SWIFT
Protocol (2/2)

Bill Kim(김정훈) | ibillkim@gmail.com

목차

Delegation

위임(Delegation)은 클래스 혹은 구조체 인스턴스에 특정 행위에 대한 책임을 넘길 수 있게 해주는 디자인 패턴 중 하나입니다.

```
class Dice {

}

protocol DiceGame {
    var dice: Dice { get }
    func play()
}
```

```
// DiceGameDelegate에 선언해서 실제 DiceGame의 행위와 관련된 구현을
// DiceGameDelegate를 따르는 인스턴스에 위임합니다.
protocol DiceGameDelegate: AnyObject {
    func gameDidStart(_ game: DiceGame)
    func game(_ game: DiceGame, didStartNewTurnWithDiceRoll diceRoll: Int)
    func gameDidEnd(_ game: DiceGame)
}

class SnakesAndLadders: DiceGame {
    var dice: Dice = Dice()
    weak var delegate: DiceGameDelegate?

    func play() {
        delegate?.gameDidStart(self)
    }

    func end() {
        delegate?.gameDidEnd(self)
    }
}
```

Delegation

```
class DiceGameTracker: DiceGameDelegate {
    var numberOfTurns = 0

    func gameDidStart(_ game: DiceGame) {

     }

     func game(_ game: DiceGame, didStartNewTurnWithDiceRoll
diceRoll: Int) {

     }

     func gameDidEnd(_ game: DiceGame) {

     }
}

let tracker = DiceGameTracker()
let game = SnakesAndLadders()

game.delegate = tracker
game.play()
```

Adding Protocols Conformance with an Extension

이미 존재하는 타입에 새 프로토콜을 따르게 하기 위해 익스텐션을 사용할 수 있습니다.

원래 값에 접근 권한이 없어도 익스텐션을 사용해 기능을 확장할 수 있습니다.

```
class Dice {
    let sides: Int
    init(sides: Int) {
        self.sides = sides
    }
}

protocol TextRepresentable {
    var textualDescription: String { get }
}

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

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

Adding Protocols Conformance with an Extension

만약 어떤 프로토콜을 충족에 필요한 모든 조건을 만족하지만 아직 그 프로토콜을 따른다는 선언을 하지 않았다면 그 선언을 **빈 익스텐션으로 선언**할 수 있습니다.

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

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

// Hamster 인스턴스인 simonTheHamster는 이제 TextRepresentable 타입으로 사용할 수 있습니다.
extension Hamster: TextRepresentable {}

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

Collections of Protocol Types

프로토콜을 Array, Dictionary등 Collection 타입에 넣기위한 타입으로 사용할 수 있습니다.

```
let d12 = Dice(sides: 12)
let simonTheHamster = Hamster(name: "Simon")

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

for thing in things {
    print(thing.textualDescription)
    // A 12-sided dice
    // A hamster named Simon
}
```

Protocol Inheritance

클래스 상속같이 **프로토콜도 상속**할 수 있습니다. 여러 프로토콜을 상속받는 경우 콤마(,)로 구분합니다.

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

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

Class-Only Protocols

구조체, 열거형에서 사용하지 않고 클래스 타입에만 사용가능한 프로토콜을 선언하기 위해서는 프로토콜에 AnyObject를 추가합니다.

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

Protocol Composition

동시에 여러 프로토콜을 따르는 타입을 선언할 수 있습니다.

```
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) // Happy birthday, Malcolm, you're 21!
```

Optional Protocol Requirements

프로토콜을 선언하면서 필수 구현이 아닌 선택적 구현 조건을 정의할 수 있습니다.

이 프로토콜의 정의를 위해서 @objc 키워드를 프로토콜 앞에 붙이고, 개별 함수 혹은 프로퍼티에는 @objc와 optional 키워드를 붙입니다.

@objc 프로토콜은 클래스 타입에서만 채용될 수 있고 구조체나 열거형에서는 사용할 수 없습니다.

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

Protocol Extensions

익스텐션을 이용해 **프로토콜을 확장**할 수 있습니다.

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

extension RandomNumberGenerator {
    func randomBool() -> Bool {
        return random() > 0.5
    }
}

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 here's a random Boolean: \(generator.randomBool())")
// Prints "And here's a random Boolean: true"
```

Protocol Extensions

프로토콜 익스텐션이 특정 조건에서만 적용되도록 선언할 수 있습니다. 이 선언에는 where 절을 사용합니다.

다음은 Collection 엘리먼트가 Equatable인 경우에만 적용되는 allEqual()메소드를 구현한 예입니다.

```swift
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()) // true
print(differentNumbers.allEqual()) // false
```

References

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

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

[3] Swift) Protocols (4) : https://zeddios.tistory.com/347

[4] Swift Protocol 적재적소에 사용하기 : https://academy.realm.io/kr/posts/understanding-swift-protocol/

[5] Swift 4.2 Protocol 공식 문서 정리 : https://medium.com/@kimtaesoo188/swift-4-2-protocol-공식-문서-정리-f3a97c6f8cc2

References

[6] 프로토콜 (Protocols) : https://jusung.gitbook.io/the-swift-language-guide/language-guide/21-protocols

[7] 오늘의 Swift 상식 (Protocol) : https://medium.com/@jgj455/오늘의-swift-상식-protocol-f18c82571dad

[8] [Swift] Protocol [01] : https://baked-corn.tistory.com/24

[9] [Swift] Protocol [02] : https://baked-corn.tistory.com/26

[10] Swift - 프로토콜 지향 프로그래밍 : https://blog.yagom.net/531/

Thank you!