Вложенные типы

- Зачастую перечисления создаются для дополнительной поддержки функциональности определенного класса. Аналогично может быть полезным создание вспомогательных классов или структур предназначенных для контекста более сложного типа. Для достижения этой цели Swift предлагает вам определить *вложенные типы*, в которые вы вкладываете вспомогательные перечисления, классы и структуры, внутри определения типа, которые они поддерживают.

- Чтобы вложить тип в другой тип, вам нужно написать свое определение во внешних фигурных скобках типа, который он поддерживает. Типы могут быть вложены на столько уровней, на сколько это необходимо.

Вложенные типы в действии

- Пример ниже определяет структуру BlackjackCard, которая создает модель игральных карт игрыBlackjack. Структура BlackjackCard содержит два вложенных перечисления типов Suit, Rank.

- В игре Blackjack карта Ace (Туз) имеет значение либо один, либо одиннадцать. Это особенность отображается структурой Values, которая вложена внутрь перечисления Rank:

```
struct BlackjackCard {
   // вложенное перечисление Suit
      enum Suit: Character {
            case Spades = "♠", Hearts = "♡", Diamonds = "♢", Clubs = "♣" }
   // вложенное перечисление 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)
            }
      }
}
```

```
// свойства и методы BlackjackCard
    let rank: Rank, suit: Suit
    var description: String {
     var output = "масть: \(suit.rawValue),"
     output += " значение: \(rank.values.first)"
     if let second = rank.values.second {
         output += " или \(second)" }
     return output
     }
 }
```

Перечисления Suit описывают четыре масти при помощи символа типа Character для их отображения.

- Перечисление Rank вместе со значением Int описывает тринадцать возможных рангов карт, для отображения их номинальной стоимости. (Значение Int не используется для карт Jack, Queen,King, Ace или по-русски Валета, Королевы, Короля и Туза).
- Как уже упоминалось ранее, структура Rank определяет вложенную внутри себя структуру Values. Эта структура инкапсулирует факт того, что большинство карт имеют одно значение, но Туз имеет два значения своей карты. Структура Values определяет два свойства для того, чтобы отобразить это.
- first типа Int
- second типа Int? или “опциональный Int”

Rank так же определяет вычисляемое свойство values, которое возвращает экземпляр структурыValues. Это вычисляемое свойство учитывает ранг карты и инициализирует новый экземплярValues с соответствующими значениями, основываясь на своем ранге. Оно использует специальные значения для карт Jack, Queen, King, Ace. Для карт с цифрами, оно использует значение ранга типа Int.

Сама структура BlackjackCard имеет два свойства rank, suit. Она так же определяет вычисляемое свойство description, которое использует значение, которое храниться в rank иsuit, для того, чтобы создать описание имени и значения карты. Свойство description использует опциональную привязку для проверки наличия второго значения, и если оно есть, то вставляет дополнительное описание для второго значения.

- Из-за того что BlackjackCard является структурой без пользовательских инициализаторов, она имеет неявный почленный инициализатор, что описано в главе “Почленная инициализация структурных типов”. Вы можете использовать этот инициализатор для инициализации новой константы theAceOfSpades:

```
let theAceOfSpades = BlackjackCard(rank: .Ace, suit: .Spades)
print("theAceOfSpades: \(theAceOfSpades.description)")
// выводит "theAceOfSpades: масть: ♠, значение: 1 или 11"
```

- Даже если Rank и Suit являются вложенными в BlackjackCard, их типы могут наследоваться из контекста, таким образом инициализация этого экземпляра может сослаться на члены перечисления по их именам (.Ace и .Spades). В примере выше свойство description корректно отображает то, что Туз (Ace) масти пики (Spades) имеет значение либо 1, либо 11.

Ссылка на вложенные типы

Для того, чтобы использовать вложенные типы снаружи определяющего их контекста, нужно поставить префикс имени типа, внутри которого он вложен, затем его имя:

```
let heartsSymbol = BlackjackCard.Suit.Hearts.rawValue
// heartsSymbol равен "♡"
```

Приведенный выше пример позволяет именам Suit и Rank быть намеренно короткими, потому что их имена изначально подобраны под контекст, в котором они определены.