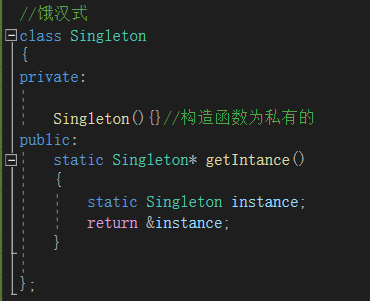
# 单例模式

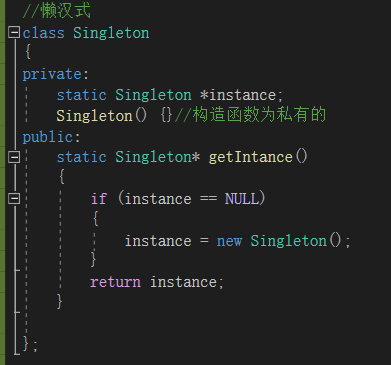
单例模式主要解决一个全局使用的类频繁的创建和销毁的问题。单例模式下可以确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例。单例模式有三个要素：一是某个类只能有一个实例；二是它必须自行创建这个实例；三是它必须自行向整个系统提供这个实例。

单例模式的实现有两种：

第一种是饿汉式：



第二种是懒汉式：

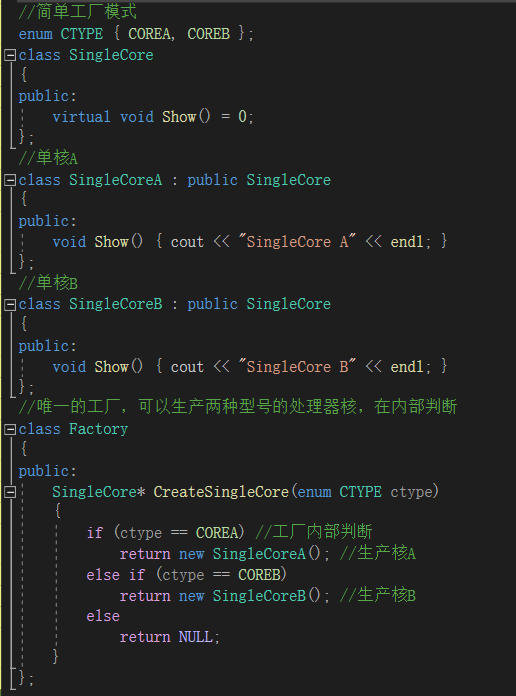


# 工厂模式

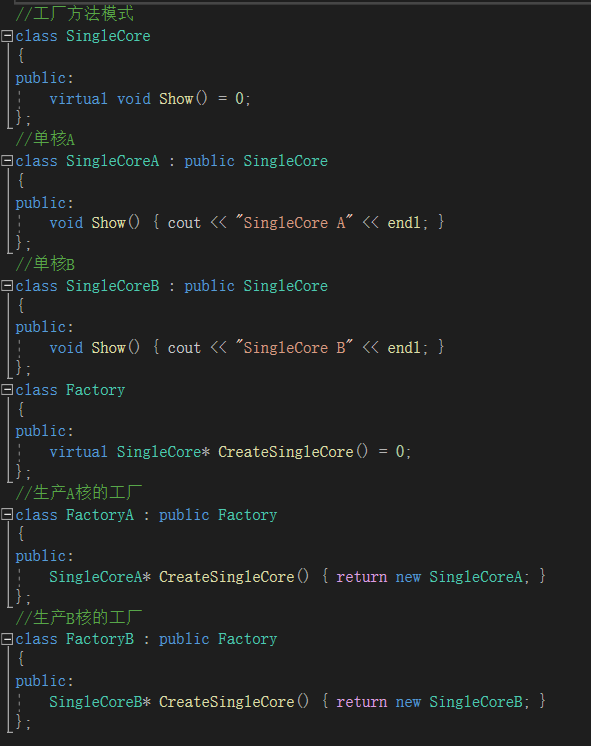
工厂模式主要解决接口选择的问题。该模式下定义一个创建对象的接口，让其子类自己决定实例化哪一个工厂类，使其创建过程延迟到子类进行。

工厂模式属于创建型模式，大致可以分为三类，简单工厂模式、工厂方法模式、抽象工厂模式。

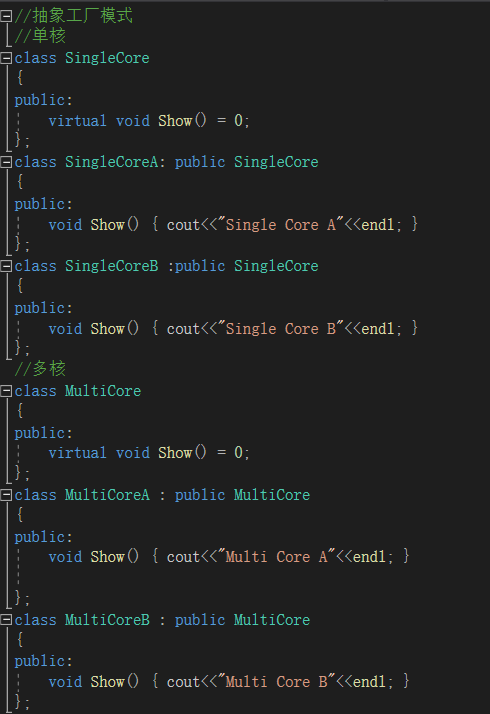
第一种简单工厂模式，它的主要特点是需要在工厂类中做判断，从而创造相应的产品。当增加新的产品时，就需要修改工厂类。

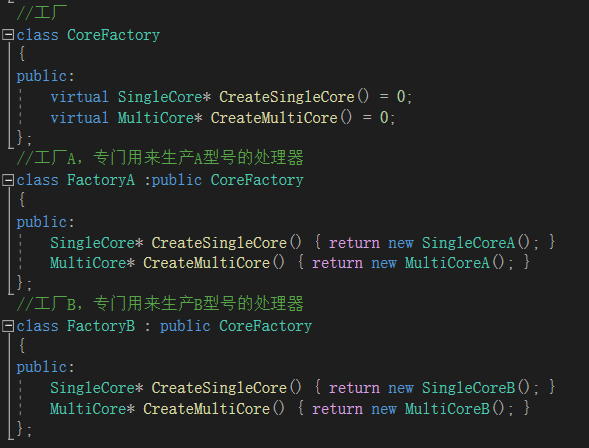


工厂方法模式，是指定义一个用于创建对象的接口，让子类决定实例化哪一个类。



抽象工厂模式为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。





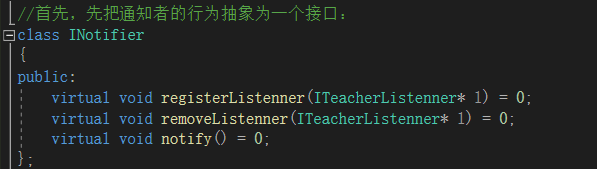
# 观察者模式

定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

例子：在教室里老师还没有来，同学都在干着各的事情，小张正在打游戏，小李正在抄作业....., 现在同学们要求班长当卧底，监视老师，当老师来了通知大家一声。然后打游戏的马上停止，抄作业的也停止。

这里班长相当于是一个通知者， 小张、小李，以及其他同学显然是监听者，他们监听了班长那的消息，一旦老师来了马上采取相关的行动。

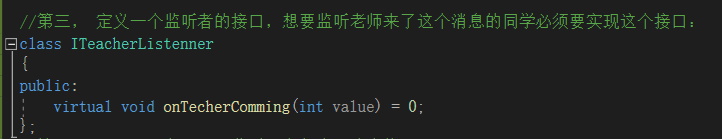
首先，先把通知者的行为抽象为一个接口：



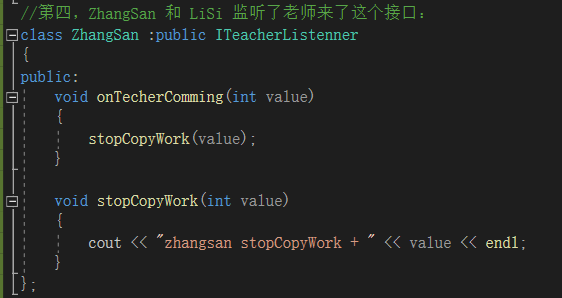
第二，然后班长作为一个具体的通知者：

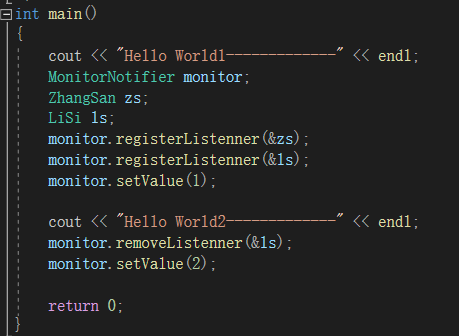


第三， 定义一个监听者的接口，想要监听老师来了这个消息的同学必须要实现这个接口：

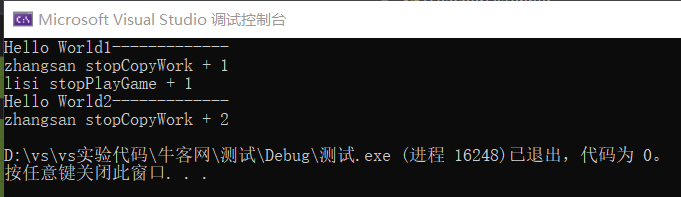


第四，ZhangSan 和 LiSi 监听了老师来了这个接口：



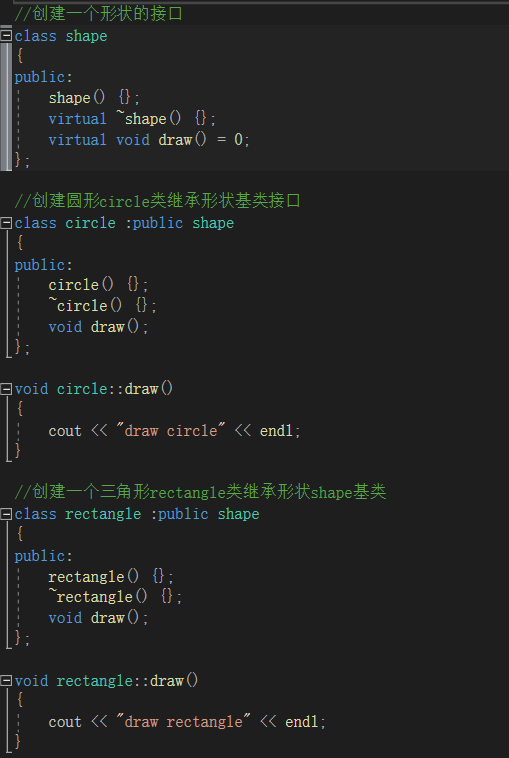


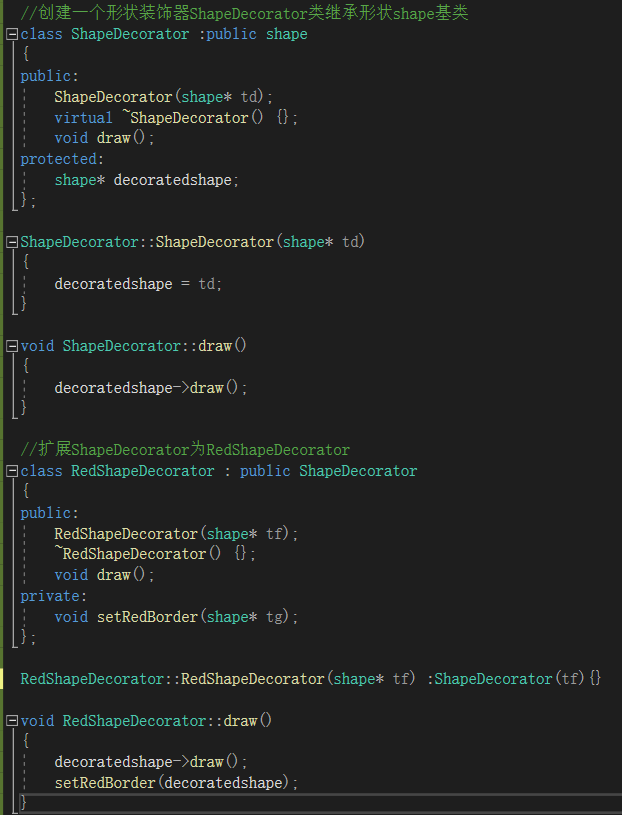
实验结果：

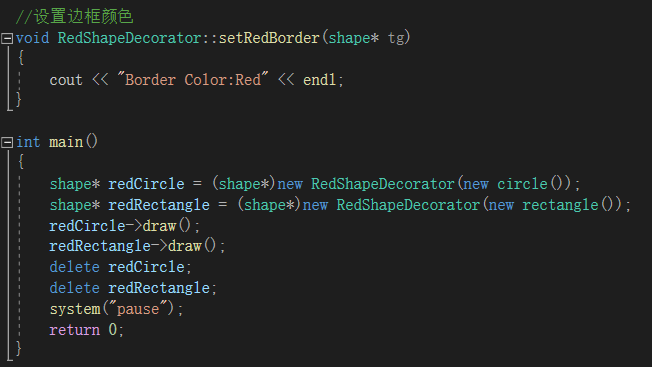


# 装饰器模式

对已经存在的某些类进行装饰，以此来扩展一些功能，从而动态的为一个对象增加新的功能。装饰器模式是一种用于代替继承的技术，无需通过继承增加子类就能扩展对象的新功能。使用对象的关联关系代替继承关系，更加灵活，同时避免类型体系的快速膨胀。







实验结果：

