2、抽象工厂模式

工厂方法模式和抽象工厂模式不好分清楚，他们的区别如下：

工厂方法模式：

一个抽象产品类，可以派生出多个具体产品类。

一个抽象工厂类，可以派生出多个具体工厂类。

每个具体工厂类只能创建一个具体产品类的实例。

抽象工厂模式：

多个抽象产品类，每个抽象产品类可以派生出多个具体产品类。

一个抽象工厂类，可以派生出多个具体工厂类。

每个具体工厂类可以创建多个具体产品类的实例，也就是创建的是一个产品线下的多个产品。

区别：

工厂方法模式只有一个抽象产品类，而抽象工厂模式有多个。

工厂方法模式的具体工厂类只能创建一个具体产品类的实例，而抽象工厂模式可以创建多个。

工厂方法创建 "一种" 产品，他的着重点在于"怎么创建"，也就是说如果你开发，你的大量代码很可能围绕着这种产品的构造，初始化这些细节上面。也因为如此，类似的产品之间有很多可以复用的特征，所以会和模版方法相随。 抽象工厂需要创建一些列产品，着重点在于"创建哪些"产品上，也就是说，如果你开发，你的主要任务是划分不同差异的产品线，并且尽量保持每条产品线接口一致，从而可以从同一个抽象工厂继承。

对于java来说，你能见到的大部分抽象工厂模式都是这样的：

---它的里面是一堆工厂方法，每个工厂方法返回某种类型的对象。

比如说工厂可以生产鼠标和键盘。那么抽象工厂的实现类（它的某个具体子类）的对象都可以生产鼠标和键盘，但可能工厂A生产的是罗技的键盘和鼠标，工厂B是微软的。

这样A和B就是工厂，对应于抽象工厂；

每个工厂生产的鼠标和键盘就是产品，对应于工厂方法；

用了工厂方法模式，你替换生成键盘的工厂方法，就可以把键盘从罗技换到微软。但是用了抽象工厂模式，你只要换家工厂，就可以同时替换鼠标和键盘一套。如果你要的产品有几十个，当然用抽象工厂模式一次替换全部最方便（这个工厂会替你用相应的工厂方法）

所以说抽象工厂就像工厂，而工厂方法则像是工厂的一种产品生产线