嵌套类型

枚举通常被创建来支持特定的类或结构的功能。类似地，定义纯粹用于更复杂类型的上下文中的实用程序类和结构可能很方便。为了达到这个目的，Swift使您能够定义*嵌套类型*，从而将支持枚举，类和结构嵌套在它们支持的类型的定义中。

要将一个类型嵌套在另一个类型中，请将其定义写入其支持的类型的外括号中。类型可以嵌套到所需的级别。

嵌套类型在行动

下面的例子定义了一个叫做“结构”的结构`BlackjackCard`，它模拟了21点游戏中使用的扑克牌。该`BlackjackCard`结构包含两个名为`Suit`and的嵌套枚举类型`Rank`。

在大酒杯中，Ace卡的价值为1或11。此功能由一个名为的结构表示，该结构`Values`嵌套在`Rank`枚举中：

```
struct BlackjackCard {

    // nested Suit enumeration
    enum Suit: Character {
        case spades = "♠", hearts = "♡", diamonds = "♢", clubs = "♣"
    }

    // nested Rank enumeration
    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 properties and methods
    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
    }
}
```

该`Suit`枚举描述了四种常见的扑克牌花色，与原料一起`Character`值来表示他们的象征。

该`Rank`枚举描述了十名三个可能扑克牌行列，与原料一起`Int`值来表示其面值。（这个原始`Int`值不用于Jack，Queen，King和Ace卡。）

如上所述，`Rank`枚举定义了它自己的另一个嵌套结构，称为`Values`。这种结构封装了大多数牌有一个值的事实，但是Ace牌有两个值。该`Values`结构定义了两个属性来表示它：

- first，类型 Int
- second，类型Int?或“可选Int”

Rank还定义了一
于其等级使用适
级的原始Int值。

该BlackjackCard结构本身有两个属性- rank和suit。它还定义了一个被称为的计算属性description，它使用存储在其中的值rank并suit构建卡的名称和值的描述。该description属性使用可选绑定来检查是否有第二个值要显示，如果是，则插入第二个值的附加描述细节。

由于BlackjackCard是没有自定义初始值设定项的结构，因此它具有隐式成员初始值设定项，如构造类型的成员初始值设定项中所述。你可以使用这个初始化器来初始化一个新的常量theAceOfSpades：

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

在本页

即使Rank和Suit被嵌套在BlackjackCard，它们的类型可以从上下文推断，所以这个实例的初始化是能够通过他们的名字的情况下（指枚举案件.ace和.spades单独的）。在上面的示例中，description属性正确地报告黑桃王牌值为1或11。

参考嵌套类型

要在其定义上下文之外使用嵌套类型，请在其名称前面添加嵌套类型的名称：

```
let heartsSymbol = BlackjackCard.Suit.hearts.rawValue
// heartsSymbol is "♡"
```

对于上面的示例，这使得的名字Suit，Rank和Values被保持故意短，因为它们的名称是天然通过在它们所定义的上下文合格。