

The Swift Programming Language in Chinese

<https://github.com/letsswift>

<http://letsswift.com/>

June 15, 2014

目录

1 Swift 中文教程（十九）类型嵌套	1
1.1 类型嵌套实例	1
1.1.1 类型嵌套的引用	3

1 Swift 中文教程 (十九) 类型嵌套

枚举类型常被用于实现特定类或结构体的功能。也能够在有多种变量类型的环境中，方便地定义通用类或结构体来使用，为了实现这种功能，Swift 允许你定义类型嵌套，可以在枚举类型、类和结构体中定义支持嵌套的类型。

要在一个类型中嵌套另一个类型，将需要嵌套的类型的定义写在被嵌套类型的区域 `{}` 内，而且可以根据需要定义多级嵌套。

1.1 类型嵌套实例

下面这个例子定义了一个结构体 `BlackjackCard`(二十一点)，用来模拟 `BlackjackCard` 中的扑克牌点数。`BlackjackCard` 结构体包含 2 个嵌套定义的枚举类型 `Suit` 和 `Rank`。

在 `BlackjackCard` 规则中，Ace 牌可以表示 1 或者 11，Ace 牌的这一特征用一个嵌套在枚举型 `Rank` 的结构体 `Values` 来表示。

```
struct BlackjackCard {
    // 枚举类型 Suit
    enum Suit: Character {
        case Spades = " ", Hearts = " ", Diamonds = " ", Clubs = " "
    }
    // 枚举类型 Rank
    enum Rank: Int {
        case Two = 2, Three, Four, Five, Six, Seven, Eight, Nine, Ten
        case Jack, Queen, King, Ace
        struct Values {
            let first: Int, second: Int?
        }
        var values: Values {
            switch self {
            case .Ace:
                return Values(first: 1, second: 11)
            case .Jack, .Queen, .King:
                return Values(first: 10, second: nil)
            default:
                return Values(first: self.rawValue, second: nil)
            }
        }
    }
}
```

```

        }
    }
}

// BlackjackCard 的初始化函数
let rank: Rank, suit: Suit
var description: String {
    var output = "suit is \(suit.rawValue),"
    output += " value is \(rank.values.first)"
    if let second = rank.values.second {
        output += " or \(second)"
    }
    return output
}
}

```

枚举型的 `Suit` 用来描述扑克牌的四种花色，并分别用一个 `Character` 类型的值代表花色符号。

枚举型的 `Rank` 用来描述扑克牌从 `Ace~10,J,Q,K,13` 张牌，并分别用一个 `Int` 类型的值表示牌的面值。(这个 `Int` 类型的值不适用于 `Ace,J,Q,K` 的牌)。

如上文所提到的，枚举型 `Rank` 在自己内部定义了一个嵌套结构体 `Values`。这个结构体包含两个变量，只有 `Ace` 有两个数值，其余牌都只有一个数值。结构体 `Values` 中定义的两个属性：

`first`, 为 `Int` `second`, 为 `Int?`, 或 “optional `Int`”

`Rank` 定义了一个计算属性 `values`，这个计算属性会根据牌的面值，用适当的数值去初始化 `Values` 实例，并赋值给 `values`。对于 `J,Q,K,Ace` 会使用特殊数值，对于数字面值的牌使用 `Int` 类型的值。

`BlackjackCard` 结构体自身有两个属性 — `rank` 与 `suit`，也同样定义了一个计算属性 `description`，`description` 属性用 `rank` 和 `suit` 的中内容来构建对这张扑克牌名字和数值的描述，并用可选类型 `second` 来检查是否存在第二个值，若存在，则在原有的描述中增加对第二数值的描述。

因为 `BlackjackCard` 是一个没有自定义构造函数的结构体，在 `Memberwise Initializers for Structure Types` 中知道结构体有默认的成员构造函数，所以你可以用默认的 `initializer` 去初始化新的常量 `theAceOfSpades`:

```

let theAceOfSpades = BlackjackCard(rank: .Ace, suit: .Spades)
println("theAceOfSpades: \(theAceOfSpades.description)")

```

```
// 输出 "theAceOfSpades: suit is , value is 1 or 11"
```

尽管 Rank 和 Suit 嵌套在 BlackjackCard 中，但仍可被引用，所以在初始化实例时能够通过枚举类型中的成员名称单独引用。在上面的例子中 description 属性能正确得输出对 Ace 牌有 1 和 11 两个值。

1.1.1 类型嵌套的引用

在外部对嵌套类型的引用，以被嵌套类型的名字为前缀，加上所要引用的属性名：

```
let heartsSymbol = BlackjackCard.Suit.Hearts.rawValue()
// 输出 " " " "
```

对于上面这个例子，这样可以使 Suit, Rank, 和 Values 的名字尽可能的短，因为它们的名字会自然的由被定义的上下文来限定。

本文部分原文来自于 <http://www.swiftguide.cn/> 翻译小组的译文，共同校对中。

参考文献