设计模式-命令模式

参考

- Go设计模式(26)-命令模式*程序员麻辣烫的博客-CSDN博客*go 命令模式
- https://www.runoob.com/design-pattern/command-pattern.html

命令模式时一种数据驱动的设计模式，它属于行为型模式。请求以命令的形式包裹在对象中，并传给调用对象。调用对象寻找可以处理该命令的合适的对象，并把该命令传给相应的对象，该对象执行命令。

使用场景

认为是命令的地方都可以使用命令模式，例如GUI中的按钮，模拟CMD

Demo分析

下面模拟游戏中的一种通用架构：客户端的各种命令请求被服务端存储起来，然后服务器的单线程顺序的去执行这些请求。

Go实现

```
package main

import "fmt"

/**
* 命令接口
 */
type Command interface {
    Execute()
}

/**
* 移动命令
 */
type MoveCommand struct {
    x, y int
}

func (m *MoveCommand) Execute() {
```

```
    fmt.Printf("Move to %d, %d\n", m.x, m.y)
}

/**
* 攻击命令
 */
type AttackCommand struct {
    skill string
}

/**
* 攻击的动作执行
 */
func (a *AttackCommand) Execute() {
    fmt.Printf("Attack with %s\n", a.skill)
}

func AddCommand(action string) Command {
    if action == "attack" {
        return &AttackCommand{skill: "sword"}
    } else {
        return &MoveCommand{x: 1, y: 2}
    }
}

func test() {
    commandList := make([]Command, 0)
    commandList = append(commandList, AddCommand("attack"))
    commandList = append(commandList, AddCommand("move"))
    commandList = append(commandList, AddCommand("attack"))
    for _, c := range commandList {
        c.Execute()
    }
}

func main() {
    test()
}
```

输出

```
Attack with sword
Move to 1, 2
Attack with sword
```

Python实现

```
class Command(object):
  """
    ## 命令的接口
  """
  def Execute(self):
    pass
```

```
class MoveCommand(Command):
  """
    ## 移动命令
  """
  def __init__(self, x, y):
    self.x = x
    self.y = y
  def Execute(self):
    print(f"Move to {self.x},{self.y}")


class AttackCommand(Command):
  """
    ## 攻击命名
  """
  def __init__(self, skill):
    self.skill = skill
  def Execute(self):
    print(f"Attack with {self.skill}")

def AddCommand(action):
  """
    ## 添加命令
  """
  if action == "attack":
    return AttackCommand("crush")
  else :
    return MoveCommand(1,2)

def test():
  commandList = []
  commandList.append(AddCommand("attack"))
  commandList.append(AddCommand("move"))
  commandList.append(AddCommand("attack"))
  for command in commandList:
    command.Execute()

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

输出

```
Attack with crush
Move to 1,2
Attack with crush
```

小结

感觉命令模式与工厂模式类似，感觉不同的是命令模式强调的是生成一种动作的对象，生成的一系列的对象用于动作的执行。同时可以延时去执行动作