嵌套类型

这是一篇社区协同翻译的文章，你可以点击右边区块信息里的『改进』按钮向译者提交改进建议。

通常我们会创建枚举来支持特定的类或结构体的功能。类似地，为了可以方便地定义在更复杂类型上下文中使用的实用工具类和结构，Swift 允许你定义 *嵌套类型* ，从而实现在枚举、类和结构体中的定义嵌套类型。

如果要将一个类型嵌套在另一个类型中，只需在支持类型的大括号内定义嵌套类型即可。类型支持多层嵌套。

嵌套类型的使用

下面的示例定义了一个名为 `BlackjackCard` 的结构体，该结构体模拟了二十一点游戏中使用的扑克牌。`BlackjackCard` 结构体包含两个名为 `Suit` 和 `Rank` 的嵌套枚举类型。

在二十一点游戏中，Ace 可以表示一或十一。这个特性由一个嵌套在 `Rank` 中的 `Values` 结构体表示：

```
struct BlackjackCard {

    // 嵌套的 Suit 枚举
    enum Suit: Character {
        case spades = "♠", hearts = "♡", diamonds = "♢", clubs = "♣"
    }

    // 嵌套的 Rank 枚举
    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 的属性和方法
    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
    }
}
```

`Suit` 枚举描述了四种常见的扑克牌花色，并用原始的 `Character` 值来表示。

`Rank` 枚举描述了十三种可能的扑克牌等级，并用枚举原始 `Int` 值来表示它的点数值。（Jack、Queen、King 和 Ace 并不是用枚举原始的 `Int` 值表示的。）

如上所述，`Rank` 枚举中定义了嵌套结构体 `Values` 。这个结构体封装了大多数扑克牌只有一个值，而 Ace 有两个值的事实。`Values` 结构体中定义了两个属性：

- `first`，类型为 `Int`
- `second`，类型为 `Int?`，或者叫做「可选 `Int` 」

`Rank` 还定义了一个计算属性 `values` ，它返回一个 `Values` 结构体的实例。此计算属性会根据扑克牌的点数，使用适当的值来初始化一个新创建的 `Values` 实例。对于 `jack` 、 `queen` 、 `king` 和 `ace` 扑克牌，使用了特殊值来表示。对于数字扑克牌，它使用自己本身的原始 `Int` 值来表示。

`BlackjackCard` 结构体本身有两个属性 --- `rank` 和 `suit` 。它还定义了一个名为 `description` 的计算属性，它使用存储在 `rank` 和 `suit` 中的值来生成扑克牌的名称和值的描述。`description` 属性会使用可选绑定来检查是否添加第二个值的描述。

因为 `BlackjackCard` 是一个没有自定义构造器的结构体，所以它有一个隐式的成员构造器，参见 结构体类型的成员构造器 中描述。你可以使用此构造器来初始化一个名为 `theAceOfSpades` 的常量：

```
let theAceOfSpades = BlackjackCard(rank: .ace, suit: .spades)
print("theAceOfSpades: \(theAceOfSpades.description)")
// 打印 "theAceOfSpades: suit is ♠, value is 1 or 11"
```

尽管 `Rank` 和 `Suit` 嵌套在 `BlackjackCard` 中，它们的类型也可以从上下文推断出来，因此这个实例的初始化方法能够通过它们的名称（ `.ace` 和 `.spades` ）来推断出对应的具体枚举实例。在上面的例子中， `description` 属性正确地反映了黑体 A 的值是 `1` 或者 `11` 。

引用嵌套类型

要在其定义上下文之外使用嵌套类型，需要在其名称前面加上嵌套在其中的类型的名称：

```
let heartsSymbol = BlackjackCard.Suit.hearts.rawValue
// heartsSymbol 是「♡」
```

对于上面的例子，可以使 `Suit` 、 `Rank` 和 `Values` 的名称尽可能的简短，因为它们的名称会自然地被定义它们的上下文限定。

上一篇

下一篇

点赞

参与译者：3



更多职位

讨论数量: 0

发起讨论

只看当前版本讨论

暂无话题~

兄弟社区

Laravel China

PythonCaff.com

GolangCaff.com

VuejsCaff.com

资源推荐

资源推荐

由 Summer 设计和编码 ❤