Swift 结构体 Swift 属性

Swift 类

Swift 类是构建代码所用的一种通用且灵活的构造体。

我们可以为类定义属性（常量、变量）和方法。

与其他编程语言所不同的是，Swift 并不要求你为自定义类去创建独立的接口和实现文件。你所要做的是在一个单一文件中定义一个类，系统会自动生成面向其它代码的外部接口。

类和结构体对比

Swift 中类和结构体有很多共同点。共同处在于：

- 定义属性用于存储值
- 定义方法用于提供功能
- 定义附属脚本用于访问值
- 定义构造器用于生成初始化值
- 通过扩展以增加默认实现的功能
- 符合协议以对某类提供标准功能

与结构体相比，类还有如下的附加功能：

- 继承允许一个类继承另一个类的特征
- 类型转换允许在运行时检查和解释一个类实例的类型
- 解构器允许一个类实例释放任何其所被分配的资源
- 引用计数允许对一个类的多次引用

语法:

```
class classname {
   Definition 1
   Definition 2
   ……
   Definition N
}
```

类定义

```
class student{
   var studname: String
   var mark: Int
```

```
    var mark2: Int
}
```

实例化类：

```
let studrecord = student()
```

实例

```
import Cocoa

class MarksStruct {
    var mark: Int
    init(mark: Int) {
        self.mark = mark
    }
}

class studentMarks {
    var mark = 300
}
let marks = studentMarks()
print("成绩为 \(marks.mark)")
```

以上程序执行输出结果为：

```
成绩为 300
```

作为引用类型访问类属性

类的属性可以通过 . 来访问。格式为：**实例化类名.属性名**：

```
import Cocoa

class MarksStruct {
   var mark: Int
   init(mark: Int) {
      self.mark = mark
   }
}

class studentMarks {
   var mark1 = 300
   var mark2 = 400
   var mark3 = 900
```

```
}
let marks = studentMarks()
print("Mark1 is \(marks.mark1)")
print("Mark2 is \(marks.mark2)")
print("Mark3 is \(marks.mark3)")
```

以上程序执行输出结果为：

```
Mark1 is 300
Mark2 is 400
Mark3 is 900
```

恒等运算符

因为类是引用类型，有可能有多个常量和变量在后台同时引用某一个类实例。

为了能够判定两个常量或者变量是否引用同一个类实例，Swift 内建了两个恒等运算符：

恒等运算符	不恒等运算符
运算符为：===	运算符为：!==
如果两个常量或者变量引用同一个类实例则返回 true	如果两个常量或者变量引用不同一个类实例则返回 true

实例

```
import Cocoa

class SampleClass: Equatable {
    let myProperty: String
    init(s: String) {
        myProperty = s
    }
}
func ==(lhs: SampleClass, rhs: SampleClass) -> Bool {
    return lhs.myProperty == rhs.myProperty
}

let spClass1 = SampleClass(s: "Hello")
let spClass2 = SampleClass(s: "Hello")

if spClass1 === spClass2 {// false
    print("引用相同的类实例 \(spClass1)")
}

if spClass1 !== spClass2 {// true
    print("引用不相同的类实例 \(spClass2)")
}
```

以上程序执行输出结果为：

```
引用不相同的类实例 SampleClass
```

Swift 结构体 Swift 属性