【将关键零件组装成产品】

Abstract的意思是“抽象的”，Factory的意思是“工厂”。将它们组合起来我们就可以知道Abstract Factory表示“抽象工厂”的意思。

通常，我们不会将“抽象的”这个词与“工厂”这个词联系到一起。所谓工厂，是将零件组装成产品的地方，这是一项具体的工作。那么“抽象工厂”到底是什么意思呢？

我们大可不必对这个词表示吃惊。因为在Abstract Factory模式中，不仅有“抽象工厂”，还有

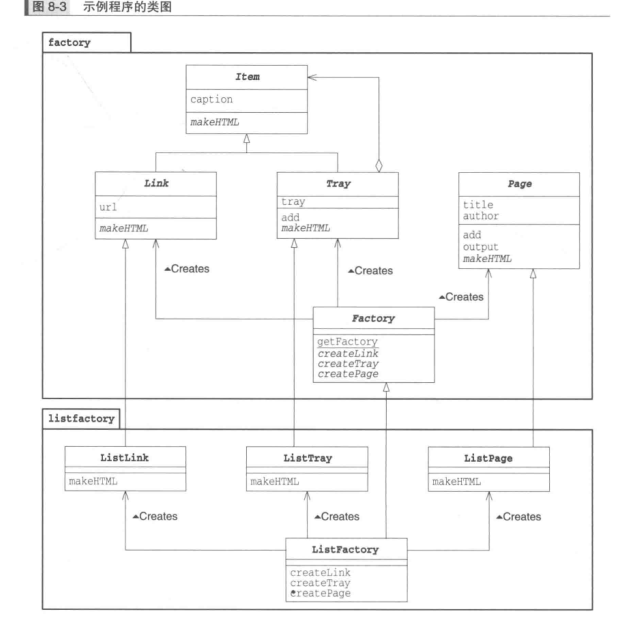
“抽象零件”和“抽象产品”。抽象工厂的工作是将“抽象零件”组装为“抽象产品”。

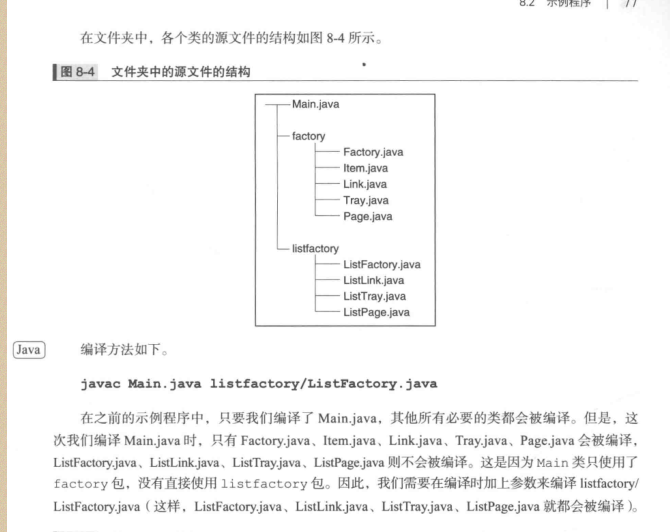
读到这里，大家可能会想“哎呀哎呀，你到底想说什么啊？”那么请大家先回忆一下面向对象编程中的“抽象”这个词的具体含义。它指的是“不考虑具体怎样实现，而是仅关注接口（API）”的状态。例如，抽象方法（Abstract Method）并不定义方法的具体实现，而是仅仅只确定了方法的名字和签名（参数的类型和个数）。

关于“忘记方法的具体实现（假装忘记），使用抽象方法进行编程”的设计思想，我们在Template Method模式（第3章）和Builder模式（第7章）中已经稍微提及了一些。

在Abstract Factory模式中将会出现抽象工厂，它会将抽象零件组装为抽象产品。也就是说，我们并不关心零件的具体实现，而是只关心接口（APl）。我们仅使用该接口（API）将零件组装成为产品。

在Tempate Method模式和Builder模式中，子类这一层负责方法的具体实现。在Abstract Factory模式中也是一样的。在子类这一层中有具体的工厂，它负责将具体的零件组装成为具体的产品。



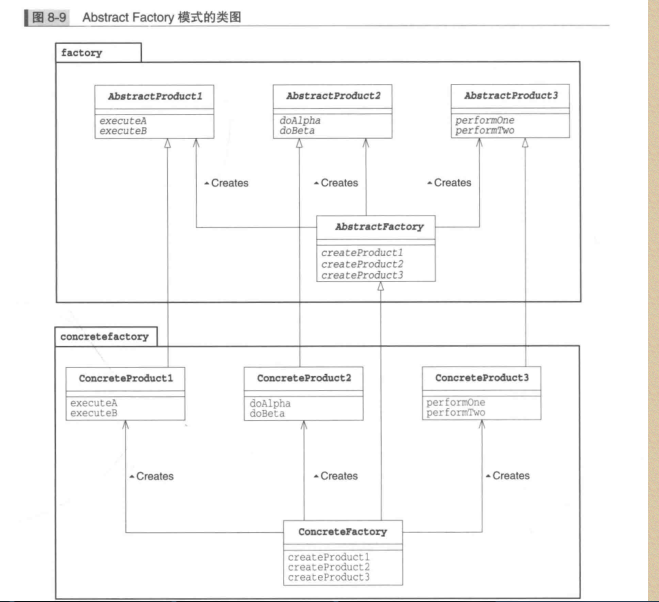


|8.4Abstract Factory模式中的登场角色在Abstract Factory模式中有以下登场角色。

◆AbstractProduct（抽象产品）AbstractProduct角色负责定义AbstractFactory角色所生成的抽象零件和产品的接口（API）。在示例程序中，由Link类、Tray类和Page类扮演此角色。

◆AbstractFactory（抽象工厂）AbstractFactory角色负责定义用于生成抽象产品的接口（API）。在示例程序中，由Factory类扮演此角色。

·Client（委托者）Client 角色仅会调用AbstractFactory角色和AbstractProduct角色的接口（API）来进行工作，对于具体的零件、产品和工厂一无所知。在示例程序中，由Main类扮演此角色。图8-9省略了Client这一角色。



◆ConcreteProduct（具体产品）ConcreteProduct角色负责实现AbstractProduct角色的接口（API）。在示例程序中，由以下包中的以下类扮演此角色。

·1istfactory包：ListLink类、ListTray类和ListPage类

·tablefactory包：TableLink类、TableTray类和TablePage类

◆ConcreteFactory（具体工厂）ConcreteFactory角色负责实现AbstractFactory角色的接口（API）。在示例程序中，由以下包中的以下类扮演此角色。

·1istfactory包：Listfactory类，

·tablefactory包：Tablefactory类

|8.5拓展思路的要点

易于增加具体的工厂

在Abstract Factory模式中增加具体的工厂是非常容易的。这里说的“容易”指的是需要编写哪些类和需要实现哪些方法都非常清楚。

假设现在我们要在示例程序中增加新的具体工厂，那么需要做的就是编写Factory、Link、Tray、Page这4个类的子类，并实现它们定义的抽象方法。也就是说将factory包中的抽象部分全部具体化即可。

这样一来，无论要增加多少个具体工厂（或是要修改具体工厂的Bug），都无需修改抽象工厂和Main部分。难以增加新的零件

请试想一下要在Abstract Factory模式中增加新的零件时应当如何做。例如，我们要在factory包中增加一个表示图像的Picture零件。这时，我们必须要对所有的具体工厂进行相应的修改才行。例如，在1istfactory包中，我们必须要做以下修改。

·在ListFactory中加入createPicture方法

·新增ListPicture类

已经编写完成的具体工厂越多，修改的工作量就会越大。

|8.6相关的设计模式

◆Builder模式（第7章）Abstract Factory模式通过调用抽象产品的接口（AP1）来组装抽象产品，生成具有复杂结构的实例。

Builder模式则是分阶段地制作复杂实例。·Factory Method模式（第4章）有时Abstract Factory模式中零件和产品的生成会使用到Factory Method模式。

◆Composite模式（第11章）有时Abstract Factory模式在制作产品时会使用Composite模式。

·Singleton模式（第5章）有时Abstract Factory模式中的具体工厂会使用Singleton模式。

|8.7延伸阅读：各种生成实例的方法的介绍

在Java中可以使用下面这些方法生成实例。

◆new一般我们使用Java关键字new生成实例。

可以像下面这样生成Something类的实例并将其保存在obj变量中。

Something obj=new Something（）；这时，类名（此处的Something）会出现在代码中”。

◆clone我们也可以使用在Prototype模式（第6章）中学习过的c1one方法，根据现有的实例复制出一个新的实例。

我们可以像下面这样根据自身来复制出新的实例（不过不会调用构造函数）。

class Something{

public Something createclone（）（

Something obj=null；

try{

obj=（Something）clone（）；

catch（CloneNotSupportedException e）{

e.printstackTrace（）；

}

return obj；

}

}

newlnstance使用本章中学习过的java.1ang.Class类的newInstance方法可以通过Class类的实例生成出Class类所表示的类”的实例（会调用无参构造函数）。

在本章的示例程序中，我们已经展示过如何使用newInstance了。下面我们再看一个例子。

假设我们现在已经有了something类的实例someobj，通过下面的表达式可以生成另外一个Something类的实例。

someobj.getClasa（）.newInstance（）

实际上，调用newInstance 方法可能会导致抛出InstantiationException异常或是I1legalAccessException异常，因此需要将其置于try..catch 语句块中或是用throws关键字指定调用newInstance方法的方法可能会抛出的异常。

①即形成强耦合关系。——译者注

②即Something类。——译者注