# 桥接模式

## 一、基本概念 **1、定义：**

##### 桥接模式（Bridge），将抽象部分与它的实现部分分离，使他们都可以独立的进行变化。

##### 简言之，就是每个子类当中相似的部分把他抽出来，仅仅针对每个类中不同的抽象的部分去进行独立的变化，而不需要把所有相似的部分都写入子类当中，使用率合成聚合原则，避免了复用性，规避继承的缺陷。

### 2、组成环节：

##### Abstraction(抽象类):定义一个抽象的Operation的方法，供所有具体子类去调用。

**DefinedAbstraction（提炼抽象类）：继承Abstraction类，把所有的具体被提炼出来的抽象部分都进行实现。**

**Implementor（抽象实现类）：将所有相似的部分都进行抽象，形成的一个方法。与Abstraction是聚合的关系。**

**DefinedImplementor（具体实现类）：将Implementor具体的方法进行实现。**

## 二、go语言中实现

##### 1、首先定义一个Implementor类型，并且实现**DefinedImplementor的具体子类的方法**。

type Implementor interface {  
 OperationImp()  
}  
  
type ConcreteImplementorA struct{}  
  
type ConcreteImplementorB struct{}  
  
func (ca ConcreteImplementorA) OperationImp() {  
 println("Common partA.")  
}  
  
func (cb ConcreteImplementorB) OperationImp() {  
 println("Common partB.")  
}

##### 2、创建Abstraction类和其子类**DefinedAbstraction，并且对方法operation进行实现**。

type Abstraction interface {  
 Operation()  
 Implementor  
}  
  
type RefinedAbstraction struct {  
 Implementor  
}  
  
func (r RefinedAbstraction) Operation() {  
 println("diff part.")  
}

1. **通过main方法进行实现。**

func main() {  
 ra := RefinedAbstraction{}  
  
 ra.Implementor = ConcreteImplementorA{}  
 ra.Operation()  
 ra.OperationImp()  
  
 ra.Implementor = ConcreteImplementorB{}  
 ra.Operation()  
 ra.OperationImp()  
}

1. **输出结果**

**diff part.**

**Common partA.**

**diff part.**

**Common partB.**