设计模式-模板模式

参考

- (1条消息) Go设计模式(19)-模板模式_程序员麻辣烫的博客-CSDN博客
- https://www.runoob.com/design-pattern/template-pattern.html

---

模板模式感觉就是类中继承的思想。在模板模式中，一个抽象类公开定义了执行它的方法的模板。它的子类可以俺需要重写方法实现，但调用将以抽象类中定义的方式进行。这种类型的设计模式属于行为模式。

使用场景

- 有多个子类共有的方法，且逻辑相同。
- 重要的、复杂的方法，可以考虑作为模板方法

Demo分析

> 模板模式的话我们平时接触的应该也比较多，我们这里通过实现一种问答的形式来写一个小的case。

Go实现

> go的话因为语言本身不是考虑面向对象的思想，而是采用组合的思想，所以暂且按照下面的方法写。

```
package main

import "fmt"

/**
* 问答
 */
type Talking struct {
    Answer  func()
    Answer2 func()
}

func (t *Talking) say() {
    fmt.Printf("Hello World\n")
```

```
    t.Answer()
    fmt.Printf("hi j\n")
    t.Answer2()
}

/**
* 相当于是子类
 */
type Person struct {
    Talking
}

func (p *Person) Answer() {
    fmt.Printf("hi\n")
}
func (p *Person) Answer2() {
    fmt.Printf("what are you saying\n")
}
func test() {
    talking := &Person{}
    talking.Talking.Answer = talking.Answer
    talking.Talking.Answer2 = talking.Answer2
    talking.say()
}

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

输出

```
Hello World
hi
hi j
what are you saying
```

Python实现

```
class Talking:
  """
    ## 父类
  """
  def __init__(self):
    pass
  def Answer(self):
    pass
  def Answer2(self):
    pass
  def say(self):
    print("Hello World")
    self.Answer()
    print("hi j")
    self.Answer2()
```

```
class Person(Talking):
  def Answer(self):
    print("hi ")
  def Answer2(self):
    print("what are you saying")

def test():
  person = Person()
  person.say()

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

输出

```
Hello World
hi
hi j
what are you saying
```

小结

根据不同的语言特征，像go这种不是面向对象思想的一般其实模板模式使用的较少，不过可以使用其他的方法也能达到一样的效果。而像JAVA这些面向对象思想的语言，模板模式的使用其实就是家常便饭了。