# 备忘录模式

1. **基本概念**

### **1、定义：**

备忘录（Memento）模式的定义：在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态。这样以后就可将对象恢复到原先的状态。

简单来说，备忘录模式就是将重要的信息进行备份，在需要使用的时候进行读取，恢复到以前的状态。

### 2、组成环节：

发起人类（Originator）：负责创建一个备忘录类，并且用其记录自己当前的状态，在需要时可以恢复当前的状态。

备忘录类（Memento）：负责发起人类的内部存储，而且可以防止其他人的访问。

保存类（Caretaker）：负责保存备忘录类，以便后续恢复使用。

## 二、go语言中实现

##### 1、首先定义一个Memento的备忘录类

type Memento interface {  
 store(state string)  
 read() string  
}

##### 2、再定义一个Caretaker的保存类，并且进行方法的实现

type Caretaker struct {  
 state string  
}  
  
func (c \*Caretaker) store(state string) {  
 c.state = state  
}  
  
func (c \*Caretaker) read() string {  
 return c.state  
}

##### 3、最后通过定义一个Originator的发起人类型 并且进行方法的实现

type Originator struct {  
 state string  
}  
  
func (o \*Originator) Get() string {  
 return o.state  
}  
  
func (o \*Originator) Storage() Memento {  
 c := &Caretaker{}  
 c.store(o.state)  
 return c  
}  
  
func (o \*Originator) Restore(m Memento) {  
 o.state = m.read()  
}

##### 4、通过main函数实现

func main() {  
 o := Originator{state: "start"}  
 println(o.Get())  
  
 c := o.Storage()  
 o.state = "end"  
 println(o.Get())  
  
 o.Restore(c)  
 println(o.Get())  
}

##### 5、输出结果

**start**

**end**

**start**