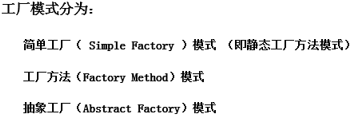
* 工厂模式：



适用性：

当一个类不知道它所必须创建的对象的类的时候。

当一个类希望由它的子类来指定它所创建的对象的时候。

当类将创建对象的职责委托给多个帮助子类中的某一个，并且你希望将哪一个帮助子类是代理者这一信息局部化的时候。

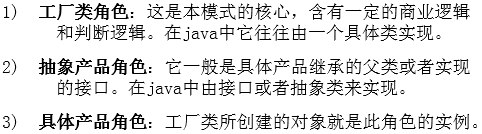
* 设计模式 – 简单工厂模式（Simple Factory）：

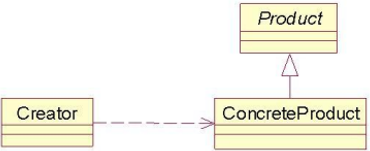
简单工厂模式是工厂方法模式的一个特例，这里给单独抽取了出来。

**优点**：客户类和工厂类分开。工厂类关注于产品的生产，客户端”消费者”任何时候需要某种产品，只需向工厂请求即可，可以无需知道每次实例化哪个类，消费者无须修改就可以接纳新产品。

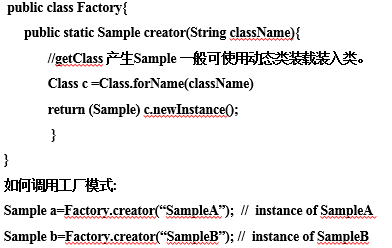
**缺点**：当产品修改或有新产品加入时，工厂类也要做相应的修改。

简单工厂模式的角色：





* 设计模式 – 简单工厂模式Demo：



* 设计模式 - 工厂方法模式（Factory Method）：

定义一个创建产品的工厂接口或抽象类，核心工厂类不再负责所有产品的创建，而是将具体创建的工作交给子类去做，成为一个抽象工厂角色，仅负责给出具体工厂类必须实现的接口，而不接触哪一个产品类应当被实例化这种细节。

工厂方法模式的**优点**：这种工厂方法模式可以允许系统在不修改具体工厂角色的情况下引入新的产品。

工厂方法模式的角色：

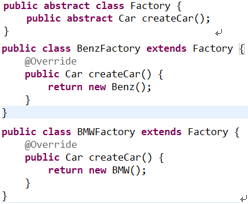
1) **抽象工厂角色**：这是工厂方法模式的核心，它与应用程序无关。是具体工厂角色必须实现的接口或者必须继承的父类。

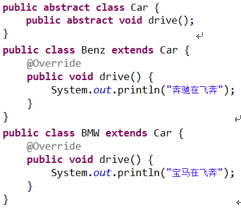
2) **具体工厂角色**：它含有和具体业务逻辑有关的代码。由应用程序调用以创建对应的具体产品的对象。

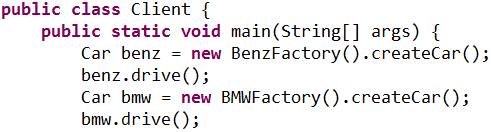
3) **抽象产品角色**：它是具体产品继承的父类或者是实现的接口。

4) **具体产品角色**：具体工厂角色所创建的对象就是此角色的实例。

* 设计模式 - 工厂方法模式Demo：







* 设计模式 - 工厂方法模式Demo2：

