Go语言面向对象

1.基础知识

- go语言仅支持封装，不支持继承和多态
- go语言没有class，只有struct
- 结构体的创建

```
//树的节点
type treeNode struct {
    value       int
    left, right *treeNode
}
```

2.类对象的创建

我们这里采用多种方法进行创建，有如下的结果，以一个树的节点为例

```
var root treeNode
root = treeNode{value: 3} //这里直接进行创建
root.left = &treeNode{}
root.right = &treeNode{5, nil, nil} //这里会按照顺序然后填写进去
root.right.left = new(treeNode)      //也可以利用new关键字进行书写 当然如果是在C/C++当中，则我们使用的是right->left的用法
nodes := []treeNode{
    {value: 5},
    {},
    {6, nil, &root},
}
```

不同于CC++的语法，在go语言中，我们在函数中可以返回一个局部变量的地址，仍然是我们以树节点为例，我们有以下可以创建的用法

```
func createNode(value int) *treeNode {
    return &treeNode{value, nil, nil}
}
```

则我们可以通过函数进行创建，则我们这个结构体是在栈上进行分配还是在堆上进行分配的呢？

要注意，垃圾回收机制是在堆上进行的，那么这样，如果我们返回的是局部变量的地址，编译器认为，我们需要在外部调用，则此时，go会在堆上分配这个变量，若此时不是返回地址，则会在栈区分配，当这个函数执行结束之后，栈区释放，变量也就结束了

3.为结构体创建成员函数（方法）

go语言的方法定义是在类外部的，我们尝试定义一个打印函数，打印当前节点的值

```
//类方法的定义
//这里其实本质上是把this传递进来，仍然是传值，这里改变之后不能改变外部的值
func (node treeNode) print() {
    fmt.Println(node.value)
}
```

但是有一个问题，go语言所有的参数都是传值的，那我们尝试书写一个setValue进行尝试

```
//go语言所有的参数都是传值的
func (node treeNode) setValue() {
    node.value = 0 //这里改变其值
}
```

这里会发现，无法改变节点的值，那我们使用什么方法进行更改，则有如下的尝试

```
func (node *treeNode) setValue() {
    node.value = 0 //这里改变其值
}
```

这里如果利用指针的话，就可以完美的解决问题

4.nil的使用

在go语言当中nil指针也可以调用方法，如下

```
var s *treeNode
s.setValue(200) //这里是空指针
s = &root
s.setValue(300)
s.print()
```

5.实现树的中序遍历

```
func (node *treeNode) traverse() { //中序遍历
   if node == nil {
      return
   }
   node.left.traverse()
   node.print()
   node.right.traverse()
}
```

※6.值接收者和指针接收者 （即实现类方法中的第一个括号内部的内容）

- 要改变内容必须使用指针接收者
- 结构过大也考虑使用指针接收者
- 一致性：如果有指针接收者，最好都是指针接收者

其实类似于java中this这样的指针，这都是指针接收者，也就是引用改变，但是对于go语言，为值接收者，则不加*则不表示一个引用，而全部是值调用，这样防止了错用，也是一种很优秀的东西。