# 工厂方法模式简介

## 引入

考虑这样一个系统，使用简单工厂模式设计的按钮工厂类可以返回一个具体类型的按钮，例如圆形按钮、矩形按钮、菱形按钮等。在这个系统中如果需要增加一种新类型的按钮，例如椭圆形按钮，那么除了增加一个具体类之外还需要修改工厂类的代码。而如果使用工厂方法模式，就不需要修改工厂类的代码了，只需要为新增的

## 概念

1. 在工厂方法模式中不再提供一个统一的工厂类来创建所有的具体产品对象，而是为每一个具体产品类提供一个工厂类，提供一个与产品登记结构对应的工厂等级结构。
2. 工厂方法模式的定义如下：

为抽象产品类（可以是抽象类、普通类或接口）定义一个用于创建对象的接口（工厂接口），这个工厂接口与抽象产品类对应，在工厂接口中定义一个抽象产品类型的方法；同时为抽象产品类的子类都定义一个工厂类，这些工厂类实现工厂接口，并覆写其中的方法；在覆写的方法中返回该工厂类对应的抽象产品子类的实例。

## 结构



工厂方法模式由抽象产品、具体产品、抽象工厂、具体工厂4个角色组成。

1. Product（抽象产品）：抽象产品通常是一个接口或抽象类，是工厂方法模式所创建的对象的超类型。
2. ConcreteProduct（具体产品）：它是抽象产品的子类，一般有多个。
3. Factory（抽象工厂）：抽象工厂通常是一个接口或抽象类，它对应着抽象产品，该接口中需要定义一个返回抽象产品类型的抽象方法供具体工厂（其子类）覆写。
4. ConcreteFactory（具体工厂）：具体工厂是抽象工厂类的子类，覆写了在抽象工厂中定义的抽象方法；具体工厂对应着具体产品，有多少个具体产品就有多少个具体工厂；具体工厂实现抽象工厂并覆写其方法后，在方法中，返回该具体工厂对应着的具体产品的实例，以抽象产品类型的方法返回。

## 技巧

在工厂方法模式中，为抽象产品定义一个抽象工厂接口，该抽象产品有多少个具体产品（具体产品是抽象产品的子类），就定义多少个具体工厂（具体工厂是抽象工厂的子类），每一具体工厂对应着抽象产品的一个具体产品。

# 实现

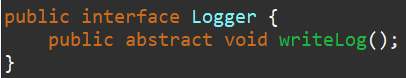
## 实例

实现说明：某系统运行日志记录器（Logger）可以通过多种途径保存着系统的运行日志，例如通过文件记录（FileLogger）或数据库记录（DataLogger），用户可以通过修改修改配置文件灵活地更换日志记录方式。

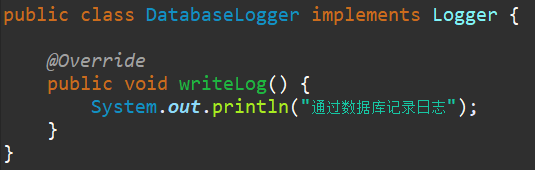


通过工厂方法模式实现：

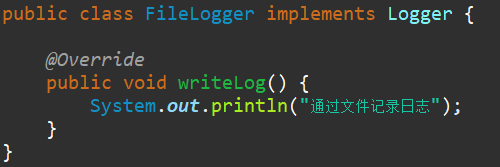
1. Logger：日志记录器，充当抽象产品角色



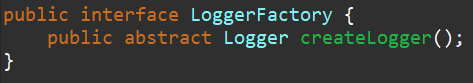
1. DatabaseLogger：数据库日志记录器，充当具体产品角色



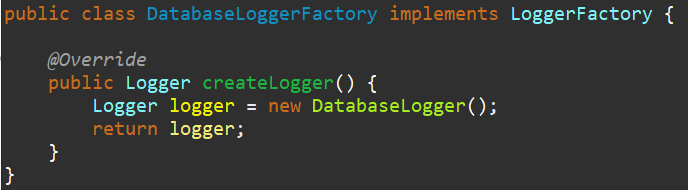
1. FileLogger：文件日志记录器，充当具体产品角色



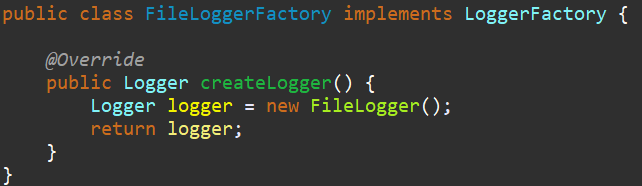
1. LoggerFactory：日志记录器工厂，充当抽象工厂角色



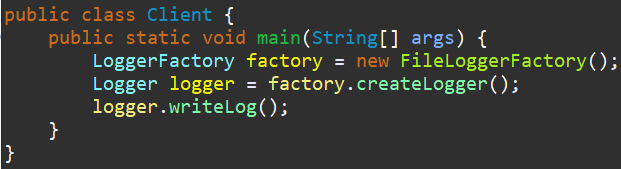
1. DatabaseLoggerFactory：数据库日志记录器工厂类，充当具体工厂角色



1. FileLoggerFactory：文件日志记录器工厂类，充当具体工厂角色



1. Client：客户端测试类，测试工厂方法模式



# 优缺点及使用环境

## 优点

1. 在工厂方法模式中，工厂方法用来创建客户所需要的产品（对象），同时还向客户端隐藏了哪种具体产品类将被实例化这一细节，用户只需要关心所需产品对应的工厂，无须关心创建细节，甚至无须知道具体产品类的类名。
2. 基于工厂角色和产品角色的多态性设计是工厂方法模式的关键。它能够让工厂自主确定创建何种产品对象，而如何创建这个对象的细节完全封装在具体工厂内部。工厂方法模式之所以又被肠胃多态工厂模式，正是因为所有的具体工厂类都具有同一抽象父类。
3. 使用工厂方法模式的另一个优点是在系统中加入新产品时无须修改抽象工厂接口和抽象产品接口，只需要为这个新添加的产品添加一个具体工厂即可。

## 缺点

1. 在添加新的具体产品类后，需要提供一个与之对应的具体工厂类，但这样系统中的类的数量将会成对增加，在一定程度上增加了系统的复杂度，有更多的类需要编译和运行，会给系统带来一些额外的开销。
2. 由于考虑到系统的扩展性，需要引入抽象层，在客户端代码中均使用抽象层进行定义，增加了系统的抽象性和理解难度。

## 适用环境

在以下情况下可以考虑使用工厂方法模式：

1. 客户端不知道它所需要的对象的类。在工厂方法模式中，客户端不需要知道具体产品类的类名，只需要知道具体产品类对应的具体工厂即可，具体产品对象将由具体工厂类创建。
2. 抽象工厂类通过其子类来指定创建哪个对象。在工厂方法模式中，对于抽象工厂类只需要提供一个创建产品的接口（抽象工厂），而由其子类（具体工厂）来确定要创建的对象，利用面向对象的多态性，在程序运行时子类对象将覆盖父类对象，从而使得系统更容易扩展。