使用内嵌来拓展已有类型

1.内嵌的用法

对于我们刚才的形式，使用了组合的形式进行类的拓展，那么我们这里采用内嵌的形式，有如下操作

```
type myTreeNode struct {
    *Treenode.TreeNode //Embedding 内嵌
}
```

这里我们再次重新写这个新的遍历函数

```
func (myNode *myTreeNode) postOrder() {
    if myNode == nil || myNode.TreeNode == nil { //如果此时是空的node
        return
    }
    node := myTreeNode{myNode.Left}
    node.postOrder()
    treeNode := myTreeNode{myNode.Right}
    treeNode.postOrder()
    myNode.Print()
}
```

这里非常类似与继承，这样我们的myTreeNode就含有了原本我们创建的Node里面的成员函数和变量，这样我们就实现了更优秀的拓展功能

2.与继承的区别和相同的地方

可以进行重载

```
func (mynode *myTreeNode) Traverse() {
    fmt.Println("This method is shadowed.")
}
```

我们会发现，创建一个myTreeNode调用Traverse则会打印这句话，但是如果我们调用一个myTreeNode.Treenode，我们就会发现，调用的是原来的那个遍历函数。

这样与继承的区别在这里，我们会发现，使用继承的情况，在java中可以将子类赋值给父类，但是在go语言中不能这样使用，这样就证明在go语言中并不存在所谓的继承语法，仅仅是一个“语法糖”，方便大家进行使用。

总结

对类进行拓展

1.定义别名

2.使用组合

3.使用内嵌