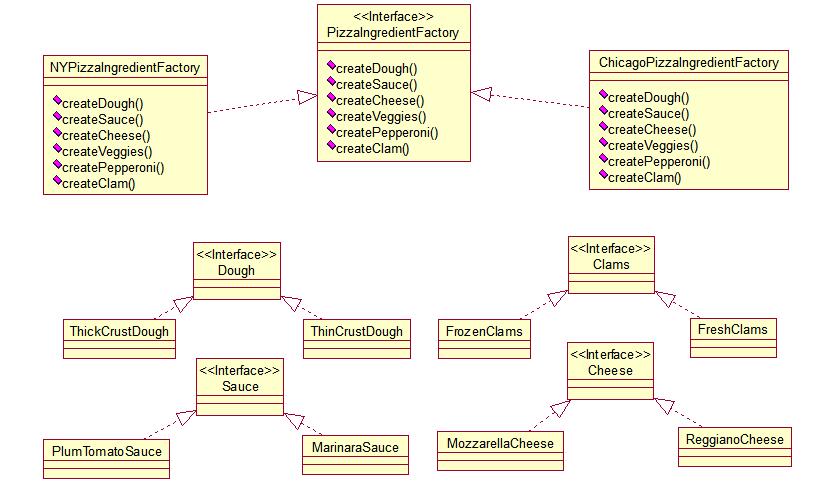
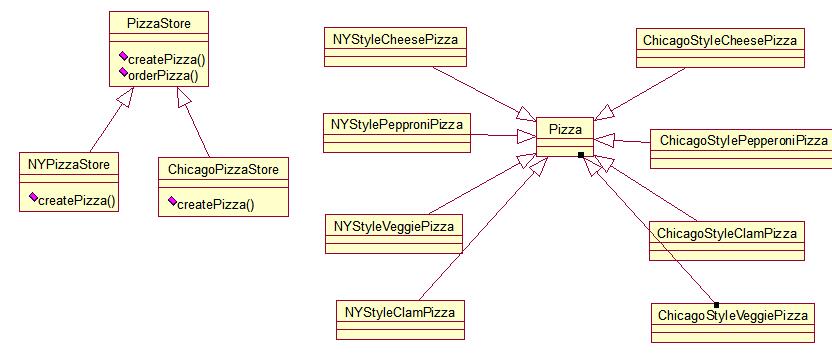
**抽象工厂模式：**提供一个接口，用于创建相关或依赖对象的家族，而不需要明确指定具体类。

**也许应该换一种方式，或许应该先把类图给大家呈现出来。感觉前面的模式讲解因为代码过多，估计没有多少人能够坚持看下去。**

**不过在给出大家类图之前还是建议将之前的pizza店中的工厂模式看一下，因为这个类图还是基于上面的例子。看着英文的名字大致也可以明白类图。**



　　在这里同样给出工厂方法的类图，大家通过比较来看一下两个模式之间的不同点：



　　在这里就不在给那些长且无用的代码了 （给了也没人看，~~~~(>\_<)~~~~）

　　那么我们现在做一个简单的总结吧。毕竟我们已经在设计模式这条路上走了一段距离:

**OO基础：抽象、封装、继承、多态**

**OO原则：**

**1.封装变化；**

**2.多用组合，少用继承；**

**3.针对接口编程，不针对实现编程；**

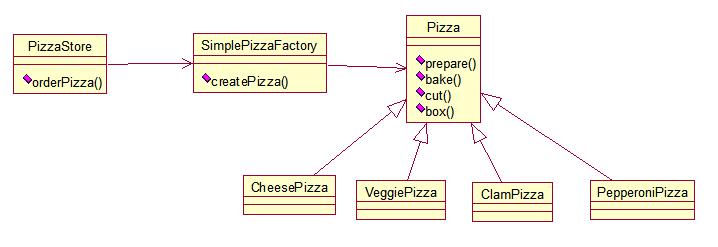
**4.为交互对象之间的松耦合设计而努力；**

**5.类应该对扩展开放，对修改封闭；**

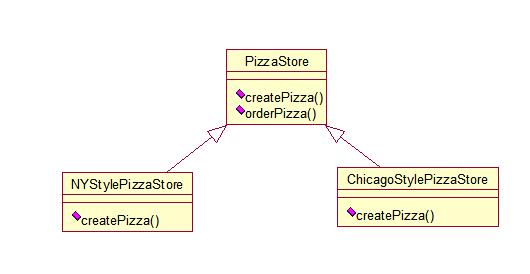
**6.依赖抽象，不要依赖具体类**

**OO模式：**

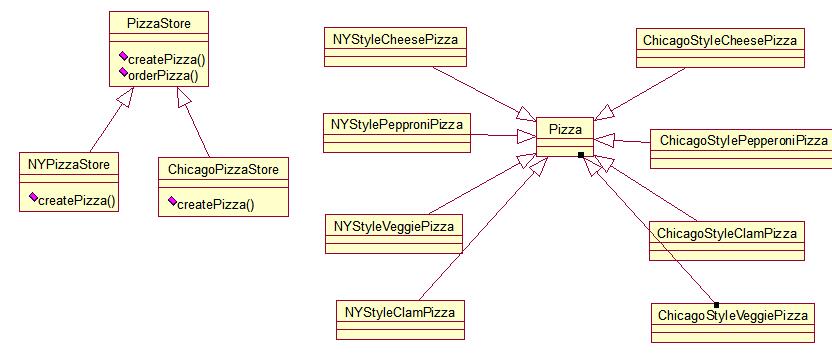
**1.策略模式 定义了一系列的算法，并将每一个算法封装起来，而且使它们还可以相互替换。策略模式让算法独立于使用它的客户而独立变化。**

****

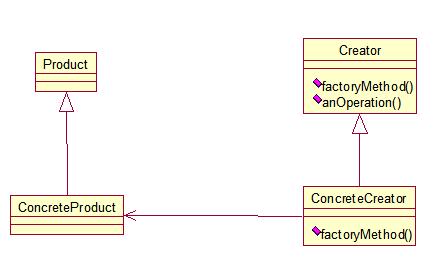
**2.观察者模式 在对象之间定义一对多的依赖，这样一来，当一个对象改变状态，依赖它的对象都会收到通知，并自动更新。**

****

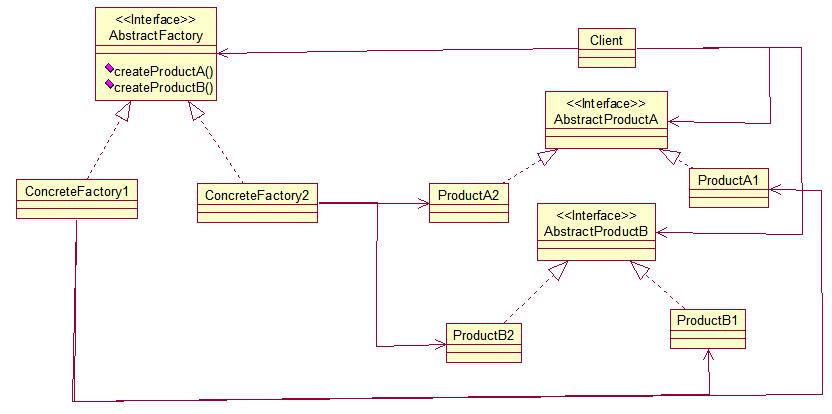
**3.装饰者模式 动态的将责任附加到对象上。想要扩展功能，装饰者提供有别于继承的另一种选择。**

****

**4.工厂方法模式 定义了一个创建对象的接口，但由子类决定要实例化的类是哪一个。工厂方法让类把实例化推迟到子类。**

****

**5.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**

****