巨匠線上真人

iOS行動程式基礎開發上架

swift：協定

本堂教學重點

1. 協定的語法
2. 屬性需求
3. 方法需求
4. 可修改方法需求
5. 初始化需求
   - 類別實作初始化需求
6. 協定可當作類型
7. 委派
8. 使用擴充增加協定遵守
   - 判斷採納協定與否
   - 使用擴充定義採納協定
9. 使用協定當元素的集合物件
10. 協定的繼承
11. 只有類別可使用的協定
12. 同時遵守多個協定
13. 檢查是否採納協定
14. 可選擇的協定需求
15. 限定擴充協定條件

1.協定的語法

```
protocol SomeProtocol {
    // protocol definition goes here
}

struct SomeStructure: FirstProtocol, AnotherProtocol {
    // structure definition goes here
}

class SomeClass: SomeSuperclass, FirstProtocol, AnotherProtocol {
    // class definition goes here
}
```

2.屬性需求

```
protocol SomeProtocol {
    var mustBeSettable: Int { get set }
    var doesNotNeedToBeSettable: Int { get }
}
protocol AnotherProtocol {
    static var someTypeProperty: Int { get set }
}
```

2.屬性需求

```
protocol FullyNamed {
    var fullName: String { get }
}


struct Person: FullyNamed {
    var fullName: String
}
let john = Person(fullName: "John Appleseed")
// john.fullName is "John Appleseed"
```

```
class Starship: FullyNamed {
    var prefix: String?
    var name: String
    init(name: String, prefix: String? = nil) {
        self.name = name
        self.prefix = prefix
    }
    var fullName: String {
        return (prefix != nil ? prefix! + " " : "") + name
    }
}
var ncc1701 = Starship(name: "Enterprise", prefix: "USS")
// ncc1701.fullName is "USS Enterprise"
```

3.方法需求

```
protocol SomeProtocol {
    static func someTypeMethod()
}

protocol RandomNumberGenerator {
    func random() -> Double
}
```

```
class LinearCongruentialGenerator: RandomNumberGenerator {
    var lastRandom = 42.0
    let m = 139968.0
    let a = 3877.0
    let c = 29573.0
    func random() -> Double {
        lastRandom = ((lastRandom * a + c).truncatingRemainder(dividingBy:m))
        return lastRandom / m
    }
}
let generator = LinearCongruentialGenerator()
print("Here's a random number: \(generator.random())")
// Prints "Here's a random number: 0.3746499199817101"
print("And another one: \(generator.random())")
// Prints "And another one: 0.729023776863283"
```

4.可修改方法需求

```
protocol Togglable {
    mutating func toggle()
}


enum OnOffSwitch: Togglable {
    case off, on
    mutating func toggle() {
        switch self {
        case .off:
            self = .on
        case .on:
            self = .off
        }
    }
}
var lightSwitch = OnOffSwitch.off
lightSwitch.toggle()
// lightSwitch is now equal to .on

```

5.初始化需求

```
protocol SomeProtocol {
    init(someParameter: Int)
}

class SomeClass: SomeProtocol {
    required init(someParameter: Int) {
        // initializer implementation goes here
    }
}
```

5.初始化需求

```
protocol SomeProtocol {
    init()
}

class SomeSuperClass {
    init() {
        // initializer implementation goes here
    }
}

class SomeSubClass: SomeSuperClass, SomeProtocol {
    // "required" from SomeProtocol conformance; "override" from SomeSuperClass
    required override init() {
        // initializer implementation goes here
    }
}
```

6.協定可當作類型

```
class Dice {
    let sides: Int
    let generator: RandomNumberGenerator
    init(sides: Int, generator: RandomNumberGenerator) {
        self.sides = sides
        self.generator = generator
    }
    func roll() -> Int {
        return Int(generator.random() * Double(sides)) + 1
    }
}

var d6 = Dice(sides: 6, generator: LinearCongruentialGenerator())
for _ in 1...5 {
    print("Random dice roll is \(d6.roll())")
}
// Random dice roll is 3
// Random dice roll is 5
// Random dice roll is 4
// Random dice roll is 5
// Random dice roll is 4
```

7.委派

```
protocol DiceGame {
    var dice: Dice { get }
    func play()
}
protocol DiceGameDelegate: AnyObject {
    func gameDidStart(_ game: DiceGame)
    func game(_ game: DiceGame, didStartNewTurnWithDiceRoll diceRoll: Int)
    func gameDidEnd(_ game: DiceGame)
}
```

7.委派

```
class SnakesAndLadders: DiceGame {
    let finalSquare = 25
    let dice = Dice(sides: 6, generator: LinearCongruentialGenerator())
    var square = 0
    var board: [Int]
    init() {
        board = Array(repeating: 0, count: finalSquare + 1)
        board[03] = +08; board[06] = +11; board[09] = +09; board[10] = +02
        board[14] = -10; board[19] = -11; board[22] = -02; board[24] = -08
    }
    weak var delegate: DiceGameDelegate?
    func play() {
        square = 0
        delegate?.gameDidStart(self)
        gameLoop: while square != finalSquare {
            let diceRoll = dice.roll()
            delegate?.game(self, didStartNewTurnWithDiceRoll: diceRoll)
            switch square + diceRoll {
            case finalSquare:
                break gameLoop
            case let newSquare where newSquare > finalSquare:
                continue gameLoop
            default:
                square += diceRoll
                square += board[square]
            }
        }
        delegate?.gameDidEnd(self)
    }
}
```

7.委派

```
class DiceGameTracker: DiceGameDelegate {
    var numberOfTurns = 0
    func gameDidStart(_ game: DiceGame) {
        numberOfTurns = 0
        if game is SnakesAndLadders {
            print("Started a new game of Snakes and Ladders")
        }
        print("The game is using a \(game.dice.sides)-sided dice")
    }
    func game(_ game: DiceGame, didStartNewTurnWithDiceRoll diceRoll: Int) {
        numberOfTurns += 1
        print("Rolled a \(diceRoll)")
    }
    func gameDidEnd(_ game: DiceGame) {
        print("The game lasted for \(numberOfTurns) turns")
    }
}
```

7.委派

```
let tracker = DiceGameTracker()
let game = SnakesAndLadders()
game.delegate = tracker
game.play()
// Started a new game of Snakes and Ladders
// The game is using a 6-sided dice
// Rolled a 3
// Rolled a 5
// Rolled a 4
// Rolled a 5
// The game lasted for 4 turns
```

8.使用擴充增加協定遵守

```
protocol TextRepresentable {
    var textualDescription: String { get }
}

extension Dice: TextRepresentable {
    var textualDescription: String {
        return "A \(sides)-sided dice"
    }
}

let d12 = Dice(sides: 12, generator: LinearCongruentialGenerator())
print(d12.textualDescription)
// Prints "A 12-sided dice"

extension SnakesAndLadders: TextRepresentable {
    var textualDescription: String {
        return "A game of Snakes and Ladders with \(finalSquare) squares"
    }
}
print(game.textualDescription)
// Prints "A game of Snakes and Ladders with 25 squares"
```

8.使用擴充增加協定遵守

判斷特定情況下才遵守協定

```
extension Array: TextRepresentable where Element: TextRepresentable {
    var textualDescription: String {
        let itemsAsText = self.map { $0.textualDescription }
        return "[" + itemsAsText.joined(separator: ", ") + "]"
    }
}
let myDice = [d6, d12]
print(myDice.textualDescription)
// Prints "[A 6-sided dice, A 12-sided dice]"

```

8.使用擴充增加協定遵守

使用擴充定義採納協定

```
struct Hamster {
    var name: String
    var textualDescription: String {
        return "A hamster named \(name)"
    }
}
extension Hamster: TextRepresentable {}
```

```
let simonTheHamster = Hamster(name: "Simon")
let somethingTextRepresentable: TextRepresentable = simonTheHamster
print(somethingTextRepresentable.textualDescription)
// Prints "A hamster named Simon"
```

9.使用協定當元素的集合物件

使用擴充定義採納協定

```
let things: [TextRepresentable] = [game, d12, simonTheHamster]
```

```
for thing in things {
    print(thing.textualDescription)
}
// A game of Snakes and Ladders with 25 squares
// A 12-sided dice
// A hamster named Simon

```

10.協定的繼承

```
protocol InheritingProtocol: SomeProtocol, AnotherProtocol {
    // protocol definition goes here
}
```

```
protocol PrettyTextRepresentable: TextRepresentable {
    var prettyTextualDescription: String { get }
}
```

```
extension SnakesAndLadders: PrettyTextRepresentable {
    var prettyTextualDescription: String {
        var output = textualDescription + ":\n"
        for index in 1...finalSquare {
            switch board[index] {
            case let ladder where ladder > 0:
                output += "▲ "
            case let snake where snake < 0:
                output += "▼ "
            default:
                output += "○ "
            }
        }
        return output
    }
}
```

```
print(game.prettyTextualDescription)
// A game of Snakes and Ladders with 25 squares:
// ○ ○ ▲ ○ ○ ▲ ○ ○ ▲ ▲ ○ ○ ○ ▼ ○ ○ ○ ○ ▼ ○ ○ ▼ ○ ▼ ○
```

11.只有類別可使用的協定

```
protocol SomeClassOnlyProtocol: AnyObject, SomeInheritedProtocol {
    // class-only protocol definition goes here
}
```

12.同時遵守多個協定

```
protocol Named {
    var name: String { get }
}
protocol Aged {
    var age: Int { get }
}
struct Person: Named, Aged {
    var name: String
    var age: Int
}
func wishHappyBirthday(to celebrator: Named & Aged) {
    print("Happy birthday, \(celebrator.name), you're \(celebrator.age)!")
}
let birthdayPerson = Person(name: "Malcolm", age: 21)
wishHappyBirthday(to: birthdayPerson)
// Prints "Happy birthday, Malcolm, you're 21!"
```

12.同時遵守多個協定

```
class Location {
    var latitude: Double
    var longitude: Double
    init(latitude: Double, longitude: Double) {
        self.latitude = latitude
        self.longitude = longitude
    }
}
class City: Location, Named {
    var name: String
    init(name: String, latitude: Double, longitude: Double) {
        self.name = name
        super.init(latitude: latitude, longitude: longitude)
    }
}
func beginConcert(in location: Location & Named) {
    print("Hello, \(location.name)!")
}

let seattle = City(name: "Seattle", latitude: 47.6, longitude: -122.3)
beginConcert(in: seattle)
// Prints "Hello, Seattle!"
```

13.檢查是否採納協定

```
let objects: [AnyObject] = [
    Circle(radius: 2.0),
    Country(area: 243_610),
    Animal(legs: 4)
]


for object in objects {
    if let objectWithArea = object as? HasArea {
        print("Area is \(objectWithArea.area)")
    } else {
        print("Something that doesn't have an area")
    }
}
// Area is 12.5663708
// Area is 243610.0
// Something that doesn't have an area
```

14.可選擇的協定需求

```
@objc protocol CounterDataSource {
    @objc optional func increment(forCount count: Int) -> Int
    @objc optional var fixedIncrement: Int { get }
}

class Counter {
    var count = 0
    var dataSource: CounterDataSource?
    func increment() {
        if let amount = dataSource?.increment?(forCount: count) {
            count += amount
        } else if let amount = dataSource?.fixedIncrement {
            count += amount
        }
    }
}
```

14.可選擇的協定需求

```
class ThreeSource: NSObject, CounterDataSource {
    let fixedIncrement = 3
}


var counter = Counter()
counter.dataSource = ThreeSource()
for _ in 1...4 {
    counter.increment()
    print(counter.count)
}
// 3
// 6
// 9
// 12
```

15.限定擴充協定條件

```
extension Collection where Element: Equatable {
    func allEqual() -> Bool {
        for element in self {
            if element != self.first {
                return false
            }
        }
        return true
    }
}

let equalNumbers = [100, 100, 100, 100, 100]
let differentNumbers = [100, 100, 200, 100, 200]


print(equalNumbers.allEqual())
// Prints "true"
print(differentNumbers.allEqual())
// Prints "false"
```