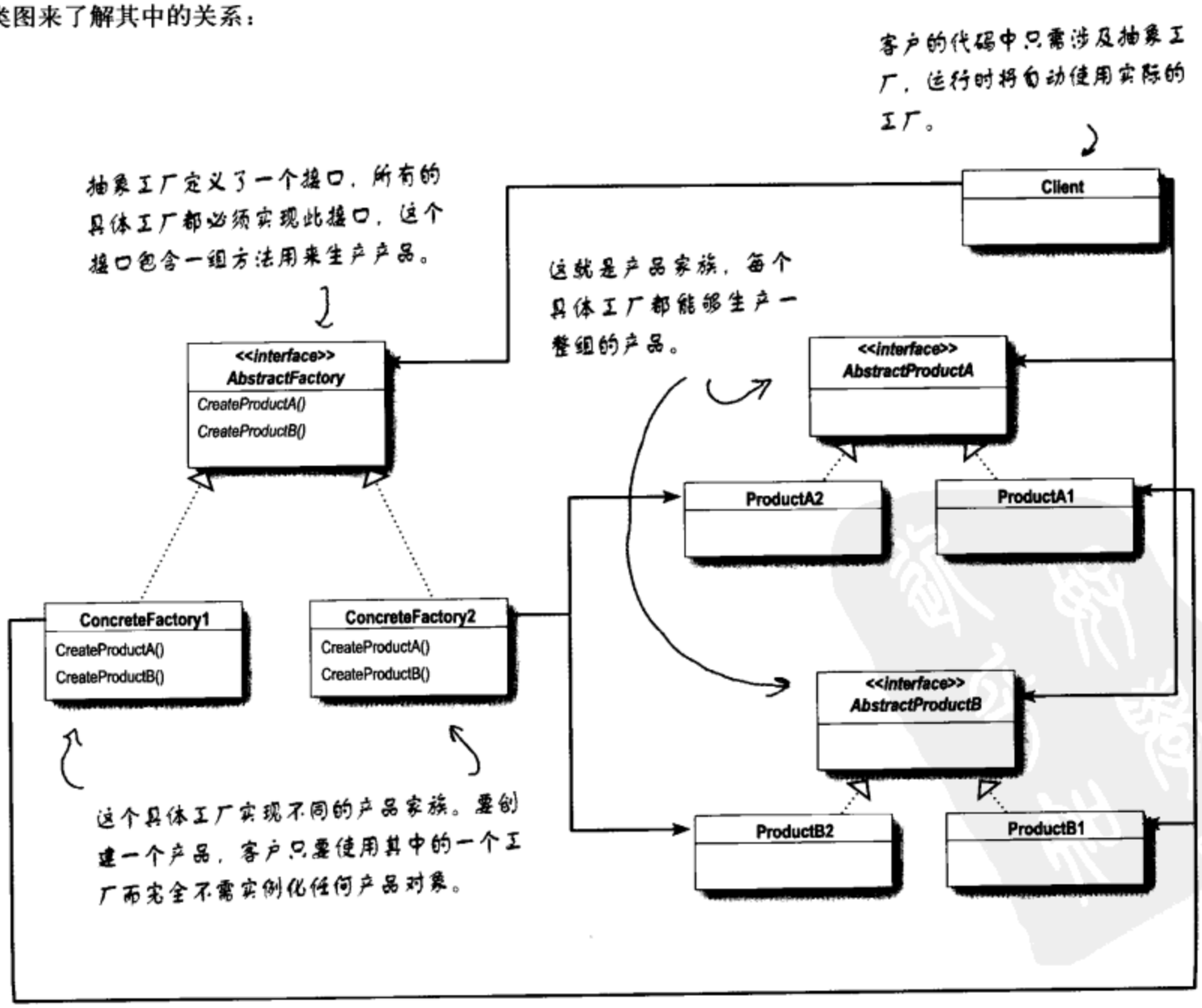
定义抽象工厂模式

我们又在模式家族中新增了另一个工厂模式，这个模式可以创建产品的家族。看看这个模式的正式定义：

> **抽象工厂模式**提供一个接口，用于创建相关或依赖对象的家族，而不需要明确指定具体类。

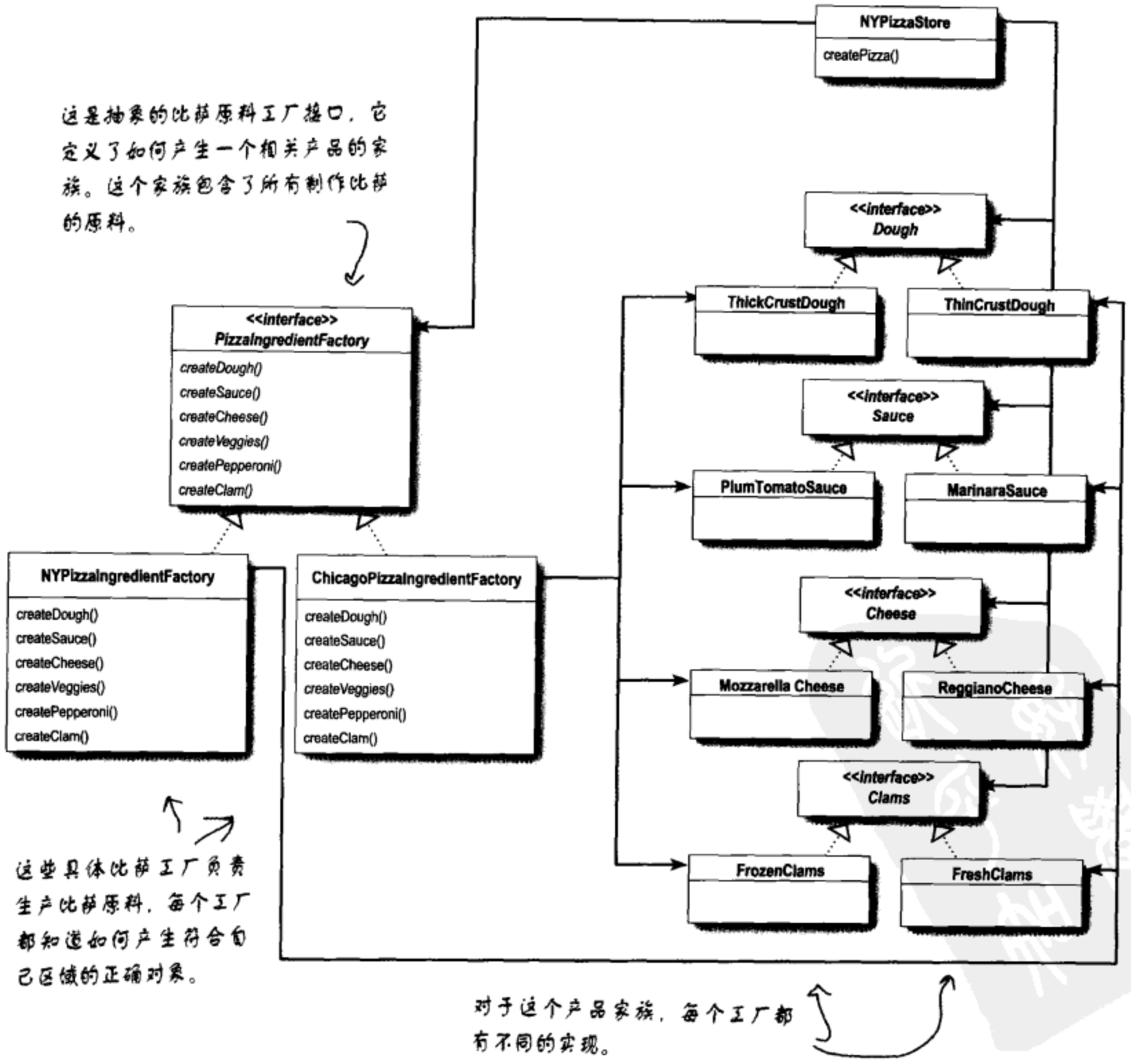
抽象工厂允许客户使用抽象的接口来创建一组相关的产品，而不需要知道（或关心）实际产出的具体产品是什么。这样一来，客户就从具体的产品中被解耦。让我们看看类图来了解其中的关系：



这是一张相当复杂的类图；让我们从PizzaStore的观点来看一看它：

比萨店的两个具体实例（NYPizzaStore、ChicagoPizzaStore）是抽象工厂的客户。

这是抽象的比萨原料工厂接口，它定义了如何产生一个相关产品的家族。这个家族包含了所有制作比萨的原料。



这些具体比萨工厂负责生产比萨原料，每个工厂都知道如何产生符合自己区域的正确对象。

对于这个产品家族，每个工厂都有不同的实现。

我注意到了，抽象工厂的每个方法实际上看起来都像是工厂方法（例如：createDough()、createSource()等）。每个方法都被声明成抽象，而子类的方法覆盖这些方法来创建某些对象。这不正是工厂方法吗？

工厂方法是不是潜伏在抽象工厂里面？

你的观察力很敏锐！是的，抽象工厂的方法经常以工厂方法的方式实现，这很有道理，对吧？抽象工厂的任务是定义一个负责创建一组产品的接口。这个接口内的每个方法都负责创建一个具体产品，同时我们利用实现抽象工厂的子类来提供这些具体的做法。所以，在抽象工厂中利用工厂方法实现生产方法是相当自然的做法。

模式告白

本周访问：
工厂方法和抽象工厂

HeadFirst：哇！今天很难得，同时请到了两种模式。这可是头一回啊！

工厂方法：呵！我其实不希望人们把我和抽象工厂混为一谈。虽然我们都是工厂模式，但并不表示我们就应该被合在一起访问。

HeadFirst：别生气，我们之所以想要同时采访你们就是为了帮读者搞清楚你们之间谁是谁。你们的确有相似的地方，听说人们常常会把你们搞混了。

抽象工厂：这是真的，有些时候我被错认为是工厂方法。嘿！工厂方法，我知道你也有相同的困扰。我们两个在把应用程序从特定实现中解耦方面真的都很有一套，只是做法不同而已。我能够理解为什么人们总是把我们搞混。

工厂方法：哎呀！这还是让我很不爽。毕竟，我使用的是类而你使用的是对象，根本就不是一回事啊。

HeadFirst：工厂方法，能请你多做一些解释吗？

工厂方法：当然。抽象工厂与我都是负责创建对象，这是我们的工作。但是我用的方法是继承……

抽象工厂：……而我是通过对象的组合。

工厂方法：对！所以这意味着，利用工厂方法创建对象，需要扩展一个类，并覆盖它的工厂方法。

HeadFirst：那这个工厂方法是做什么的呢？

工厂方法：当然是用来创建对象的了。其实整个工厂方法模式，只不过就是通过子类来创建对象。用这种做法，客户只需要知道他们所使用的抽象类型就可以了，而由子类来负责决定具体类型。所以，换句话说，我只负责将客户从具体类型中解耦。

抽象工厂：这一点我也做得到，只是我的做法不同。

HeadFirst：抽象工厂，请继续…… 你刚刚说了一些关于对象组合的事？

抽象工厂：我提供一个用来创建一个产品家族的抽象类型，这个类型的子类定义了产品被产生的方法。要想使用这个工厂，必须先实例化它，然后将它传入一些针对抽象类型所写的代码中。所以，和工厂方法一样，我可以把客户从所使用的实际具体产品中解耦。

HeadFirst：噢！我了解了，所以你的另一个优点是可以把一群相关的产品集合起来。

抽象工厂：对。

HeadFirst：万一需要扩展这组相关产品（比方说新增一个产品），又该怎么办呢？难道这不需要改变接口吗？

抽象工厂：那倒是真的，如果加入新产品就必须改变接口，我知道大家不喜欢这么做……

工厂方法：＜窃笑＞

抽象工厂：我说，工厂方法，你偷笑什么？

工厂方法：拜托，那可是很严重的！改变接口就意味着必须深入改变每个子类的接口！听起来可是很繁重的工作呀。

抽象工厂：是的，但是我需要一个大的接口，因为我可是被用来创建整个产品家族的。你只不过是创建一个产品，所以你根本不需要一个大的接口，你只需要一个方法就可以了。

HeadFirst：抽象工厂，我听说你经常使用工厂方法来实现你的具体工厂。

抽象工厂：是的，我承认这一点，我的具体工厂经常实现工厂方法来创建他们的产品。不过对我来说，这些具体工厂纯粹只是用来创建产品罢了……

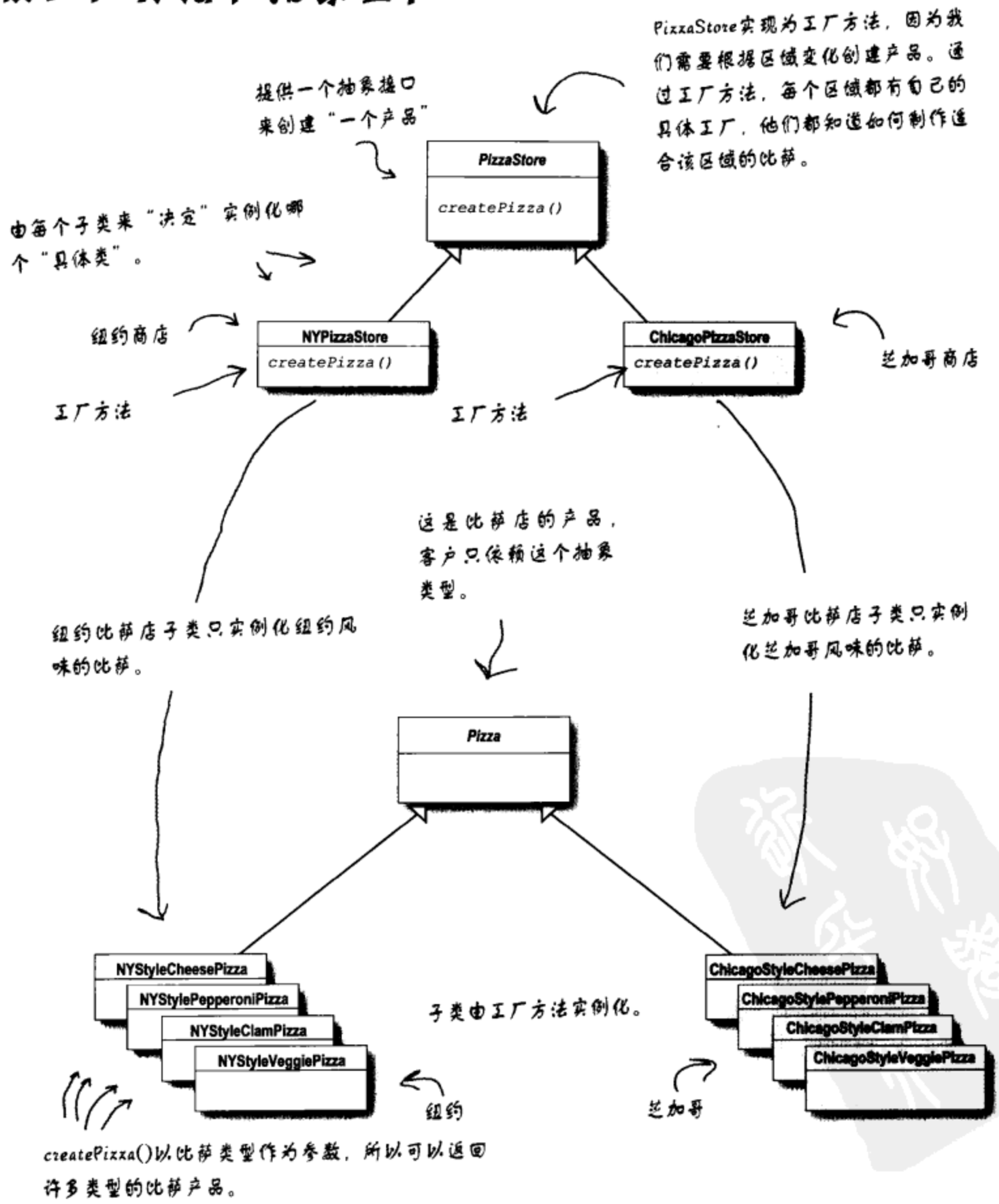
工厂方法：……而对我来说，抽象创建者（creator）中所实现的代码通常会用到子类所创建的具体类型。

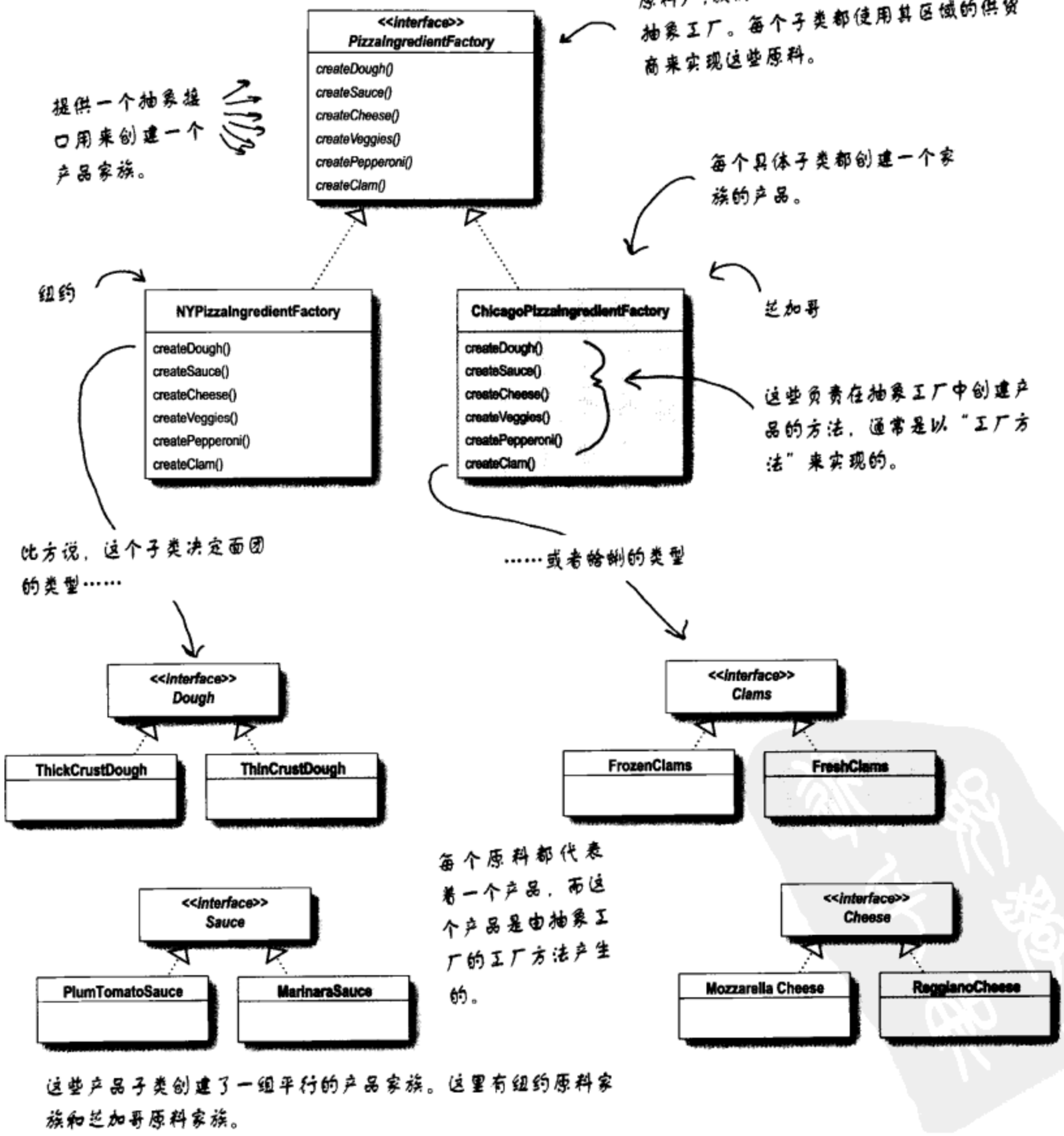
HeadFirst：听起来你们都有自己的一套。我相信人们喜欢有选择的余地，毕竟，工厂这么有用，大家希望在各种不同的情况下都可使用工厂。你们俩都能将对象的创建封装起来，使应用程序解耦，并降低其对特定实现的依赖。真的是很棒。不管是使用工厂方法还是抽象工厂，都可以给人们带来好处。节目结束前，请两位各说几句话吧。

抽象工厂：谢谢。我是抽象工厂，当你需要创建产品家族和想让制造的相关产品集合起来时，你可以使用我。

工厂方法：而我是工厂方法，我可以把你的客户代码从需要实例化的具体类中解耦。或者如果你目前还不知道将来需要实例化哪些具体类时，也可以用我。我的使用方式很简单，只要把我继承成子类，并实现我的工厂方法就可以了。

比较工厂方法和抽象工厂

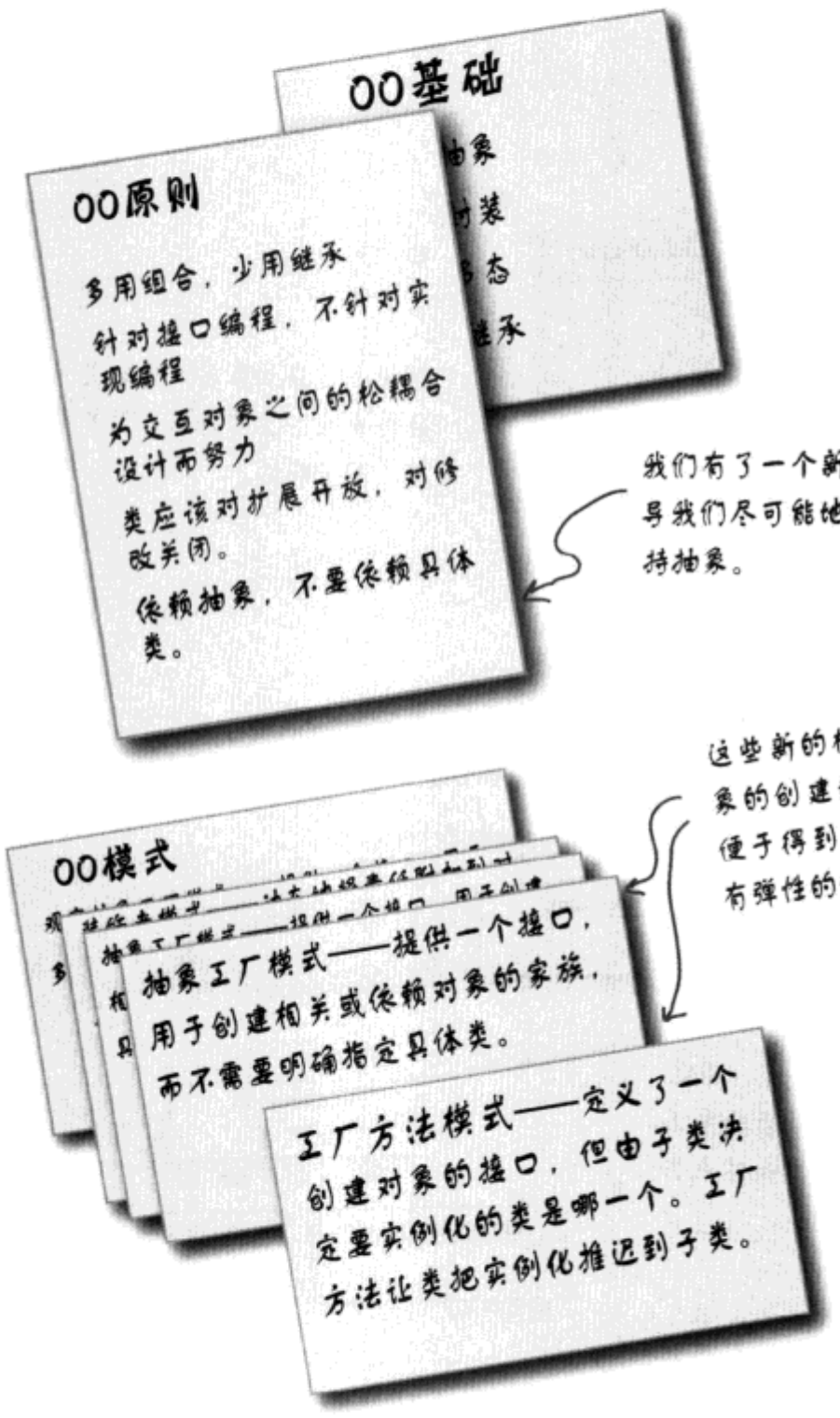




这些产品子类创建了一组平行的产品家族。这里有纽约原料家族和芝加哥原料家族。

设计箱内的工具

在本章，我们多加了两个工具到你的工具箱中：工厂方法和抽象工厂。这两种模式都是将对象创建的过程封装起来，以便将代码从具体类解耦。



要点

- 所有的工厂都是用来封装对象的创建
- 简单工厂，虽然不是真正的设计模式，但仍不失为一个简单的方法，可以将客户程序从具体类解耦。
- 工厂方法使用继承：把对象的创建委托给子类，子类实现工厂方法来创建对象。
- 抽象工厂使用对象组合：对象的创建被实现在工厂接口所暴露出来的方法中。
- 所有工厂模式都通过减少应用程序和具体类之间的依赖促进松耦合。
- 工厂方法允许类将实例化延迟到子类进行。
- 抽象工厂创建相关的对象家族，而不需要依赖它们的具体类。
- 依赖倒置原则，指导我们避免依赖具体类型，而要尽量依赖抽象。
- 工厂是很有威力的技巧，帮助我们针对抽象编程，而不要针对具体类编程。