7.5-7.6 golang中的面向对象特性

面向对象编程的思想是将某种数据结构抽象成数据和行为，并将数据和行为实现在对象当中，在编程过程中实现了数据和行为的完全分离。

面向对象编程的特点一般包含：

1. 封装：一个类型分别由属性和函数组成
2. 多态：针对不同类的相同操作可以通过实现一个接口来完成
3. 继承：不同类之间可以相互继承属性和函树，一个对象基于另一个对象，继承另一个对象的属性和行为函数
4. **对象的封装：**

**1.1 Java等一般语言：**

Java和其他面向对象的语言都是通过类（class）层次结构来描述对象，类中不仅包含对象的属性，同时也有对象的行为，也就是各种函数。比如针对一辆车，对象车可能有：价格、速度和颜色几种属性，描述车的行为可能有：加速，刹车。行驶等：

public class Car {

public double price;

public string color;

public string speed;

public double getPrice() {

return this.price;

}

…

}

行为函数对属性变量的操作同通过隐式的this指针完成，this指针指向当前对象的实例。

**1.2 Go语言**

在go中没有类的概念，而是通过用户自定义类型结构体（struct）来描述对象。struct中并不包含对象的行为函数，而是通过接收器的方法实现。通过在函数名前面的类型变量，使得函数称为某个对象的行为函数。行为函数对对象属性的操作是根据需要决定的，需要对属性进行修改的函数其接收器就是指针型变量，不需要修改属性就可按值进行传递：

type Car struct {

price float32

color string

speed string

}

func (car Car) SpeedUp() {

…

}

func (car \*Car) GetPrice() double {

return car.price

}

go中定义的结构体并不包含对象的属性，而是通过类型变量将其绑定到特定的结构体上去。

1. **继承和多态**

**2.1 继承**

go语言的继承是通过匿名组合实现的。在一个结构体嵌入一个匿名结构，而这个结构体就可以直接继承匿名结构体的属性和行为函数，而在访问组合体中的方法和函数时，接收器的指针仍然指向的是匿名的结构体，也就是说而且匿名结构体仍然能够被访问

* 1. **多态**

在Java等语言中实现多态需要明确实现一个特定的接口，Java中需要使用关键字implements。而在go语言中，一个结构体只需要实现了某个接口包含的所有函数，就可以说这个结构体实现了某个接口。因此，go语言中对于对象类型的判断取决于对象的行为。

一个对象只要包含某个接口的所有行为方法，那么就可以用这个接口变量来存储对象的实例，只要结构体实现了同一个接口就可以调用同一个函数处理不同的对象类型，利用这个原理go中可以用空接口来表示所有的对象类型，这就是go中实现多态的方式。

* 1. interface的实现

go中的interface的数据结构实际上一个结构体，结构体中包含两个属性成员：指向方法列表的指针tab和指向实现接口的对象实例的指针data：

type iface struct {

tab \*itab

data unsafe.Pointer

}

tab中不仅包含接口包含的方法列表，还包含了接口的元信息和实现类型的元信息，这些元信息包含接口或者自定义结构体的名称、类型和方法名。tab的数据结构也是一个结构体：

type itab struct {

inter \*interfacetype //接口的元信息

\_type \*\_type //实现接口的结构体的元信息

link \*itab

bad int32

unused int32

fun [0]unsafe.Pointer //接口的方法列表数组

}

通过接口进行函数调用的操作就是通过函数的类型访问itab的方法列表。go编译器在编译时通过检测接口元数据是否和结构体元数据匹配从而判断结构体是否实现了接口，完成接口和结构体的动态绑定。