内嵌类型

枚举通常用于实现特定类或结构体的功能。类似的，它也可以在更加复杂的类型环境中方便的定义通用类和结构体。为实现这种功能，Swift 允许你定义*内嵌类型*，借此在支持类型的定义中嵌套枚举、类、或结构体。

若要在一种类型中嵌套另一种类型，在其支持类型的大括号内定义即可。可以根据需求多级嵌套数个类型。

内嵌类型的使用

下方的例子定义了一个名为 BlackJackCard 的结构体，模拟了21点游戏中的扑克牌。BlackjackCard 结构体包含两个内嵌的枚举类型 Suit 和 Rank 。

在21点游戏中，Ace 可以表示一或十一两个值，这通过 Rank 枚举中内嵌的结构体 Values 决定：

```
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
    }
}
```

Suit 枚举用于描述扑克牌的四种花色，并用原始值 Character 来代表各自的花色。

Rank 枚举用于描述扑克牌可能出现的十三种点数，并用原始值 Int 来代表各自的点数值（这里的 Int 并不会用于 J、Q、K、Ace 的表示）。

如上所述， Rank 枚举中定义了一个内嵌结构体 Values 。这个结构体封装了大多牌只有一个值，而 Ace 可以有两个值这一事实。 Values 结构体定义了两个属性来表示这些：

- Int 类型的 first
- Int? 类型的 second，或者说“可选 Int ”

Rank 还定义了一个计算属性， values ，它用于返回 Values 结构体的实例。这个计算属性会根据牌的点数，用适当的值初始化新的 Values 实例。对于 Jack 、 Queen 、 King 、和 Ace 使用特殊的值。而对于数值的牌，则使用它本身的 Int 原始值。

BlackjackCard 结构体本身有两个属性—— rank 和 suit 。还定义了一个名为 description 的计算属性，用 rank 和 suit 储存的值构建对扑克牌花色和值的描述。 description 属性使用可选绑定来检查是否有第二个值要描述，若有，则添加对第二个值的描述。

由于 BlackjackCard 是一个没有自定义初始化器的结构体，如结构体类型的成员初始化器所述，它有一个隐式的成员初始化器。你可以使用这个初始化器去初始化新的常量 theAceOfSpades ：

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

尽管 Rank 和 Suit 被嵌套在 BlackjackCard 中，但其类型仍可从上下文中推断出来，因此，该实例的初始化器可以单独通过成员名称（ .ace 和 .spades ）引用枚举类型。在上面的例子中， description 属性正确的反馈了黑桃 Ace 拥有 1 或 11 两个值。

引用内嵌类型

要在定义外部使用内嵌类型，只需在其前缀加上内嵌了它的类的类型名即可：

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

对于上面的栗子来说，可以使 Suit 、 Rank 和 Values 的名字尽可能的短，因为它们的名字由定义时的上下文自然限定。