**Go语言实践3**

**自定义结构类型**

1. *type TNameA struct {*
2. *var1 T1*
3. *var2 T2*
4. *}*
5. *type TNameB struct {*
6. *var3 T1*
7. *var4 TNameA*
8. *}*
9. 自定义结构类型初始化的方法
10. instance\_a := TNameA{var1: value1, var2: value2}
11. instance\_a:= TNameA{value1, value2} // 与变量声明的顺序有关
12. instance\_b:= TNameB{var3: value3, var4: TNameA{value1, value2}} // 嵌套
13. // ：= 称为短变量声明操作符=声明+初始化
14. 关于强制类型转化
15. go内部不存在隐式类型转化，要进行显式强制类型转化 形式如 b := type(a)
16. 举例
17. type Duration int64
18. var dur Duration
19. dur = Duration(6400)
20. golang中函数的值传递和引用传递
21. 对**数值类型**(int32 int64 uint64 float) **字符串**类型(string) 和**bool**类型，默认使用值传递，也就是函数传参时候是复制的。
22. 对**切片**，**映射**，通道，**接口**和函数类型，默认使用引用传递，也就是函数内改变这些变量，会影响到函数外。
23. **自定义的结构类型，默认也是用值传递，因此如果结构体非常大，推荐传递指针来减小开销**。
24. [https://golangbot.com/interfaces-part-1/](https://wx.qq.com/cgi-bin/mmwebwx-bin/webwxcheckurl?requrl=https%3A%2F%2Fgolangbot.com%2Finterfaces-part-1%2F&skey=%40crypt_d69d7c18_53eb1be6b1d30b16eb79badd6f4c2064&deviceid=e300181382803481&pass_ticket=VDaNodFACxWhio0fWfmXFQAlGxAUqlSa9ILhN%252BEqw0zJrAb47Tai7ma91Db0h2c4&opcode=2&scene=1&username=@26ac78631e438dc9b3582452caa107cb9550ba233f9e32262b9f9c2d90853914)
25. <https://golangbot.com/interfaces-part-2/>

**自定义结构体的函数方法**

1. *type TNameA struct {*
2. *var1 T1,*
3. *var2 T2 }*
4. *func (ins TNameA) print\_var(){*
5. *fmt.Printf(ins.var1, ins.var2)*
6. *}*
7. *func (ins \*TNameA) change\_var(a T1, b T2){*
8. *ins.var1 = a*
9. *ins.var2 = b // 注意这里 指针也可以直接用'.'来调用函数或者成员变量*
10. *}*
11. *a:=TNameA{1, 2}*
12. *a.print\_var()*
13. *a.change\_var(3, 4)*
14. 在func和传入参数之间增加了一个参数，称为**接收者**，作用类似于this指针。
15. 接受者参数指针时：称为**指针接收者**，此时类似于引用传递，可以改变实例的值。
16. 接受者为普通自定义结构体类型：称为**值接收者**，此时用复制一份结构体实例，函数内操作不会改变实例的值。
17. **golang允许值或者指针来通过'.'调用成员函数和变量，非常方便**。

**接口和多态**

C++用虚函数和基类的指针来实现多态，golang中则直接将拥有同名函数的类抽象成接口(interface)来实现多态.

还是以经典的游戏举例，要实现两个兵种在一个回合里攻防的函数game\_loop 使用了如下的接口

1. *package main*
2. *import "fmt"*
3. *type Farmer struct {*
4. *atk int32*
5. *def int32*
6. *life int32*
7. *}*
8. *type Footman struct {*
9. *atk int32*
10. *weapon\_level int32*
11. *def int32*
12. *life int32*
13. *}*
14. *type game\_unit interface { // 接口是一个方法集合 有点像虚基类*
15. *do\_attack(defender game\_unit) bool // 接口的输入可以是接口 引用传递*
16. *loss\_life(damage int32) bool // 有返回值 有普通输入类型*
17. *}*
18. *func (f \*Farmer) do\_attack(defender game\_unit) bool {*
19. *// 指针接收者 要求接口也是指针类型*
20. *return defender.loss\_life(f.atk)*
21. *}*
22. *func (f \*Farmer) loss\_life(damage int32) bool {*
23. *f.life -= damage*
24. *if f.life <= 0 {*
25. *return false*
26. *} else {*
27. *return true*
28. *}*
29. *}*
30. *func (f \*Footman) do\_attack(defender game\_unit) bool {*
31. *return defender.loss\_life(f.atk \* f.weapon\_level)*
32. *}*
33. *func (f \*Footman) loss\_life(damage int32) bool {*
34. *f.life -= (damage - f.def \* f.weapon\_level)*
35. *if f.life <= 0 {*
36. *return false*
37. *} else {*
38. *return true*
39. *}*
40. *}*
41. ***func game\_loop(atker game\_unit, der game\_unit) {***
42. *// 传入参数是两个接口 可以是Farmer/Footman 也可以是对应的指针*
43. *if(atker.do\_attack(der)){ // 指针接受者 因此接口必须要是指针类型*
44. *return*
45. *} else {*
46. *der.do\_attack(atker)*
47. *return*
48. *}*
49. *}*
50. *func main() {*
51. *farmer := Farmer{10, 1, 100}*
52. *footman := Footman{20, 2, 10, 200}*
53. *game\_loop(****&farmer****,* ***&footman****) // 传入指针类型*
54. *}*
55. 接口类似C++中的虚基类，是一个方法集。函数在各自的类中都必须有实现。比如上述例子中的do\_attack和loss\_life。
56. 成员函数接收者类型与传入的接口类型对应关系。
57. 接收者为指针类型时，传入的接口必须是对应类的指针，如上述例子.
58. 接收者为值类型时，传入的接口可以是对应类的指针，也可以是值。

**嵌入类型**

嵌入类型的概念和C++中继承非常相似。

1. 要在结构体中嵌入一个类型，只需要声明类型的名字即可，调用的时候直接调用类型名。嵌入的称为内部类型(类似基类)外部结构称为外部类型(类似子类)
2. *type unit struct {*
3. *life int32*
4. *}*
5. *func (u \*unit) show\_life\_point(){*
6. *fmt.Printf("%d\n", u.life)*
7. *}*
8. *type farmer struct {*
9. *unit*
10. *def int32*
11. *}*
13. 内部类型实现的方法，可以直接被外部类型调用。如果外部类型实现了同名方法则优先调用外部类型的（和继承中的函数调用方式完全一致），这个过程称为“提升“
14. 内部类型如果实现了接口方法集中的方法，那么也会自动提升到外部类型。如果外部类型实现了同名的接口方法，则也优先使用外部类型的。

**公开或未公开的标识符**

一个包里的变量/类型/函数如果要被另一个包使用，就必须是首字母大写的，否则就是包私有的，无法被公开访问(类似public/private的概念)

**课后题**

实现一个基于go的LRU cache

1. <https://github.com/golang/groupcache/tree/master/lru>
2. key 和value需要为任意类型，在代码里是如何通过接口实现的？
3. OnEvicted 这个自带的函数类型起到了什么作用？