공식문서 읽기 : Type Casting

https://docs.swift.org/swift-book/LanguageGuide/TypeCasting.html

Type Casting

타입 캐스팅은 인스턴스의 타입을 확인하거나 어떤 클래스의 인스턴스를 해당 클래스 계층 구조의 슈퍼 클래스나 서브 클래스로 취급하는 방법이다. Swift의 타입 캐스팅안 is, as 연산자로 구분 가능하다. 간단한 표현으로 타입을 확인하거나 다른 타입으로 캐스트할 수 있다.

타입 캐스팅을 사용해 프로토콜을 준수하는지도 확인할 수 있다.

Difining a Class Hierarchy for Type Casting

클래스와 서브클래스의 계층 구조와 함께 타입캐스팅을 사용해 특정 클래스의 인스턴스 타입을 확인하고 해당 인스턴스를 동일한 계층 내의 다른 클래스로 캐스팅할 수 있다.

```
class MediaItem {
    var name: String
    init(name: String) {
        self.name = name
    }
}

class Movie: MediaItem {
    var director: String
    init(name: String, director: String) {
        self.director = director
        super.init(name: name)
    }
}

class Song: MediaItem {
    var artist: String
    init(name: String, artist: String) {
        self.artist = artist
        super.init(name: name)
    }
}
```

위 코드에는 세개의 클래스가 정의되어 있다. MediaItem(슈퍼클래스)와 이를 상속받는 Movie, Song 클래스가 정의되어 있는에 이를 활용해 타입캐스팅을 정의해본다.

```
let library = [ Movie(name: "Casablanca", director: "Michael Curtiz"),
                Song(name: "Blue Suede Shoes", artist: "Elvis Presley"),
                Movie(name: "Citizen Kane", director: "Orson Welles"),
                Song(name: "The One And Only", artist: "Chesney Hawkes"),
                Song(name: "Never Gonna Give You Up", artist: "Rick Astley") ]

// the type of "library" is inferred to be [MediaItem]
```

library에는 Movie 클래스와 Song 클래스의 인스턴스들이 있다. 이렇게되면 library의 타입은 Array<MediaItem>이 된다. 만약 Array의 요소들을 Movie, Song 타입으로 사용하고 싶다면 타입을 확인하거나 해당 타입으로의 다운캐스팅 과정이 필요하다.

Checking Type

- is 연산자를 사용하면 인스턴스가 특정 서브 클래스 타입인지 확인한다. 만약 인스턴스가 서브클래스의 타입이라면 true를, 아니라면 false를 리턴한다.

```
var movieCount = 0
var songCount = 0

for item in library {
    if item is Movie {
        movieCount += 1
    } else if item is Song {
        songCount += 1
    }
}
print("Media library contains \(movieCount) movies and \(songCount) songs")

// Prints "Media library contains 2 movies and 3 songs"
```

for in loop에서 library라는 배열의 모든 항목에 접근하여, 각각 Movie, Song 클래스로 다운캐스팅이 가능할 경우 true를 반환하기 때문에 블럭 내의 count += 1을 수행한다.

Downcasting

특정 클래스 타입의 상수와 변수는 서브 클래스의 인스턴스를 참조할 수 있다. 만약 이런 경우가 만족한다고 생각하는 경우 as? as! 를 사용해 서브 클래스 타입으로 다운캐스트할 수 있다.

이런 경우 다운캐스팅에 실패할 경우가 있기 때문에 옵셔널 값을 사용할 수 있다.

- as? 다운캐스트 하려는 타입의 옵셔널 값을 반환
- as! 다운캐스트를 시도하고 반환된 옵셔널 결과를 강제로 unwrapping

다운캐스트의 성공이 불확실하다면 as?를 사용한다. 이 경우 실패하면 nil을 반환하는데, 이를 통해 다운캐스트의 성공 여부를 알 수 있다.

만약 다운캐스트의 성공이 확실하다면 as!를 사용할 수 있다. 하지만 이 경우 다운캐스트에 실패하면 런타임 오류가 발생한다.

```
for i in library {
    if let movie = i as? Movie {
        print("Movie: \(movie.name), dir. \(movie.director)")
    } else if let song = i as? Song {
        print("Song: \(song.name), by \(song.artist)")
    }
}

// Movie: Casablanca, dir. Michael Curtiz
// Song: Blue Suede Shoes, by Elvis Presley
// Movie: Citizen Kane, dir. Orson Welles
// Song: The One And Only, by Chesney Hawkes
// Song: Never Gonna Give You Up, by Rick Astley
```

각각의 항목에 대해 MediaType 클래스인지 Movie 클래스인지 Song 클래스인지 확신할 수 없기 때문에 as? 를 사용하는 것이 좋다.

Type Casting for Any and AnyObject

Swift는 타입이 정해지지 않은 두 개의 타입을 제공한다.

- Any: 함수 타입을 포함한 모든 타입의 인스턴스를 나타낼 수 있다.
- AnyObject: 클래스 타입의 인스턴스만 나타낼 수 있다.

이러한 편리한 기능을 제공하지만 Any, AnyObject를 사용하는 것보다는 실제 작업에 사용될 타입을 구체적으로 지정하는 것이 가장 좋다.

만약 배열의 타입이 Any로 만들어졌을 경우, 이 배열에는 어떠한 타입의 요소던 추가될 수 있다. 이런 경우 모든 항목에 대해 특정 타입을 찾아 추가하고 싶다면 switch문의 case에서 is, as 연산자를 사용한다.

```
var things = [Any]()
things.append(0)
things.append(0.0)
things.append(42)
things.append(3.14159)
things.append("hello")
things.append((3.0, 5.0))
things.append(Movie(name: "Ghostbusters", director: "Ivan Reitman"))
things.append({ (name: String) -> String in "Hello, \(name)" })
```

```
for thing in things {
    switch thing {
    case 0 as Int:
        print("zero as an Int")
    case 0 as Double:
        print("zero as a Double")
    case let someInt as Int:
        print("an integer value of \(someInt)")
    case let someDouble as Double where someDouble > 0:
        print("a positive double value of \(someDouble)")
    case is Double:
        print("some other double value that I don't want to print")
    case let someString as String:
        print("a string value of \"\(someString)\"")
    case let (x, y) as (Double, Double):
        print("an (x, y) point at \(x), \(y)")
    case let movie as Movie:
        print("a movie called \(movie.name), dir. \(movie.director)")
    case let stringConverter as (String) -> String:
        print(stringConverter("Michael")) default: print("something else")

    }
}
// zero as an Int
// zero as a Double
// an integer value of 42
// a positive double value of 3.14159
// a string value of "hello"
// an (x, y) point at 3.0, 5.0
// a movie called Ghostbusters, dir. Ivan Reitman
// Hello, Michael
```

위의 코드에서는 things의 모든 항목에 접근할 때마다 switch문으로 타입캐스트를 진행하여 성공한 케이스의 코드를 실행한다.

```
let optionalNumber: Int? = 3
things.append(optionalNumber) // Warning
things.append(optionalNumber as Any) // OK
```

Any 타입은 옵셔널 타입을 포함한 모든 타입의 값을 나타낸다. Swift는 Any 타입의 값으로 특정 타입의 옵셔널 값을 사용하려는 경우 경고를 발생한다. 위의 코드처럼 Int? 형을 Any 타입으로 사용하면 오류를 발생한다는 것이다. 이렇게 옵셔널 값을 Any 값으로 사용해야하는 경우 as 연산자를 사용해 특정 타입의 옵셔널 값을 Any로 캐스팅해서 사용할 수 있다.