# 享元模式

## 一、基本概念 **1、定义：**

##### 享元模式（Flyweight），运用共享技术有效的支持大量细粒度的对象。

##### 优势是如果一个应用程序使用了大量的对象，从而造成大量开销，享元模式可以让多个用户去共享一个对象。

##### 劣势是由于多个用户进行共享，可能会对彼此造成影响。

### 2、组成环节：

##### Flyweight(享元类):定义了接受并作用于外部状态。

##### ConcreteFlyweight(具体享元类):继承Flyweight类，并为内部状态增加存储空间。

##### FlyweightFactory(享元工厂类):用来创建并管路Flyweight对象，确保他人来合理共享Flyweight。

##### UnsharedConcreteFlyweight(非共享享元类):继承不需要共享的Flyweight子类。

## 二、go语言中实现

##### 1、首先定义Flyweight类，并列出该类型所拥有的方法。

type Flyweight interface {  
 Operation()  
}

##### 2、再定义ConcreteFlyweight类，并列出该类型所拥有的方法和具体实现方法。

type ConcreteFlyweight struct {  
 name string  
}  
  
func (c \*ConcreteFlyweight) Operation() {  
 fmt.Println("share flyweight:" + c.name)  
}

##### 3、对UnsharedConcreteFlyweight方法进行定义，并且对中各类的实现方法进行讲述。

type UnsharedConcreteFlyweight struct {  
 name string  
}  
  
func (c \*UnsharedConcreteFlyweight) Operation() {  
 fmt.Println("Unshare flyweight:" + c.name)  
}

##### 4、对FlyweightFactory方法进行定义，并且对中各类的实现方法进行讲述。

type FlyweightFactory struct {  
 m map[string]struct{}  
}  
  
func (f \*FlyweightFactory) GetFlyweight(key string) ConcreteFlyweight {  
 if \_, ok := f.m[key]; !ok {  
 f.m[key] = struct{}{}  
 }  
 return ConcreteFlyweight{name: key}  
}

##### 5、通过main方法进行client端调用。

func main() {  
 f := FlyweightFactory{m: make(map[string]struct{})}  
  
 x := f.GetFlyweight("x")  
 x.Operation()  
  
 y := f.GetFlyweight("y")  
 y.Operation()  
  
 uf := UnsharedConcreteFlyweight{name: "z"}  
 uf.Operation()  
  
 newx := f.GetFlyweight("x")  
 newx.Operation()  
}

##### 6、输出结果

**share flyweight:x**

**share flyweight:y**

**Unshare flyweight:z**

**share flyweight:x**

**（newx和x为同一个）**