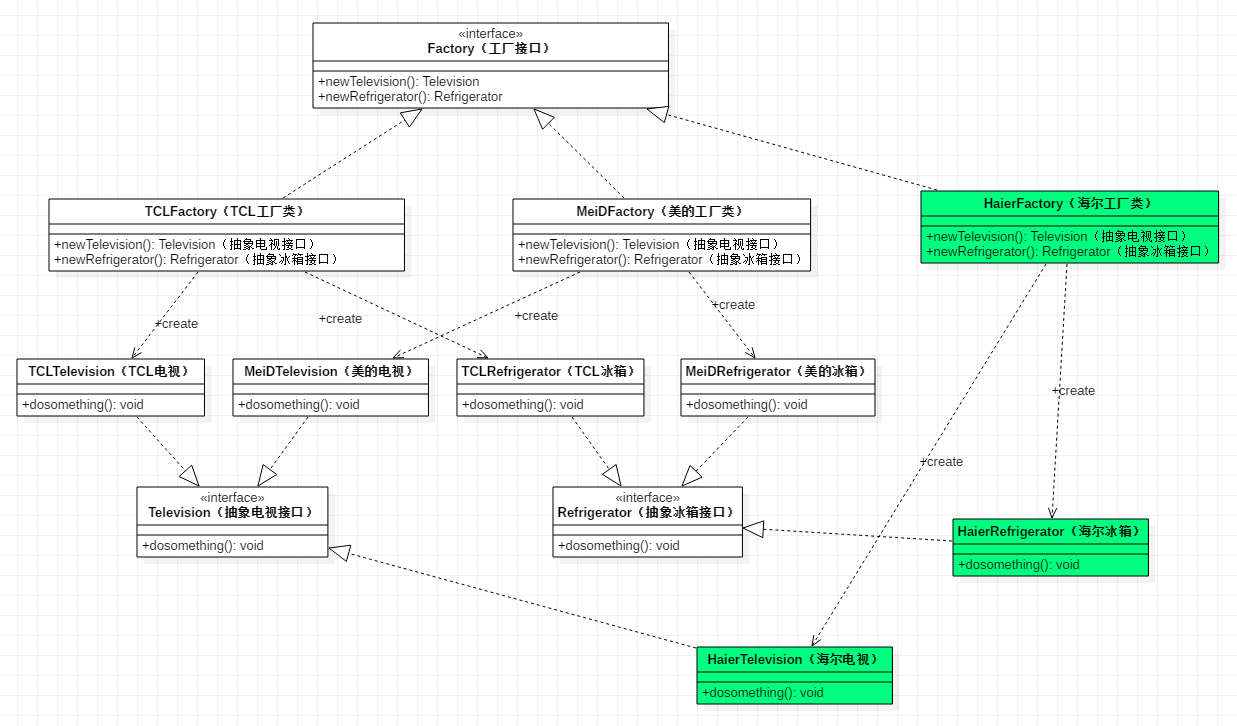




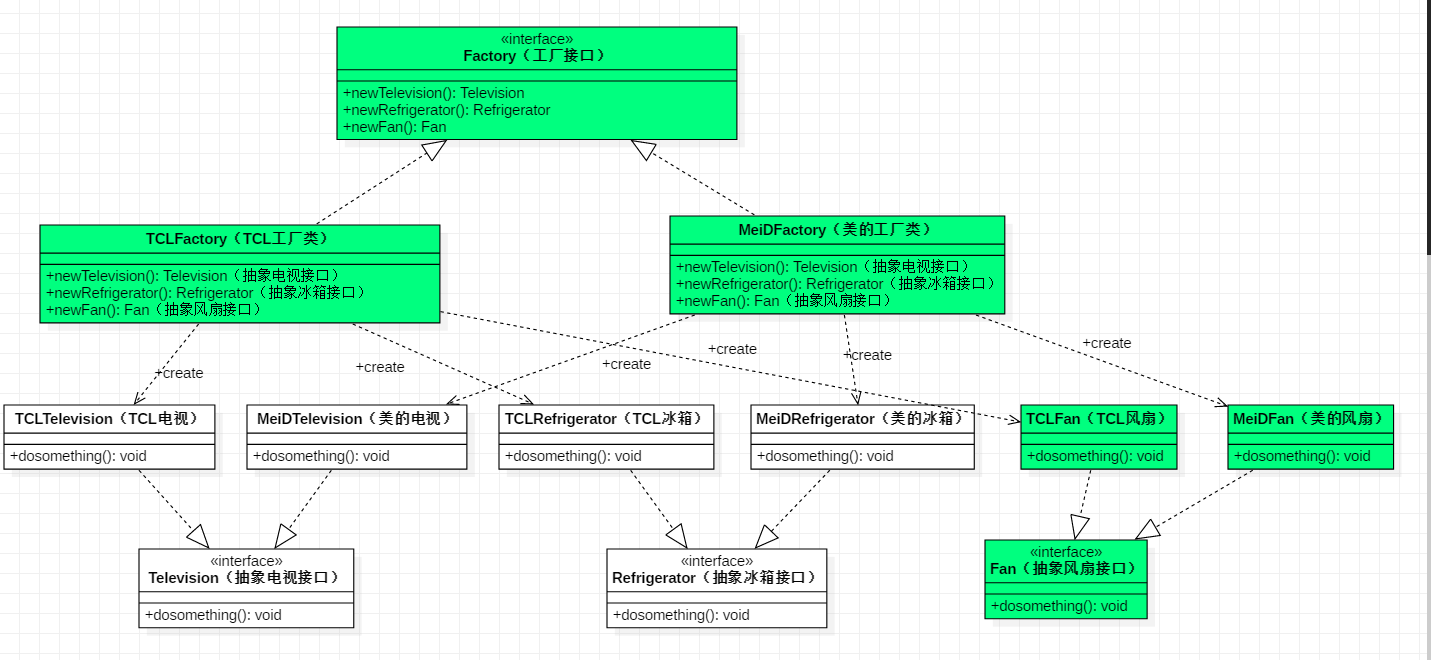




1. **括展：**
   1. **新增品牌厂**



* 1. **新增产品种类**



由原图可知，当需要增加一个产品类型时，所有的工厂类都需要进行修改，不满足开闭原则。所以这是抽象工厂模式的一个弊端。