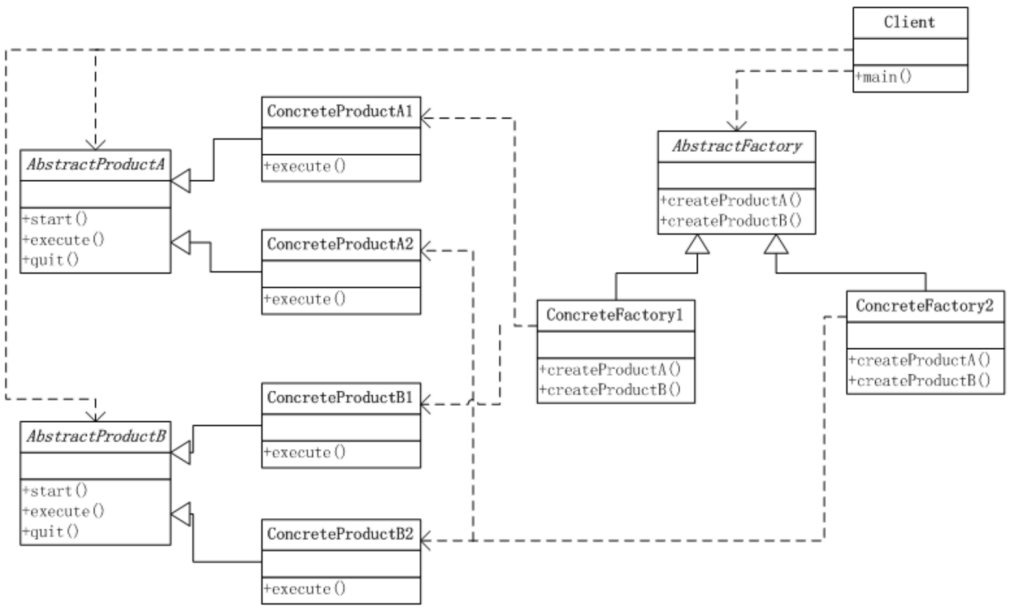
Abstract Factory 抽象工厂

动机（Motivation）:在软件系统中，经常面临着“一系列相互依赖的对象”的创建工作；同时，由于需求的变化，往往存在更多系列对象的创建工作。如何应对这种变化？如何绕过常规的对象创建方法(new)，提供一种“封装机制”来避免客户程序和这种“多系列具体对象创建工作”的紧耦合?

定义:提供一个接口，让该接口负责创建一系列“相关或者相互依赖的对象”，无需指定它们具体的类。——《设计模式》GoF

- 客户`Client`
- 抽象工厂接口`AbstractFactory`
- 抽象工厂的实现类`ConcreteFactory`
- 抽象产品接口`AbstractProduct`
- 产品实现类`ConcreteProduct`



要点总结: 如果没有应对“多系列对象构建”的需求变化，则没有必要使用Abstract Factory模式，这时候使用简单的工厂完全

可以。“系列对象”指的是在某一特定系列下的对象之间有相互依赖、或作用的关系。不同系列的对象之间不能相互依赖。Abstract Factory模式主要在于应对“新系列”的需求变动。其缺点在于难以应对“新对象”的需求变动

意图：提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。

主要解决：主要解决接口选择的问题。

何时使用：系统的产品有多于一个的产品族，而系统只消费其中某一族的产品。

如何解决：在一个产品族里面，定义多个产品。

关键代码：在一个工厂里聚合多个同类产品。

优点：当一个产品族中的多个对象被设计成一起工作时，它能保证客户端始终只使用同一个产品族中的对象。

缺点：产品族扩展非常困难，要增加一个系列的某一产品，既要在抽象的 Creator 里加代码，又要在具体的里面加代码。

使用场景： 1、QQ 换皮肤，一整套一起换。 2、生成不同操作系统的程序。

注意事项：产品族难扩展，产品等级易扩展。