SWIFT
Properties

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Properties?

프로퍼티란 클래스, 구조체, 열거형에서 소속된 변수 및 속성 (Attributes) 등을 불리우는 개념이다.

프로퍼티의 종류는 크게 아래와 같이 분류할 수 있다.

1) Stored Property(저장 프로퍼티)
: 인스턴스의 변수나 상수

2) Computed Property(연산 프로퍼티)
: 직접적인 값을 저장하지 않고 값을 연산한 결과값

3) Type Property(타입 프로퍼티)
: 특정 타입에 사용되는 프로퍼티(클래스 변수)

Stored Properties

- 객체의 값(속성)을 저장하고 있는 기본적인 프로퍼티
- 객체가 생성이 되면 자동적으로 초기화된다.
- 클래스, 구조체에서 지원, 열거형(Enum)에는 지원되지 않음
- var로 선언하면 "변수"를 저장
- let으로 선언하면 "상수"를 저장, 선언 이후 변경 불가

Stored Properties

구조체 사용 예시

```
struct FixedLengthRange {
    var firstValue: Int
    let length: Int
}

let rangeOfThreeItems = FixedLengthRange(firstValue: 0, length: 3)
// 구조체는 Value Type 데이터이므로 let으로 타입 변경 가능
rangeOfThreeItems.firstValue = 6 // error!
rangeOfThreeItems.length = 10 // error!

var rangeOfThreeItems2 = FixedLengthRange(firstValue: 0, length: 3)
rangeOfThreeItems2.firstValue = 6
```

Stored Properties

클래스 사용 예시

```
class FixedLengthRange {
   var firstValue: Int
   let length: Int

   init(firstValue : Int, length:Int) {
        self.firstValue = firstValue
        self.length = length
   }
}

Let rangeOfThreeItems = FixedLengthRange(firstValue: 0, length: 3)

// 클래스는 기본적으로 Reference Type 데이터이므로 let으로 선언하여도 원본에 바로 접근
rangeOfThreeItems.firstValue = 3
rangeOfThreeItems.length = 10 // error!
```

Lazy Stored Properties

- 변수가 사용된 이후에 저장되지 않는 프로퍼티
- 즉, 값이 사용되기 전까지는 계산되지 않는다.
- lazy라는 키워드를 사용하여 선언
- let 상수 타입은 사용 불가
- lazy 프로퍼티가 초기화 되지 않은 상태에서 여러 쓰레드가 동시에 이 lazy프로퍼티에 액세스 한다면, 이 프로퍼티가 단 한번만 초기화 된다는 것을 보장할 수 없음

Lazy Stored Properties

```
class DataImporter {
    /*
        DataImporter는 외부 파일에서 데이터를 가져오는 클래스입니다.
        이 클래스는 초기화 하는데 매우 많은 시간이 소요된다고 가정하겠습니다.
    */

    var filename = "data.txt"
}
```

```
class DataManager {
    lazy var importer = DataImporter()
    var data = [String]()
}

let manager = DataManager()

manager.data.append("Some data")
manager.data.append("Some more data”)

// DataImporter 인스턴스는 이 시점에 생성돼 있지 않습니다.

print(manager.importer.filename)
// the DataImporter 인스턴스가 생성되었습니다.
// "data.txt" 파일을 출력합니다.
```

Computed Properties

- 특정 연산을 통해 필요할 때 연산을 통해 값을 리턴
- 클래스, 구조체, 열거형에서 모두 사용 가능
- var로 선언하여야 한다.
- 반드시 연산 프로퍼티를 위한 저장 프로터티가 하나 있어야 한다.
- 실제 값을 가지고 있는 것이 아리나, getter, seeter등을 통해서 값을 설정하고 전달해준다.
- get, set 동시에 구현 가능하며, get만 구현하는 것도 가능
- set만 구현하는 것은 안된다.
- set에서 파라미터를 생략할 수 있으며, newValue 키워드를 사용
- 기존 언어의 Getter, Setter보다 코드의 분산도 줄이고 직관적임

Computed Properties

```swift
struct Point{
    var x: Int
    var y: Int

    var oppositePoint: Point{
        set(point) {
            x = -point.x
            y = -point.y
       }
       get {
            return Point(x: -x, y: -y)
       }
    }

    var oppositePoint2: Point{
        set { // Swift에서의 set 함수에는 반드시 입력값이 있으므로 미지 지정한 newValue 키워드를 통하여 축약 가능하다.
            x = -newValue.x
            y = -newValue.y
        }
        get {
            return Point(x: -x, y: -y)
        }
    }
}

var point = Point(x: 10, y: 10)
print(point.oppositePoint) // Point(x: -10, y: -10)
print(point) // Point(x: 10, y: 10)

point.oppositePoint = Point(x: 10, y: 10)
print(point) // Point(x: -10, y: -10)
```

Type Properties

- 인스턴스 생성없이 객체내의 프로퍼티에 접근가능
- 프로퍼티를 타입 자체와 연결하는 것을 지칭
- 타입 프로퍼티는 저장 타입 프로퍼티와 연산 타입 프로퍼티가 있음
- 저장 타입 프로퍼티는 상수(let) 및 변수(var) 가능
- 저장 타입 프로퍼티는 반드시 기본값을 설정하여야 한다
- 연산 타입 프로퍼티는 무조건 변수(var)로 선언 가능
- static 키워드를 사용하여 정의
- 클래스 타입에 대한 연산 타입 프로퍼티의 경우, class 키워드를 사용하여 서브클래스가 슈퍼클래스의 구현을 재정의(override)할 수 있음

Type Properties

```
struct SomeStructure {
	static var storedTypeProperty = "Some value."
	static var computedTypeProperty: Int {
		return 1
	}
}

enum SomeEnumeration {
	static var storedTypeProperty = "Some value."
	static var computedTypeProperty: Int {
		return 6
	}
}

class SomeClass {
	static var storedTypeProperty = "Some value."
	static var computedTypeProperty: Int {
		return 27
	}

	class var overrideableComputedTypeProperty: Int {
		return 107
	}
}

class ChildSomeClass : SomeClass{
	// 슈퍼클래스의 특정 타입 프로터티를 재정의 가능
	override static var overrideableComputedTypeProperty: Int{
		return 2222
	}
}
```

Type Properties

별도의 인스턴스 생성없이 바로 '.'을 통해서 프로퍼티 접근 가능

```
SomeStructure.storedTypeProperty = "Another value."

print(SomeStructure.storedTypeProperty) // Prints "Some value."
print(SomeStructure.storedTypeProperty) // Prints "Another value."
print(SomeEnumeration.computedTypeProperty) // Prints "6"
print(SomeClass.computedTypeProperty) // Prints "27"
```

Property Observers

- 프로퍼티에 새값이 설정될 때 해당 이벤트를 감지할 수 있는 옵저버를 제공해줌
- 프로퍼티 옵저벼는 새 값이 이전 값과 같더라도 항상 호출
- 지연 저장 프로퍼티에서는 사용할 수 없음
- 연산 프로퍼티 setter에서 값의 변화를 감지할 수 있음

제공되는 옵저버

willSet : 값이 저장되기 바로 직전에 호출됨
didSet : 새 값이 저장되고 난 직후에 호출됨

Property Observers

```
class StepCounter {
        var totalSteps: Int = 0 {
             willSet(value) {
                  print("About to set totalSteps to \(newTotalSteps)")
             }
             didSet {
                 if totalSteps > oldValue  {
                    print("Added \(totalSteps - oldValue) steps")
                }
            }
        }
}

let stepCounter = StepCounter()

stepCounter.totalSteps = 200
// About to set totalSteps to 200
// Added 200 steps

stepCounter.totalSteps = 360
// About to set totalSteps to 360
// Added 160 steps

stepCounter.totalSteps = 896
// About to set totalSteps to 896
// Added 536 steps
```

References

[1] Swift) Properties - Stored Property(저장 프로퍼티): https://zeddios.tistory.com/243

[2] Swift) Properties - Computed Property(연산 프로퍼티) : https://zeddios.tistory.com/245

[3] Swift - 프로퍼티 : https://penguin-story.tistory.com/37

[4] Swift의 프로퍼티에 대한 이해 : https://soooprmx.com/archives/6707

[5] 프로퍼티 : https://yagom.github.io/swift_basic/contents/13_property/

References

[6] [Swift] 프로퍼티(Property) : https://jinshine.github.io/2018/05/22/Swift/6.프로퍼티(Property)/

[7] [스위프트 대충보기] 9. 프로퍼티(property) : https://github.com/enshahar/swiftsummary/blob/master/%5B스위프트%20대충보기%5D%209.%20프로퍼티(property).md

[8] 프로퍼티 get, set, didSet, willSet in iOS : https://medium.com/ios-development-with-swift/프로퍼티-get-set-didset-willset-in-ios-a8f2d4da5514

[9] [Swift] Computed Properties (연산 프로퍼티) : https://hyesunzzang.tistory.com/133

[10] Swift - 프로퍼티(Properties) : http://seorenn.blogspot.com/2014/06/swift-properties.html

References

[11] Swift4 : 프로퍼티 : #Properties : #get, set " #willSet,didSet : https://the-brain-of-sic2.tistory.com/6

[12] 프로퍼티 (Properties) : https://jusung.gitbook.io/the-swift-language-guide/language-guide/10-properties

Thank you!