设计模式-中介者模式

参考

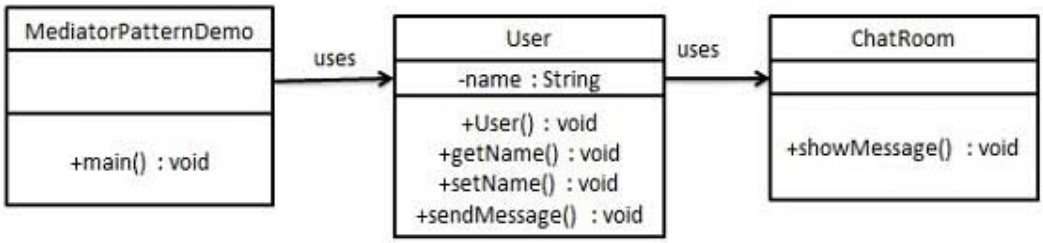
- (1条消息) Go设计模式(28)-中介者模式_程序员麻辣烫的博客-CSDN博客

中介者模式是用来降低多个对象和类之间的通信复杂性。这种模式提供了一个中介类，该类通常处理不同类之间的通信，并支持松耦合，使代码易于维护。中介者模式属于行为模式。

使用场景

- 系统中对象之间存在比较复杂的引用关系，导致它们之间的依赖关系结构混乱而且难以复用该对象
- 通过一个中间类来封装多个类中的行为，而不像生成太多的子类。

Demo分析



> 分析：
>
> 我们通过聊天室实例来演示中介模式。实例中，多个用户可以向聊天室发送消息，聊天室向所有的用户显示消息。我们将创建两个类ChatRoom和User。User对象使用ChartRoom方法来分享它们的消息。

Go实现

```
package main

import (
    "fmt"
    "time"
)
```

```go
/**
* 中介类
 */
type ChatRoom struct{}

func (c ChatRoom) Send(user *User, message string) {
    fmt.Printf("%v %s say: %s\n", time.Now(), user.GetName(), message)
}

/**
* 用户类
 */
type User struct {
    Name string
}

func (u *User) GetName() string { return u.Name }

func (u *User) SetName(name string) { u.Name = name }

func (u *User) send(message string) {
    c := ChatRoom{}
    c.Send(u, message)
}

func MediatorPatternDemo() {
    rebort := &User{"Rebort"}
    john := &User{"John"}
    rebort.send("Hello, John")
    john.send("Hello, Rebot")
}

func main() {
    MediatorPatternDemo()
}
```

输出

```
2022-08-16 16:22:01.7938729 +0800 CST m=+0.003103101 Rebort say: Hello, John
2022-08-16 16:22:01.8102581 +0800 CST m=+0.019488301 John say: Hello, Rebot
```

Python实现

```python
import time

class ChatRoom:
  @classmethod
  def SendMessage(self, user, message):
    print(f"{time.time()} {user.getName()} say: {message} ")

class User:
  def __init__(self, name):
    self.name = name
```

```
  def getName(self):
    return self.name
  def setName(self, name):
    self.name = name
  def SendMessage(self, message):
    ChatRoom.SendMessage(self, message)

def test():
  rebort = User("rebort")
  john = User("john")
  rebort.SendMessage("hello john")
  john.SendMessage("hi rebort")

if __name__ == '__main__':
  test()
```

输出

```
1660638579.3382804 rebort say: hello john
1660638579.3382804 john say: hi rebort
```

小结

中介者模式其实就是将类中的方法独立出去，统一进行处理。而独立出去专门处理的部分就是中介者的核心。