1、单例模式

基本概念：保证一个类仅有一个实例，并提供一个访问它的全局访问点。

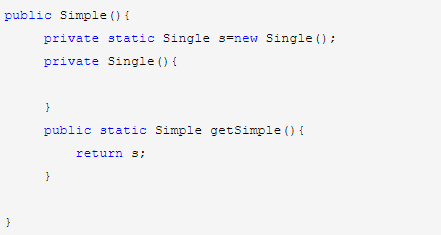
是工厂模式的特例，

饿汉式：在程序启动或单例模式类被加载的时候，单例模式的实例就已经被创建。

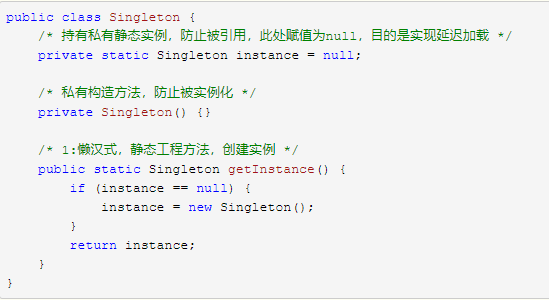
懒汉式：当程序第一次访问单例模式时，实例才创建。

当单例模式经常被用到的时候，饿汉式是一个不错的选择。

饿汉式：



懒汉式：容易引起不同步的问题，所以应该创建同步创建实例的方法

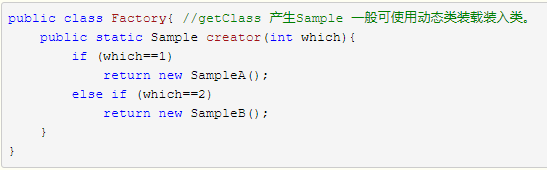


2、工厂模式

为创建对象提供过度接口，以便将创建对象的具体过程屏蔽隔离起来，达到提高灵活性的目的。

**简单工厂模式：（定义一个用于创建对象的接口）**

又称静态工厂模式。由工厂类决定哪一个产品类应当被实例化。



工厂方法模式：

不在只有一个工厂类决定哪一个产品类应当被实例化，这个决定被交给抽象工厂的子类去做。

抽象工厂角色：具体工厂角色：抽象产品角色：具体产品角色。



**简单工厂模式和工厂方法模式的区别？**

**工厂方法模式的核心是一个抽象工厂类，而不像简单工厂模式，把核心放在一个实类上，**工厂方法模式可以允许很多工厂类从抽象工厂类继承下来，从而可以在实际上成为多个简单工厂模式的综合。如果我们非常要确定一个系统只需要一个实的工厂类，那就选择简单工厂类。

**抽象工厂模式：**



抽象工厂模型中，抽象产品可能是一个或者多个，从而构成一个产品族，在只有一个产品族的情况下，抽象工厂模式实际退化到工厂方法模式。

总结：

1、简单工厂模式是由一个具体的类去创建其他类的实例，父类是相同的，父类是简单的。

2、工厂方法模式是一个有抽象父类定义的公共接口，子类负责生成具体对象。

3、抽象工厂模型提供一个创建一系列相关或相互依赖的对象接口，而无需指定他们具体的类。

3、观察者模式（发布/订阅）

观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一主题对象。这个主题对象在状态发生变化时，会通知所有观察者对象，使它们能够自动更新自己。观察者模式又叫发布-订阅(Publish/Subscribe)模式。

观察者模式何时适用？

1、当一个抽象模型有两个方面，其中一个方面依赖于另一方面。

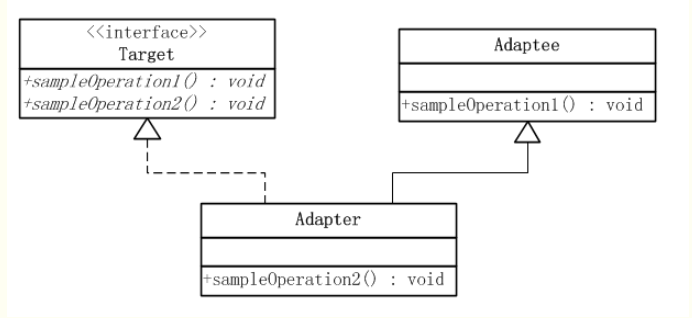
2、当对一个对象的改变需要同时改变其它对象，而不知道具体由多少对象有待改变。

3、当一个对象必须通知其他对象，而它又不能假定其他对象是谁，换言之，你不希望这些对象是紧密耦合的。让耦合的双方都依赖于抽象，而不是依赖于具体。

4、适配器（Adapter）模式

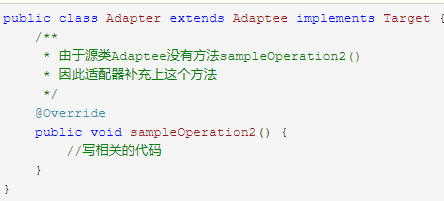
适配器模式把一个类的接口变换成客户端所期待的另一种接口，从而使原本因接口不匹配而无法在一起工作的两个类能够在一起工作。

**类适配器**：

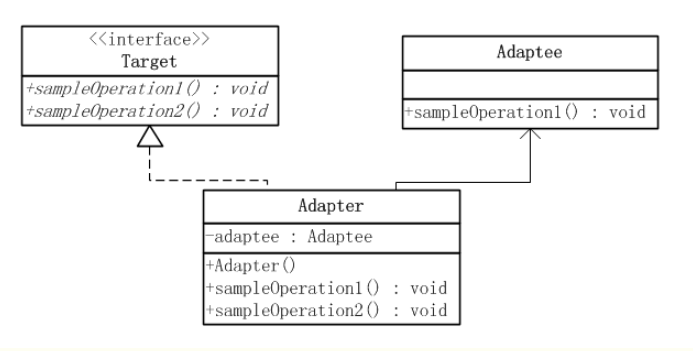


Adaptee类并没有sampleOperation2()方法，而客户端则期待这个方法。

为使客户端能够使用Adaptee类，提供一个中间环节，即类Adapter，把Adaptee的API与Target类的API衔接起来。Adapter与Adaptee是继承关系，这决定了这个适配器模式是类的



**对象适配器**：



Adaptee类并没有sampleOperation2()方法，而客户端则期待这个方法。为使客户端能够使用Adaptee类，需要提供一个包装(Wrapper)类Adapter。这个包装类包装了一个Adaptee的实例，从而此包装类能够把Adaptee的API与Target类的API衔接起来。Adapter与Adaptee是委派关系，这决定了适配器模式是对象的。



类适配器使用对象继承的方式，是静态的定义方式。

对象适配器使用的对象组合的方法，是动态的组合方式。

**适配器的优点？**

更好的复用性，更好的扩展性。

**适配器的缺点?**

过多的使用适配器，会让系统显得非常凌乱，不易进行把握，比如 ，明明看到调用的A接口，其实内部被适配成了B接口的实现，一个系统如果出现太多的这种情况，无异于一场灾难，因此如果不是很有必要，可以不使用适配器，而是直接对系统进行重构。

5、代理模式

为其他对象提供一种代理以控制对这个对象的访问。也可以说，在出发点到目的地之间有一道中间层，意为代理。

**为什么要使用？**

1、授权机制不同级别的用户对同一对象拥有不同的访问权利，如在论坛系统中，就使用Proxy进行授权机制控制，访问论坛有两种人：注册用户和游客。

2、某个客户端不能直接操作到某个对象，但又必须和那个对象有所互动。

ForumProxy权限控制：（论坛）网关，安全代理系统.

ForumPermissions中定义了各种级别权限的用户