SWIFT
Methods

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Method

메소드 : 특정 타입의 클래스, 구조체, 열거형과 관련된 함수

1) 인스턴스 메소드 : 특정 인스턴스에서 실행할 수 있는 메소드
2) 타입 메소드 : 특정 형과 관련된 메소드(클래스 메소드와 유사)

Objective C에서는 클래스 타입에서만 메소드를 선언할 수 있지만 Swift에서는 구조체, 열거형에서도 메소드를 선언할 수 있다.

Instance Method

인스턴스 메소드 : 특정 클래스, 구조체, 열거형의 인스턴스에 속한 메소드를 지칭

```swift
class Counter {
    var count = 0

    // Instance Method
    func increment() {
        count += 1
    }
}

let counter = Counter()

counter.increment()
print(counter.count) // 1
```

The self Property

모드 프로퍼티는 암시적으로 인스턴스 자체를 의미하는 self 라는 프로퍼티를 가지고 있습니다.

인스턴스 메소드 안에서 self 프로퍼티를 이용하여 인스턴스 자체를 참조하는데 사용할 수 있습니다.

```
class Counter {
	var count = 0

	// Instance Method
	func increment() {
		self.count += 1 // self 프로퍼티를 이용해 인스턴스 자체를 참조
	}
}
```

mutating

기본적으로 구조체와 열거형은 값 타입(Value Type)입니다.
따라서 인스턴스 메소드 내에서는 값 타입의 프로퍼티를 변경할 수 없습니다.

아래의 코드에서는 다음과 같이 컴파일 에러가 발생합니다.

```
struct Point {
    var x = 0.0, y = 0.0
    func moveBy(x deltaX: Double, y deltaY: Double) {
        self.x += deltaX        ⊗ Left side of mutating operator isn't mutable: 'self' is immutable
        self.y += deltaY        ⊗ Left side of mutating operator isn't mutable: 'self' is immutable
    }
}

var somePoint = Point(x: 1.0, y: 1.0)   2 ⚠ Variable 'somePoint' was never mutated; consider changing t...
somePoint.moveBy(x: 2.0, y: 3.0)

// "The point is now at (3.0, 4.0)" 출력
print("The point is now at (\(somePoint.x), \(somePoint.y))")
```

mutating

따라서 값 타입 형의 메소드에서 프로퍼티를 변경하고 싶을 경우에는 mutating 이라는 키워드를 사용하여 프로퍼티를 변경할 수 있습니다.

```
struct Point {
    var x = 0.0, y = 0.0
    mutating func moveBy(x deltaX: Double, y deltaY: Double) {
        self.x += deltaX
        self.y += deltaY
    }
}

var somePoint = Point(x: 1.0, y: 1.0)
somePoint.moveBy(x: 2.0, y: 3.0)

print("The point is now at (\(somePoint.x), \(somePoint.y))”)
```

mutating

mutating 메소드 안에서 self 프로퍼티를 사용하여 완전히 새로운 인스턴스를 생성할 수 있습니다.

```
struct Point {
    var x = 0.0, y = 0.0

    mutating func moveBy(x deltaX: Double, y deltaY: Double) {
        // self를 통하여 새로운 자신의 Point 인스턴스를 생성하여 값을 설정할 수 있습니다.
        // self.x += deltaX
        // self.y += deltaY
        self = Point(x: x + deltaX, y: y + deltaY)
    }
}
```

mutating

열거형(enum)에서 mutating 메소드 사용하여 다른 상태로의 전환 등을 표현할 수 있습니다.

```
enum TriStateSwitch {
    case off, low, high

    mutating func next() {
        switch self {
            case .off:
                self = .low
            case .low:
                self = .high
            case .high:
                self = .off
            }
    }
}

var ovenLight = TriStateSwitch.low
ovenLight.next()

// ovenLight 값은 .high
print(ovenLight)

ovenLight.next()

// ovenLight 값은 .off
print(ovenLight)
```

Type Methods

타입 메소드 : 특정 타입 자체에서 호출하여 사용하는 함수를 지칭
func 키워드 앞에 static 또는 class 키워드를 추가하여 사용

1) static 메소드 : 서브클래스에서 오버라이드를 할 수 없음
2) class 메소드 : 서브클래스에서 오버라이드가 가능

기존 Objective C는 클래스에서만 타입 메소드를 선언할 수 있던 것과 달리 Swift는 클래스, 구조체, 열거형에서 모두 타입 메소드를 사용할 수 있습니다.

Type Methods

```
class SomeClass {
    var value:Int = 0 // 타입 메소드 안에서 사용 불가
    static var valueStatic:Int = 0

    // 타입 메소드
    class func someTypeMethod() {
          value = 10 // 컴파일 에러
          self.value = 1 // 컴파일 에러
          // 타입 메소드 안에서의 self는 인스턴스가 아닌 타입 자신을 의미
          self.valueStatic = 1 // static으로 선언하여 사용 가능
    }
}

SomeClass.someTypeMethod()    // 타입 메소드 호출!
```

Type Methods

타입 메소드에서 @discardableResult 키워드를 사용하면 리턴 값이 있는 메소드를 호출 후 리턴값을 사용하지 않을 경우 컴파일 경고가 발생하는데 그 경고가 발생하지 않도록 해줍니다.

```
struct LevelTracker {
     var currentLevel = 1
     @discardableResult
     mutating func advance(to level: Int) -> Bool {
        if currentLevel > 10 {
          return true
        } else {
           return false
        }
     }
}
```

만약 @discardableResult 키워드를 붙이지 않을 경우 컴파일 시 아래와 같은 경고 메시지가 표시됩니다.

```
var tracker = LevelTracker()
tracker.advance(to: 1)
```

⚠ Result of call to 'advance(to:)' is unus

References

[1] [Swift]Methods 정리 : http://minsone.github.io/mac/ios/swift-methods-summary

[2] Methods : https://docs.swift.org/swift-book/LanguageGuide/Methods.html

[3] Swift) Method : https://zeddios.tistory.com/258

[4] [Swift 3] 메소드 (Method) : https://beankhan.tistory.com/162

[5] Swift - 메소드(Method) : http://seorenn.blogspot.com/2014/06/swift-method.html

References

[6] Swift의 static 메서드와 class 메서드 : https://medium.com/@miles3898/swift의-static-메서드와-class-메서드-975d367c4c19

[7] Swift - Methods : https://www.tutorialspoint.com/swift/swift_methods.htm

[8] Swift4 :메소드 : Method : #메소드의 범위 : #mutating : #self #값 타입 수정 : https://the-brain-of-sic2.tistory.com/7

[9] Methods : https://wlaxhrl.tistory.com/44

[10] [Swift 5.2] Methods - Apple Documentation : https://you9010.tistory.com/298

Thank you!