# 模板方法模式

1. **基本概念**

### **1、定义：**

模板方法（TemplateMethod）模式的定义：是一种行为设计模式， 它在基类中定义了一个算法的框架， 允许子类在不修改结构的情况下重写算法的特定步骤。

简单来说，模板方法使得子类可以不改变一个算法的结构即可重定义该算法里面的某些步骤

主要用于删减重复代码，增加代码的复用性。

### 2、组成环节：

抽象模板类（AbstractClass）：实现了一个模板方法，定义了算法的骨架子，将公用的部分记录下来，剩下的部分由子类继承。

具体模板类（ConcreteClass）：抽象模板类的子类，可以实现抽象模板类中由子类继承的部分。

## 二、go语言中实现

##### 1、首先定义一个AbstractClass的抽象模板类

type Abstrct interface {  
 Template()  
 Method()  
}

type AbstrctClass struct{}

##### 2、为抽象模板类进行template方法的实现

func (a AbstrctClass) Template() {  
 fmt.Println("Template.")  
}

##### 3、定义两个ConcreteClass的具体模板类

type ConcreteClassA struct{}

type ConcreteClassB struct{}

##### 4、为这两个ConcreteClass的具体模板类的方法进行实现

func (ca ConcreteClassA) Template() {  
 a := AbstrctClass{}  
 a.Template()  
}  
  
func (ca ConcreteClassA) Method() {  
 fmt.Println("ConcreteClassA.")  
}

func (cb ConcreteClassB) Template() {  
 a := AbstrctClass{}  
 a.Template()  
}  
  
func (cb ConcreteClassB) Method() {  
 fmt.Println("ConcreteClassB.")  
}

##### 4、通过main函数实现

func main() {  
 ca := ConcreteClassA{}  
 ca.Template()  
 ca.Method()  
 cb := ConcreteClassB{}  
 cb.Template()  
 cb.Method()  
}

##### 5、输出结果

**Template.**

**ConcreteClassA.**

**Template.**

**ConcreteClassB.**