Swift第五节课：枚举 & Optional

C语言枚举的写法回顾

在将 `Swift` 的枚举之前，我们先来回顾一下 `C` 语言的枚举写法：

```
enum  枚举名  {
    枚举值1,
    枚举值2,
    ……};
```

比如我们要表示一周 7天，这个时候我们用 `C` 语言的枚举写法应该是这样的：

```
enum weak
{
     MON, TUE, WED, THU, FRI, SAT, SUN
};
```

第一个枚举成员的默认值为整型的 0，后面的枚举值依次类推，如果我们想更改，只需要这样操作

```
enum weak
{
     MON = 1, TUE, WED, THU, FRI, SAT, SUN
};
```

那如何定义一个枚举变量

```
enum weak
{
     MON = 1, TUE, WED, THU, FRI, SAT, SUN
} weak;

enum
{
     MON = 1, TUE, WED, THU, FRI, SAT, SUN
} weak;
```

`Swift` 中枚举写法类比

接下来回到 `Swift` 枚举中，我们思考一个例子：如果说我们要定一个颜色的枚举 `LGColor` ，在 `Swift` 中应该怎么写

```
enum weak
{
     case MONDAY
     case TUEDAY
     case WEDDAY
     case THUDAY
     case FRIDAY
     case SATDAY
     case SUNDAY
}
```

上述代码也可以直接一个 `case` ,然后用逗号隔开

```
enum weak
{
    case MON, TUE, WED, THU, FRI, SAT, SUN
}
```

上述代码中我们的枚举值默认是整形，这个 `C` 是一致的，如果我们想要表达的 `String` 怎么办?

```
enum weak: String
{
    case MON = "MON"
    case TUE = "TUE"
    case WED = "WED"
    case THU = "THU"
    case FRI = "FRI"
    case SAT = "SAT"
    case SUN = "SUN"
}
```

`=` 号左边的值在 `Swift` 中我们把他叫做 `RawValue` ,如果我们不想写后面的字符串，这个时候我们就可以使用 `隐士 RawValue 分配`

```
enum weak: Int {
    mon, tue, wed, thu, fri = 10, sat, sun
}
```

这里不仅对 `Int` 类型适用，对 `String` 类型同样适用，看下面这段代码的打印结果

```
enum weak: String {
    case mon, tue, wed, thu, fri = "Hello", sat, sun
}

print(weak.mon.rawValue)
print(weak.fri.rawValue)
```

枚举的遍历

那我们的枚举能不能像集合那样遍历那？答案是可以的：

```
enum weak: String {
    case mon, tue, wed, thu, fri = "Hello", sat, sun
```

```
}

extension weak: CaseIterable{}

var allCase = weak.allCases

for c in allCase{
    print(c)
}
```

关联值

如果我们想用枚举表达更复杂的信息，而不仅仅是一个 `RawValue` 这么简单，这个时候我们就可以使用 `Associated Value`

```
enum Shape{
    case circle(radious: Double)
    case rectangle(width: Int, height: Int)
}
```

枚举的嵌套

```
enum CombineDirect{
    enum BaseDirect{
        case up
        case down
        case left
        case right
    }

    case leftUp(combineElement1: BaseDirect, combineElement2: BaseDirect)
    case rightUp(combineElement1: BaseDirect, combineElement2: BaseDirect)
```

```
    case leftDown(combineElement1: BaseDirect, combineElement2: BaseDirect)
    case rightDown(combineElement1: BaseDirect, combineElement2: BaseDirect)
}

let combind = CombineDirect.leftDown(combineElement1: .left, combineElement2: .down)
```

```
struct Skill{
    enum KeyType{
        case up
        case down
        case left
        case right
    }

    let key: KeyType

    func launchSkill(){
        switch key {
        case .left,.right:
            print("left, right")
        case .down,.up:
            print("up, down")
        }
    }
}
```

枚举中包含属性

`enum` 中能包含计算属性，类型属性。不能包含存储属性

```
enum Shape{
    case circle(radious: Double)
```

```
    case rectangle(width: Int, height: Int)

    var radious: Double

    static var height = 20.0

    var width: Double{
        get{
            return 10.0
        }
    }
}
```

枚举中包含方法

```
enum KeyType{
    case up
    case down
    case left
    case right

    func isVaild() -> Bool{
        switch self {
        case .up, .down, .left, .right:
            return true
        default:
            return false
        }
    }
}
```