链滴

多态在 Go 中的体现

作者：moddemod

原文链接：https://ld246.com/article/1566926258471

来源网站：链滴

写在前面的话

首先Go这门语言是没有面向对象的设计概念的没有多态继承封装这些概念的，但是其设计灵魂的伸处依然渗透着面向对象的思想在里面，只是换了一种表达。

Java

为什么这么说呢？ Java是号称面向对象的语言，我们先从面向对象的设计里感受下多态..

首先我们从Java的一个例子出发

```
public interface JustInterface {

    void justMethod(Object object);

}

public class JustInterface Impl implements JustInterface{

    @Override
    public void justMethod(Object object) {
        // TODO Auto-generated method stub
        do something...
    }

}

// 在Main.class里可以这样声明和初始化:
public static void main( String[] args ) {

    JustInterface justInterface = new JustInterfaceImpl();
    justInterface.justMethod;

}
```

这里先定义了一个接口，我们都知道接口必须要去实现后才能调用，所以第二部分写了一个实现类，后再主方法里面new了一个实现类，我们创建了一个对象，但是发现它的返回值却指向了它的接口，种为什么会成功？ 因为实现类实现了该接口，所以调用成功了。

我们再来看，在生活中，我们可以说这是一个男人或者女人，但是同时我们说这是一个人没问题吧，是很正常的逻辑，也就是说你说一个男人是一个人，显然这是没有什么问题的，这就是一种多态的体。

为什么这么说呢?

显然，我们创建了一个接口，因为接口是比较抽象的一个概念，在Java中她可以有很多实现类，然而于每一个实现了该接口的类，我们都可以把创建这个实现类的对象指向该接口，这就回到说一个男人一个人的问题上了，这也是没有任何问题的。

在面向对象的思维里，其实很多时候我们是不关心怎么去做的，我们都希望调一个接口，然后就完成任务，这是我们都期望的。

Go

回到Go里面来说，

```
package main

import "fmt"

// 定义一个数据写入器
type DataWriter interface {
    WriteData(data interface{}) error
}


// 定义文件结构用于实现DataWriter
type file struct {

}


// 这样就算file实现了DataWriter接口
func (d *file) WriteData(data interface{}) error {
    // 模拟写入数据
    fmt.Println("WriteData:", data)
    return nil
}

func main() {

    // 实例化file
    f := new(file)
    // 声明一个DataWriter的接口
    var writer DataWriter
    // 将接口赋值f，也就是*file类型
    writer = f
    writer.WriteData("123")
    //writer := DataWriter(new(file))
}
```

我们先创建了一个接口类型，然后定义了一个file类型，同时让file类型去实现这个接口类型，在main法中，我们同样new了一个file类型的内存空间，然后我们声明了一个DataWriter接口类型，关键的码就是这一句writer = f, 很多人好奇为什么两个数据类型不一样可以直接赋值？

其实这也是多态的体现，由于f是*file类型，也就是指向new的那块内存空间，将f赋值给writer以后，rite就可以直接调用WriteData()方法，这也可以理解为实现指向接口！