作业

1、实现对圆形、三角形、长方形求面积。

```go
package main

package main

import (
    "fmt"
    "math"
)

type Interface interface {
    Area() float32 // 只要是图形，就必须实现面积方法
}

// 圆形
type Circle struct {
    r float32 // 半径
}

// 构造函数
func NewCircle(r float32) *Circle {
    return &Circle{r}
}

func (c *Circle) Area() float32 {
    return math.Pi * c.r * c.r
}

// 三角形
type Triangle struct {
    base, height float32
}

// 构造函数
func NewTriangle(base, height float32) *Triangle {
    return &Triangle{base, height}
}

func (t *Triangle) Area() float32 {
    return t.base * t.height / 2
}

// 长方形
type Rectangle struct {
    width, height float32
}

// 构造函数
func NewRectangle(width, height float32) *Rectangle {
```

```
    return &Rectangle{width, height}
}

func (r *Rectangle) Area() float32 {
    return r.width * r.height
}

func main() {
    // 圆形
    c := NewCircle(2)
    fmt.Println(c, c.Area())
    // 三角形
    t := NewTriangle(3, 4)
    fmt.Println(t, t.Area())
    // 矩形
    r := NewRectangle(4, 5)
    fmt.Println(r, r.Area())

    var s Interface = c
    fmt.Println(s.Area())
}
```

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?

```
package main

import (
    "fmt"
    "math"
    "sort"
)

type Interface interface {
    Area() float32 // 只要是图形，就必须实现面积方法
}

type Shape struct {
    area float32 // 小写属性对包外不可见
}

// 圆形
type Circle struct {
    Shape
    r float32 // 半径
}

// 构造函数
func NewCircle(r float32) *Circle {
    // return &Circle{Shape{0}, r}
```

```go
    return &Circle{r: r}
}

func (c *Circle) Area() float32 {
    if c.area == 0 { // 如果是零值，就重新计算面积
        c.area = math.Pi * c.r * c.r
    }
    return c.area
}

// 三角形
type Triangle struct {
    Shape
    base, height float32
}

// 构造函数
func NewTriangle(base, height float32) *Triangle {
    return &Triangle{base: base, height: height} // Shape零值
}

func (t *Triangle) Area() float32 {
    if t.area == 0 {
        t.area = t.base * t.height / 2
    }
    return t.area
}

// 长方形
type Rectangle struct {
    Shape
    width, height float32
}

// 构造函数
func NewRectangle(width, height float32) *Rectangle {
    return &Rectangle{width: width, height: height}
}

func (r *Rectangle) Area() float32 {
    if r.area == 0 {
        r.area = r.width * r.height
    }
    return r.area
}

func main() {
    // 圆形
    c := NewCircle(2)
    fmt.Println(c, c.Area())
    // 三角形
    t := NewTriangle(3, 4)
    fmt.Println(t, t.Area())
    // 矩形
    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  }
```