SWIFT
Access Control

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Access Control

접근제어(Access Control)는 특정 코드의 접근을 다른 소스파일이나 모듈에서 제한하는 것입니다.

이렇게 접근제어를 함으로써 특정 코드의 세부적인 구현을 감추고 딱 필요한 만큼 공개해 다른 곳에서 사용할 수 있도록 합니다.

접근제어는 클래스, 구조체, 열거형 등 개별 타입에도 적용할 수 있고 그 타입에 속한 프로퍼티, 메소드, 초기자, 서브스크립트에도 적용할 수 있습니다.

Access Control

Swift의 접근제어는 모듈과 소스파일에 기반을 두고 있습니다.

모듈은 코드를 배포하는 단일 단위로 하나의 프레임워크나 앱이 이 단위로 배포되고 다른 모듈에서 Swift의 import키워드를 사용해 import될 수 있습니다.
Xcode의 각 빌드 타겟은 Swift에서 분리된 단일 모듈로 취급됩니다.

소스파일은 모듈안에 있는 소스파일을 의미합니다.
각 소스파일에 여러 특정 타입을 선언해 사용할 수 있습니다.

Access Levels

Swift에서는 아래의 5개의 접근레벨을 제공합니다.

Keyword	Access Level	Range
open	개방 접근 수준	모듈 외부에서도 접근 가능
public	공개 접근 수준	모듈 외부에서도 접근 가능
internal	내부 접근 수준	하나의 모듈 내부에서만 접근 가능
fileprivate	파일 내부 접근 수준	하나의 파일 내에서만 접근 가능
Private	비공개 접근 수준	정의한 블록 내부에서만 접근 가능

Access Levels

open & public : Open과 Public 접근자 모두 선언한 모듈이 아닌 다른 모듈에서 사용가능합니다. 두 접근자의 차이점은 open은 다른 모듈에서 오버라이드와 서브클래싱이 가능하지만 public 접근자로 선언된 것은 다른 모듈에서는 오버라이드와 서브클래싱이 불가능 합니다.

internal : 기본 접근레벨로 아무 접근레벨을 선언하지 않으면 internal로 간주됩니다. internal레벨로 선언되면 해당 모듈 전체에서 사용 가능합니다.

fileprivate : 특정 엔티티를 선언한 파일 안에서만 사용 가능합니다.

private : 특정 엔티티가 선언된 괄호({}) 안에서만 사용 가능합니다.

Access Levels

접근레벨 가이드 원칙 (Guiding Principle of Access Levels)

- public 변수는 다른 internal, file-private 혹은 private 타입에서 정의될 수 없습니다. 왜냐하면 그 타입은 public 변수가 사용되는 모든 곳에서 사용될 수 없을 것이기 때문입니다.

- 함수는 그 함수의 파라미터 타입이나 리턴 값 타입보다 더 높은 접근 레벨을 갖을 수 없습니다. 왜냐하면 함수에는 접근 가능하지만 파라미터에 접근이 불가능 하거나 혹은 반환 값 타입보다 접근 레벨이 낮아 함수를 사용하는 관련 코드에서 이용할 수 없을 수 있기 때문입니다.

- 아무런 접근 레벨을 명시하지 않은 경우 internal을 갖게 됩니다.

- 단일 타겟 앱에서는 특별히 접근레벨을 명시할 필요가 없지만 필요에 따라 fileprivate, private등을 사용해 앱내에서 구현 세부사항을 숨길 수 있습니다.

- 기본적으로 open이나 public으로 지정된 엔티티만 다른 모듈에서 접근 가능합니다. 하지만 유닛테스트를 하는 경우 모듈을 import할때 import앞에 @testable이라는 에트리뷰트를 붙여주면 해달 모듈을 테스트가 가능한 모듈로 컴파일해 사용합니다.

Access Control Syntax

각 접근자를 사용해 클래스와 변수, 상수를 선언한 예는 다음과 같습니다.

```
public class SomePublicClass {}
internal class SomeInternalClass {}
fileprivate class SomeFilePrivateClass {}
private class SomePrivateClass {}

public var somePublicVariable = 0
internal let someInternalConstant = 0
fileprivate func someFilePrivateFunction() {}
private func somePrivateFunction() {}

// internal 접근레벨은 생략할 수 있습니다.
class SomeInternalClass {} // implicitly internal
```

Custom Types

클래스 그리고 내부 변수 및 함수에 대해서 다양하게 접근 제어를 설정할 수 있습니다.

```
public class SomePublicClass {
    public var somePublicProperty = 0
    var someInternalProperty = 0                    // implicitly internal class member
    fileprivate func someFilePrivateMethod() {}
    private func somePrivateMethod() {}
}

class SomeInternalClass {                           // implicitly internal class
    var someInternalProperty = 0                    // implicitly internal class member
    fileprivate func someFilePrivateMethod() {}
    private func somePrivateMethod() {}
}

fileprivate class SomeFilePrivateClass {
    func someFilePrivateMethod() {}                 // implicitly file-private class member
    private func somePrivateMethod() {}
}

private class SomePrivateClass {
    func somePrivateMethod() {}                     // implicitly private class member
}
```

Custom Types

튜플타입의 접근레벨은 튜플에서 사용되는 모든 타입의 접근레벨 중 가장 제한적인 접근레벨을 갖습니다.

예를 들어, 만약 하나는 internal 다른 하나는 private접근 권한을 갖는 2개의 타입으로 구성된 튜플은 더 낮은 레벨인 private 접근 레벨을 갖습니다.

또한 튜플은 자체적으로 접근레벨을 선언하지 않고 사용하는 클래스, 구조체, 열거형 그리고 함수 등에 따라 자동으로 최소 접근레벨을 부여 받습니다. 즉, 튜플은 명시적으로 접근권한을 명시하지 않습니다.

Custom Types

함수 타입의 접근레벨은 함수의 파라미터 타입과 리턴타입의 접근레벨 중 최소의 접근레벨로 계산돼 사용됩니다.

그래서 그것에 맞는 접근 레벨을 함수 앞에 명시해 줘야 합니다.

```
internal class SomeInternalClass {}
private class SomePrivateClass {}

// 명시적인 접근 레벨을 명시하지 않은 함수는 컴파일 시 에러가 발생
func someFunction() -> (SomeInternalClass, SomePrivateClass) {
    return (SomeInternalClass(), SomePrivateClass())
}

// private로 접근 레벨을 명시하여 컴파일 에러 제거
private func someFunction() -> (SomeInternalClass, SomePrivateClass) {
    return (SomeInternalClass(), SomePrivateClass())
}
```

Custom Types

열거형에서 각 case는 enum의 접근레벨을 따르고 개별적으로 다른 접근레벨을 지정할 수 없습니다.

```
public enum CompassPoint {
    case north
    case south
    case east
    case west
}
```

Custom Types

서브클래스는 수퍼클래스보다 더 높은 접근레벨을 갖을 수 없습니다.

예를들어, 수퍼클래스가 internal 를 갖는데 그것을 서브클래싱해서 pubic 서브클래스를 만들 수 없습니다.

하지만 메소드는 서브클래스에서 더 높은 접은 레벨을 갖는 메소드로 오버라이드 할 수 있습니다.

```
public class A {
    fileprivate func someMethod() {}
}

internal class B : A {
    override internal func someMethod() {}
}
```

Custom Types

상수, 변수, 프로퍼티 등은 그 타입보다 더 높은 접근레벨을 갖을 수 없습니다.

즉, private타입에서 public프로퍼티를 선언할 수 없습니다.

```
private class SomePrivateClass {}

// 컴파일 에러 미발생
private var privateInstance = SomePrivateClass()

// 컴파일 에러 발생
public var privateInstance = SomePrivateClass()
```

Custom Types

상수, 변수, 프로퍼티 그리고 서브스크립트의 게터와 세터는 자동으로 해당 상수, 변수, 프로퍼티 그리고 서브스크립트가 갖는 접근레벨을 동일하게 갖습니다.

필요에 따라 세터의 접근레벨을 게터보다 낮게 정할 수 있습니다.

이를 위해 다음 키워드를 var혹은 subscript앞에 붙여 사용합니다.

키워드 예시) fileprivate(set), private(set), internal(set)

```
public struct TrackedString {
    public private(set) var numberOfEdits = 0
    public var value: String = "" {
        didSet {
            numberOfEdits += 1
        }
    }
    public init() {}
}
```

Custom Types

초기자(Initializers)의 접근레벨은 타입의 레벨과 같거나 낮습니다.

한가지 예외상황은 지정초기자(required Initializer)인데, 이 지정초기자는 반드시 이 초기자가 속한 타입과 접근레벨이 같아야 합니다.

기본 초기자는 타입의 접근레벨이 public으로 지정돼있지 않은 이상 타입과 같은 접근레벨을 갖습니다.

만약 타입의 접근 레벨이 public으로 지정돼 있으면 기본 초기자는 internal 접근레벨을 갖습니다.

Custom Types

프로토콜의 접근레벨과 그 안의 요구사항의 접근레벨은 항상 동일합니다.

이미 존재하는 프로토콜을 상속받아 새로운 프로토콜을 선언하는 경우 새 프로토콜은 상속을 한 프로토콜과 같은 접근레벨을 갖습니다.

특정 타입은 그 타입보다 낮은 접근레벨을 갖는 프로토콜을 따를 수 있습니다.

예를들어 public 타입을 다른 모듈에서 사용하지만 internal 프로토콜은 오직 internal프로토콜이 선언된 곳에서만 사용가능 합니다.

만약 타입이 public이고 프로토콜이 internal이라면 그 프로토콜을 따르는 타입도 internal입니다.

Custom Types

클래스, 구조체, 열거형 등에 익스텐션에서 새로운 맴버를 추가하면 새로 추가된 것은 기존에 타입이 선언된 접근레벨과 같은 레벨을 갖습니다.

익스텐션에 명시적으로 접근레벨을 지정할수도 있습니다.

익스텐션을 프로토콜로 사용하는 경우 명시적인 접근레벨 변경이 불가능합니다.
대신 프로토콜의 접근레벨이 익스텐션의 프로토콜 구현에서 사용 됩니다.

```
struct SomeStruct {
    private var privateVariable = 12
}

// 원본 선언에서 private 멤버로 선언한 것을 익스텐션에서 멤버로 접근할 수 있습니다.
// 하나의 익스텐션에서 private멤버로 선언한 것을 같은 파일의 다른 익스텐션에서 접근할 수 있습니다.
// 하나의 익스텐션에서 private멤버로 선언한 것을 원본 선언에서 멤버로 접근할 수 있습니다.
extension SomeStruct : SomeProtocol {
    func doSomething() {
        // 프로토콜을 따르는 익스텐션에서 그 타입의 private변수에 접근할 수 있습니다.
        print(privateVariable)
    }
}
```

Custom Types

지네릭 타입 혹은 함수는 지네릭 타입 혹은 함수 자체 그리고 타입 파라미터의 접근레벨의 최소 접근레벨을 갖습니다.

타입 별칭(Type Aliases)은 별칭을 붙인 타입보다 같거나 낮은 접근레벨을 갖습니다.

예를들어, private 타입의 별칭은 private, file-private, internal, public, open 타입의 접근 권한을 갖을 수 있습니다.

References

[1] Swift) Λccess Control(접근 제어) - (1) : https://zeddios.tistory.com/383

[2] [Swift]Λccess Control 정리 : http://minsone.github.io/mac/ios/swift-access-control-summary

[3] Swift 접근 제어자(Λccess Control) : https://hcn1519.github.io/articles/2018-01/Swift_ΛccessControl

[4] [Swift] 접근제어 Λccess Control : https://baked-corn.tistory.com/80

[5] 접근 제어(Λccess Control) : https://kka7.tistory.com/131

References

[6] Swift Access Control : https://dejavuqa.tistory.com/158

[7] 접근제어 (Access Control) : https://jusung.gitbook.io/the-swift-language-guide/language-guide/25-access-control

[8] Swift - 액세스 컨트롤(Access Control) : http://seorenn.blogspot.com/2014/07/swift-access-control.html

[9] Access Control : https://wlaxhrl.tistory.com/32

[10] Swift - Access Control : https://www.tutorialspoint.com/swift/swift_access_control.htm

Thank you!