# 中介者模式

## 一、基本概念 **1、定义：**

##### 中介者模式（Mediator），使用一个中介对象来封装一系列的对象交互。中介者使各对象不需要显式的相互引用，从而使其耦合松散，而且可以独立的改变它们之间的交互。

##### 优势是该模式很容易在系统中应用，渐少各个Colleague之间的耦合，站在一个更宏观的角度去看系统。

##### 劣势是由于ConcreteMediator的集中控制化，使得中介者会变得更加复杂。

### 2、组成环节：

##### Mediator(抽象中介类):定义了对象到中介者对象的接口。

##### ConcreteMediator(具体中介类):实现抽象类的方法，它需要知道所有具体对象类，从而转发命令。

##### Colleague(抽象对象类):定义了抽象的对象类具体传达命令的接口。

##### ConcreteColleague(具体对象类):每个具体对象的自己行为，不知道其他对象情况，依靠中介类来进行沟通。

## 二、go语言中实现

##### 1、首先定义Mediator类，并列出该类型所拥有的方法。

type mediator interface {  
 Set(Colleague1 colleague, Colleague2 colleague)  
 Send(message string, Colleague colleague)  
}

##### 2、再定义Colleague类，并列出该类型所拥有的方法。

type colleague interface {  
 setMediator(m mediator)  
 Send(message string)  
 Notify(message string)  
}

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

type ConcreteMediator struct {  
 c1 colleague  
 c2 colleague  
}  
  
func (c \*ConcreteMediator) Set(Colleague1 colleague, Colleague2 colleague) {  
 c.c1 = Colleague1  
 c.c2 = Colleague2  
}  
  
func (c \*ConcreteMediator) Send(message string, Colleague colleague) {  
 if c.c1 == Colleague {  
 c.c2.Notify(message)  
 } else {  
 c.c1.Notify(message)  
 }  
}

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

type ConcreteColleagueA struct {  
 m mediator  
}  
  
func (ca \*ConcreteColleagueA) Notify(message string) {  
 println("ca 收到信息：" + message)  
}  
  
func (ca \*ConcreteColleagueA) Send(message string) {  
 ca.m.Send(message, ca)  
}  
  
func (ca \*ConcreteColleagueA) setMediator(m mediator) {  
 ca.m = m  
}  
  
type ConcreteColleagueB struct {  
 m mediator  
}  
  
func (cb \*ConcreteColleagueB) Send(message string) {  
 cb.m.Send(message, cb)  
}  
  
func (cb \*ConcreteColleagueB) setMediator(m mediator) {  
 cb.m = m  
}  
  
func (cb \*ConcreteColleagueB) Notify(message string) {  
 println("cb 收到信息：" + message)  
}

##### 5、通过main方法进行client端调用。

func main() {  
 ca := ConcreteColleagueA{}  
 cb := ConcreteColleagueB{}  
  
 m := ConcreteMediator{}  
 m.Set(&ca, &cb)  
 ca.setMediator(&m)  
 cb.setMediator(&m)  
  
 ca.Send("12345")  
 cb.Send("67890")  
}

##### 6、输出结果

**cb 收到信息：12345**

**ca 收到信息：67890**