# 工厂方法模式

## 一、基本概念 **1、定义：**

##### 定义一个工厂类，他可以根据参数的不同返回不同类的实例，被创建的实例通常都具有共同的父类。

##### 工厂方法模式（Factory Method Pattern）是一种创建型设计模式，它通过定义一个用于创建对象的接口，但将实际的对象创建工作推迟到子类中。这样可以在不改变客户端代码的情况下，动态地创建具体的对象实例。

### 2、组成环节：

##### 1、Factory(抽象工厂):具体工厂的父类，负责实现创建所有产品的内部逻辑，提供工厂类的公共接口

##### 2、Product(抽象类产品)：具体产品类所有对象的父类，封装了产品对象的公共方法，描述了具体工厂的公开接口

##### 3、ProductA、ProductB…(具体产品)：抽象类产品的子类，可供具体工厂类进行调用，所有被创建的对象都是某个具体类的实例。它要实现抽象产品中声明的抽象方法。

**4、Factory A、Factory B…(具体工厂)：为核心部分，抽象工厂的子类，工厂类可以被外界直接调用，创建所需对象，实现工厂方法创建实例产品。**

## 二、go语言中实现

##### 1、首先定义一个factory类型

type factory interface {  
 Get() prodcut  
}

##### 2、在定义两个具体类型的factory

type T1factory struct{}

type T2factory struct{}

##### 3、定义一个统一product的接口，并定义其实现的方法。

type prodcut interface {  
 Do()  
}

##### 4、定义product内的具体方法

type T1 struct{}  
type T2 struct{}

##### 5、对具体的product进行do的实现。

func (t T1) Do() {  
 fmt.Printf("T1 do.\n")  
}  
  
func (t T2) Do() {  
 fmt.Printf("T2 do.\n")  
}

##### 6、对具体的工厂方法进行Get的实现。

func (t T1factory) Get() prodcut {  
 return T1{}  
}

func (t T2factory) Get() prodcut {  
 return T2{}  
}

##### 7、开始进行方法的调用。

func main() {  
 var t1 T1factory  
 var t2 T2factory  
 t1.Get().Do()  
 t2.Get().Do()  
}

##### 8、结果的输出。

T1 do.

T2 do.

## 三、与简单工厂的区别

优势：

工厂类集中了所有实例（产品）的创建逻辑，一旦这个工厂不能正常工作，整个系统都会受到影响；

违背“开放 - 关闭原则”，一旦添加新产品就不得不修改工厂类的逻辑，这样就会造成工厂逻辑过于复杂。

简单工厂模式由于使用了静态工厂方法，静态方法不能被继承和重写，会造成工厂角色无法形成基于继承的等级结构。

劣势：

简单工厂在工厂类中包含了必要的逻辑判断，根据客户端的选择条件动态实例化相关的类，对于客户端来说，去除了与具体产品的依赖。