工厂模型

工厂模式（Factory Pattern）

Java 中最常用的设计模式之一。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用一个共同的接口来指向新创建的对象。

介绍

**意图：**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，工厂模式使其创建过程延迟到子类进行。

**主要解决：**主要解决接口选择的问题。

**何时使用：**我们明确地计划不同条件下创建不同实例时。

**如何解决：**让其子类实现工厂接口，返回的也是一个抽象的产品。

**关键代码：**创建过程在其子类执行。

**优点：** 1、一个调用者想创建一个对象，只要知道其名称就可以了。

1. 扩展性高，如果想增加一个产品，只要扩展一个工厂类就可以。
2. 屏蔽产品的具体实现，调用者只关心产品的接口。

**缺点：**每次增加一个产品时，都需要增加一个具体类和对象实现工厂。

**适用范围**：一个超类（超类可以是接口、抽象类、父类）和多个子类。

**注意事项：**作为一种创建类模式，在任何需要生成复杂对象的地方，都可以使用工厂方法模式。有一点需要注意的地方就是复杂对象适合使用工厂模式，而简单对象，特别是只需要通过 new 就可以完成创建的对象，无需使用工厂模式。如果使用工厂模式，就需要引入一个工厂类，会增加系统的复杂度。

实现

我们将创建一个 *Shape* 接口和实现 *Shape* 接口的实体类。下一步是定义工厂类 *ShapeFactory*。

*FactoryPatternDemo*，我们的演示类使用 *ShapeFactory* 来获取 *Shape* 对象。它将向 *ShapeFactory* 传递信息（*CIRCLE / RECTANGLE / SQUARE*），以便获取它所需对象的类型。



实验代码ch1

|  |  |
| --- | --- |
| 2 | （1）entity包是实现接口类，类中实现接口方法。  （2）factory包是创建实例化对象的工厂。  （3）Interface\_Pachage包是接口类，里面有一个接口方法。  （4）text包是测试类，调用工厂创建接口类的实例化对象。 |

ch1这实验是通过工厂类来创建实现接口Shape类的实例化对象，返回该接口Shape,调用Shape接口下的方法。该实验暴露出一个问题（具体看代码）

注意：（1）继承接口（抽象类）时，该类是接口（抽象类）的实现类，必须实现父接口（抽象类）中的接口（抽象方法）。

（2）一个复杂类应该对应一个工厂类（用于创建该类实例化对象），该复杂类中的方法必须将放在一个接口（抽象类），并且该类继承接口（抽象接口）。工厂类创建实现类成功是返回该类的父接口或者抽象类。

（3）ch1实验是三个实现类都继承了Shape接口,三个类等价于一个类(这三个类都有共同的要实现方法，只是具体如何实现不同而已)。如果说，在一个实现类中，在添加一个方法（该方法没有接口化），通过工厂返回的接口类是无法实现该方法的。（具体看代码）

**一、一句话概括工厂模式**

**简单工厂**：一个工厂类，一个产品抽象类。

**工厂方法**：多个工厂类，一个产品抽象类。

**抽象工厂**：多个工厂类，多个产品抽象类。

**二、生活中的工厂模式**

简单工厂类：一个麦当劳店，可以生产多种汉堡。

  工厂方法类：一个麦当劳店，可以生产多种汉堡。一个肯德基店，也可以生产多种汉堡。

抽象工厂类：百胜餐饮集团下有肯德基和百事公司，肯德基生产汉堡，百事公司生成百事可乐。

资料

先看一下工厂模式是用来干什么的——属于创建模式，解决子类创建问题的。换句话来说，调用者并不知道运行时真正的类名，只知道从“Circle"可以创建出一个shape接口的类，至于类的名称是否叫'Circle"，调用者并不知情。所以真正的对工厂进行扩展的方式（防止程序员调用出错）可以考虑使用一个枚举类（防止传入参数时，把circle拼写错误）。

如果调用者参肯定类型是Circle的话，那么其工厂没有存在的意义了！

比如 IShape shape = new Circle();这样不是更好？也就是说调用者有了Circle这个知识是可以直接调用的，根据DP（迪米特法则）其实调用者并不知道有一个Circle类的存在，他只需要知道这个IShape接口可以计算圆面积，而不需要知道；圆这个类到底是什么类名——他只知道给定一个”circle"字符串的参数,IShape接口可以自动计算圆的面积就可以了！

其实在.net类库中存在这个模式的的一个典型的。但他引入的另一个概念“可插入编程协议”。那个就是WebRequest req = WebRequest.Create("[http://ccc......](http://ccc....../)");可以自动创建一个HttpWebRequest的对象，当然，如果你给定的是一个ftp地址，他会自动创建一个FtpWebRequest对象。工厂模式中着重介绍的是这种通过某个特定的参数，让你一个接口去干对应不同的事而已！而不是调用者知道了类！

比如如果圆的那个类名叫"CircleShape“呢？不管是反射还是泛型都干扰了你们具体类的生成！其实这个要说明的问题就是这个，调用者（clinet)只知道IShape的存在，在创建时给IShape一个参数"Circle",它可以计算圆的面积之类的工作，但是为什么会执行这些工作，根据迪米特法则，client是不用知道的。

我想问一下那些写笔记的哥们，如果你们知道了泛型，那么为什么不直接使用呢？干吗还需要经过工厂这个类呢？不觉得多余了吗？

如果，我只是说如果，如果所有从IShape继承的类都是Internal类型的呢？而client肯定不会与IShape一个空间！这时，你会了现你根本无法拿到这个类名！

Create时使用注册机制是一种简单的办法，比如使用一个枚举类，把功能总结到一处。而反射也是一种最简单的办法，调用者输入的名称恰是类名称或某种规则时使用，比如调用者输入的是Circle，而类恰是CircleShape，那么可以通过输入+”Shape"字符串形成新的类名，然后从字符串将运行类反射出来！

工厂的创建行为，就这些作用，还被你们用反射或泛型转嫁给了调用者（clinet)，那么，这种情况下，要工厂类何用？！