工厂方法模式(Factory Method Pattern)：**定义一个用于创建对象的接口，让子类决定将哪一个类实例化。**

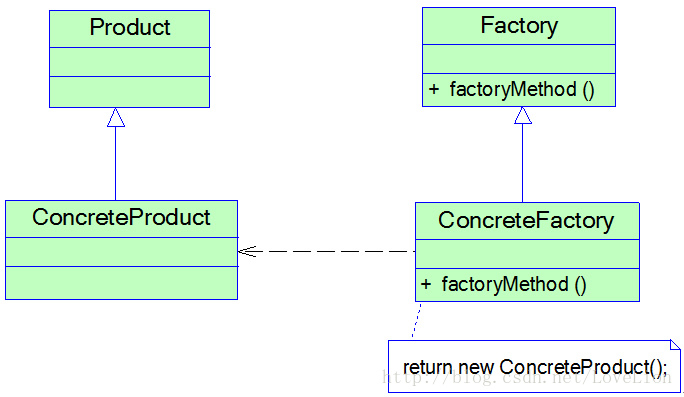
工厂方法模式让一个类的实例化延迟到其子类，是一种类创建型模式。

工厂方法模式又简称为**工厂模式**(Factory Pattern)，又可称作**虚拟构造器模式**(Virtual Constructor Pattern)或**多态工厂模式**(Polymorphic Factory Pattern)。

在简单工厂模式最大的缺点是当有新产品要加入到系统中时，必须修改工厂类，那么就意味着违反“开闭原则”。

工厂方法模式中，我们不**再提供一个统一的工厂类来创建所有的产品对象，而是针对不同的产品提供不同的工厂，系统提供一个与产品等级结构对应的工厂等级结构。**

工厂方法模式提供**一个抽象工厂接口来声明抽象工厂方法，而由其子类来具体实现工厂方法，创建具体的产品对象**。工厂方法模式结构如图所示：



图中包含如下几个角色：

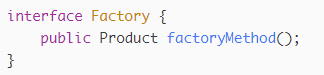
● Product（抽象产品）：它是定义产品的接口，是工厂方法模式所创建对象的超类型，也就是产品对象的公共父类。

      ● ConcreteProduct（具体产品）：它实现了抽象产品接口，某种类型的具体产品由专门的具体工厂创建，具体工厂和具体产品之间一一对应。

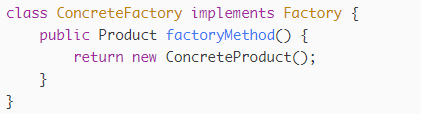
      ● Factory（抽象工厂，可以是接口、抽象类或具体类）：在抽象工厂类中，声明了工厂方法(Factory Method)，用于返回一个产品。抽象工厂是工厂方法模式的核心，所有创建对象的工厂类都必须实现该接口。

      ● ConcreteFactory（具体工厂）：它是抽象工厂类的子类，实现了抽象工厂中定义的工厂方法，并可由客户端调用，返回一个具体产品类的实例。

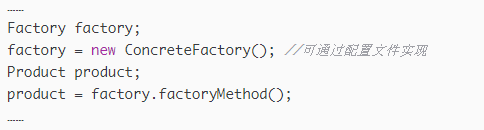
与简单工厂模式相比，工厂方法模式最重要的区别是引入了抽象工厂角色（可以是接口、抽象类或具体类）。



在抽象工厂中声明了工厂方法但并未实现工厂方法，具体产品对象的创建由其子类负责，客户端针对抽象工厂编程，可在运行时再指定具体工厂类，具体工厂类实现了工厂方法，不同的具体工厂可以创建不同的具体产品，在实际使用时，具体工厂类在实现工厂方法时除了创建具体产品对象之外，还可以负责产品对象的初始化工作以及一些资源和环境配置工作，例如连接数据库、创建文件等。其典型代码如下所示：



在客户端代码中，只需关心工厂类即可，不同的具体工厂可以创建不同的产品，典型的客户端类代码片段如下所示



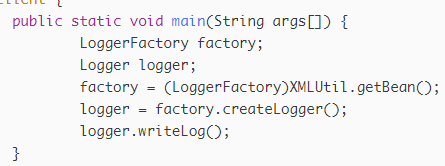
可以通过配置文件来存储具体工厂类ConcreteFactory的类名，更换新的具体工厂时无须修改源代码，系统扩展更为方便。

也可以不再使用new关键字来创建工厂对象。而是将具体工厂类的名称存储在配置文件（如XML中），通过读取配置文件获取类名字符串，再使用Java反射根据类名字符串生成对象。

配置文件读取工具类（XMLUtil）



客户端



工厂方法模式总结：

工厂方法模式是简单工厂模式的延伸，它继承了简单工厂模式的优点，同时还弥补了简单工厂模式的不足。工厂方法模式是使用频率最高的设计模式之一，是很多开源框架和API类库的核心模式。

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，同时还向客户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。它能够让工厂可以自主确定创建何种产品对象，而如何创建这个对象的细节则完全封装在具体工厂内部。工厂方法模式之所以又被称为多态工厂模式，就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，无须修改抽象工厂和抽象产品提供的接口，无须修改客户端，也无须修改其他的具体工厂和具体产品，而只要添加一个具体工厂和具体产品就可以了，这样，系统的可扩展性也就变得非常好，完全符合“开闭原则”。

缺点：

(1) 在添加新产品时，需要编写新的具体产品类，而且还要提供与之对应的具体工厂类，系统中类的个数将成对增加，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) 由于考虑到系统的可扩展性，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，且在实现时可能需要用到DOM、反射等技术，增加了系统的实现难度。

适用场景：

(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，只需要知道所对应的工厂即可，具体的产品对象由具体工厂类创建，可将具体工厂类的类名存储在配置文件或数据库中。

       (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，而由其子类来确定具体要创建的对象，利用面向对象的多态性和里氏代换原则，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。

[工厂方法模式](https://blog.csdn.net/LoveLion/article/details/17517213)

[工厂三兄弟之工厂方法模式（一）：日志记录器的设计](https://blog.csdn.net/lovelion/article/details/9306457)

[工厂三兄弟之工厂方法模式（二）：工厂方法模式概述](https://blog.csdn.net/lovelion/article/details/9306745)

[工厂三兄弟之工厂方法模式（三）：日志记录器的工厂方法模式解决方案，反射与配置文件](https://blog.csdn.net/lovelion/article/details/9307137)

[工厂三兄弟之工厂方法模式（四）：重载的工厂方法，工厂方法的隐藏，工厂方法模式总结](https://blog.csdn.net/lovelion/article/details/9307561)