

作业

1、实现对圆形、三角形、长方形求面积。

```
1 package main
2
3 package main
4
5 import (
6     "fmt"
7     "math"
8 )
9
10 type Interface interface {
11     Area() float32 // 只要是图形，就必须实现面积方法
12 }
13
14 // 圆形
15 type Circle struct {
16     r float32 // 半径
17 }
18
19 // 构造函数
20 func NewCircle(r float32) *Circle {
21     return &Circle{r}
22 }
23
24 func (c *Circle) Area() float32 {
25     return math.Pi * c.r * c.r
26 }
27
28 // 三角形
29 type Triangle struct {
30     base, height float32
31 }
32
33 // 构造函数
34 func NewTriangle(base, height float32) *Triangle {
35     return &Triangle{base, height}
36 }
37
38 func (t *Triangle) Area() float32 {
39     return t.base * t.height / 2
40 }
41
42 // 长方形
43 type Rectangle struct {
44     width, height float32
45 }
46
47 // 构造函数
48 func NewRectangle(width, height float32) *Rectangle {
```

```

49     return &Rectangle{width, height}
50 }
51
52 func (r *Rectangle) Area() float32 {
53     return r.width * r.height
54 }
55
56 func main() {
57     // 圆形
58     c := NewCircle(2)
59     fmt.Println(c, c.Area())
60     // 三角形
61     t := NewTriangle(3, 4)
62     fmt.Println(t, t.Area())
63     // 矩形
64     r := NewRectangle(4, 5)
65     fmt.Println(r, r.Area())
66
67     var s Interface = c
68     fmt.Println(s.Area())
69 }

```

2、利用第1题，构造3个以上图形，至少圆形、三角形、矩形各有一个，对上题的图形按照面积降序排列

为了方便，每一个形状内部都应该记录一个计算好的面积值，而不是每次调用Area()方法都要计算一次。怎样才能让所有形状都具有记录面积的属性呢？

```

1  package main
2
3  import (
4      "fmt"
5      "math"
6      "sort"
7  )
8
9  type Interface interface {
10     Area() float32 // 只要是图形，就必须实现面积方法
11 }
12
13 type Shape struct {
14     area float32 // 小写属性对包外不可见
15 }
16
17 // 圆形
18 type Circle struct {
19     Shape
20     r float32 // 半径
21 }
22
23 // 构造函数
24 func NewCircle(r float32) *Circle {
25     // return &Circle{Shape{0}, r}

```

```

26     return &Circle{r: r}
27 }
28
29 func (c *Circle) Area() float32 {
30     if c.area == 0 { // 如果是零值，就重新计算面积
31         c.area = math.Pi * c.r * c.r
32     }
33     return c.area
34 }
35
36 // 三角形
37 type Triangle struct {
38     Shape
39     base, height float32
40 }
41
42 // 构造函数
43 func NewTriangle(base, height float32) *Triangle {
44     return &Triangle{base: base, height: height} // Shape零值
45 }
46
47 func (t *Triangle) Area() float32 {
48     if t.area == 0 {
49         t.area = t.base * t.height / 2
50     }
51     return t.area
52 }
53
54 // 长方形
55 type Rectangle struct {
56     Shape
57     width, height float32
58 }
59
60 // 构造函数
61 func NewRectangle(width, height float32) *Rectangle {
62     return &Rectangle{width: width, height: height}
63 }
64
65 func (r *Rectangle) Area() float32 {
66     if r.area == 0 {
67         r.area = r.width * r.height
68     }
69     return r.area
70 }
71
72 func main() {
73     // 圆形
74     c := NewCircle(2)
75     fmt.Println(c, c.Area())
76     // 三角形
77     t := NewTriangle(3, 4)
78     fmt.Println(t, t.Area())
79     // 矩形
80     r := NewRectangle(4, 5)

```

```
81     fmt.Println(r, r.Area())
82
83     // var s Interface = c
84     // fmt.Println(s.Area())
85
86     // 排序需先放到一个序列中
87     var shapes = []Interface{c, t, r}
88     fmt.Println(shapes)
89
90     sort.Slice(shapes, func(i, j int) bool {
91         return shapes[i].Area() > shapes[j].Area() // 换成>就成了降序
92     })
93     fmt.Println(shapes)
94
95     for _, v := range shapes {
96         fmt.Printf("%T\t%.2f\n", v, v.Area())
97     }
98 }
```

