Part 12

클래스(Class)

상속(Inheritance)과 초기화(Initialization)

스위프트 공식문서 - **Inheritance / Initialization / Deinitialization**

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

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

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

초기화(Initialization)

초기화와 생성자(Initializer)

구조체/클래스/(열거형)

```
class  Dog  {
      var name: String
      var weight: Double
      ...
      init(name: String, weight: Double) {
            self.name = name
            self.weight = weight
      }
      ...
}
```

초기화 메서드(생성자)

- 함수의 구현이 특별한 키워드인 **init으로 명명**됨
- **인스턴스를 생성 과정**: 저장 속성에 대한 초기값을 설정하여 사용가능한 상태가 되는 것
- 생성자 메서드 실행의 목적은, 모든 **저장 속성 초기화**를 통한 인스턴스 생성
  즉, 생성자 실행의 종료시점에는 모든 저장 속성에 값이 저장되어 있어야 함

- 설계도(클래스, 구조체, 열거형)을 실제로 사용하기 위해 인스턴스를 찍어내는 과정
- 생성자 실행시, 메모리 내에 실제 인스턴스를 생성하는 복잡한 코드가 (자동)구현된다고 생각하면 됨

오버로딩 지원

- 다양한 파라미터 조합으로 **생성자를 여러개 구현 가능**
  (여러가지 방식으로 인스턴스를 생성하는 방법을 제공하는 것)

생성자 직접 구현하지 않으면 ⭐ **(사용자 정의 (직접) 구현이 일단 원칙)** (개발자의 의도가 우선)

(1) 모든 저장 속성에 기본값(또는 옵셔널 타입) 전제

- **클래스**: 기본생성자 **init( )** 제공 (초기화 방법 필요)
- **구조체**: 기본생성자 **init( )** 제공 + 멤버와이즈 이니셜라이저도 **기본 제공**
  (초기화 방법 필요) (편의를 위한 기본 제공/새로운 값으로 설정 가능)

(2) (일부) 저장 속성에 기본값(또는 옵셔널 타입) 전제

- **클래스**: 원칙적으로 일부 값만 가지고, 생성자 구현하지 않는 방법이 존재하지 않음
- **구조체**: 멤버와이즈 이니셜라이저 **기본 제공**
  (편의를 위한 기본 제공 ➡ 일부 / 전체 저장 속성 새로운 값으로 설정 가능)

```
Dog(name: "초코", weight: 15.0)
```

(**멤버와이즈 이니셜라이저**: 직접 구현하지 않아도, 자동으로 제공해주는 저장 속성 설정가능한 생성자)

구조체/클래스의 생성자/소멸자 비교

이니셜라이저(Initializers)

구분	구조체(Struct)	클래스(Class)
1) 지정 생성자 (Designated)	**init( ) { }** (모든 저장속성에 기본값(or옵셔널) 설정으로 생성자 구현 안할시, 기본 생성자 기본제공) **init(파라미터) { }** (생성자 구현 안할시, Memberwise생성자 기본제공)	**init( ) { }** (모든 저장속성에 기본값(or옵셔널) 설정으로 생성자 구현 안할시, 기본 생성자 기본제공) **init(파라미터) { }**
2) 편의 생성자 (Convenience)	X	**convenience init(파라미터) { }** (상속과 관련)
3) 필수 생성자 (Required)	X	**required init(파라미터) { }** (상속과 관련)
4) 실패가능 생성자 (Failable)	**init?(파라미터) { }** **init!(파라미터) { }**	**init?(파라미터) { }** **init!(파라미터) { }**
5) 소멸자 (Deinitializers)	X	**deinit { }**

Part 14
확장(Extensions)

스위프트 공식문서 - **Extensions**

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

타입의 확장

확장 문법

클래스 / 구조체 / (열거형)

```
extension SomeType {
    // 새로운 기능을 추가
}
```

확장 가능 멤버의 종류 (메서드 형태만 가능)

(1) (타입) 계산 속성, (인스턴스) 계산 속성
(속성감시자 추가 안되는 이유: 저장 속성을 관찰하기 때문)
(2) (타입) 메서드, (인스턴스) 메서드
(3) **새로운 생성자** (⭐다만, 클래스의 경우 **편의생성자만 추가 가능**)
(본체의 지정생성자를 호출하는 방법만 가능)
[예외]: 값타입(구조체, 열거형)의 경우, 지정 생성자 형태로도 (자유롭게) 생성자 구현 가능
(4) 서브스크립트
(5) 새로운 중첩 타입 정의 및 사용
(6) 프로토콜 채택 및 프로토콜 관련 메서드

- 클래스 / 구조체 / 열거형 타입에 확장이 가능
- 새로운 메서드(기능)를 추가할 수 있지만 (상속처럼) **본체에 대한 재정의는 불가**

- 애플이 미리 만들어놓은 원본 소스 코드에는 권한이 없지만, **확장을 사용해서 기능을 확장하는 것은 가능 (소급 리모델링)**

*Part12 - 2) 초기화 과정과 생성자 참고

타입의 확장

생성자의 경우 주의

클래스 / 구조체 / (열거형)

```
extension Sometype  {
    ...
    convenience init( ... ) {
        self.init( ... )
    }
    ...
}
```

(확장에서) 새로운 생성자 구현 가능

- 새로운 생성자를 구현해도됨 (새로운 인스턴스 초기화 방법을 제공)

- **본체의 지정생성자 호출**
  (1) 클래스 - 편의생성자만 구현 가능 (본래(본체)의 지정 생성자를 호출하는 방법으로만 구현 가능)
  (2) 구조체 등 - [예외] 지정생성자의 형태로도 (자유롭게) 생성자 구현 가능 (상속 관련 없기 때문)

  [구조체 참고] 모든 저장속성에 기본값 + 본체에서 생성자 구현하지 않은 경우,
  본체에서 기본 생성자 / 멤버와이즈 생성자가 제공 됨
  (직접 생성자 구현시 ➡ 제공 안되는 것이 원칙이지만, 확장에서의 구현은 괜찮음
  따라서 기본 생성자/멤버와이즈 생성자 지속 제공되며, 확장에서 호출도 가능)

(본체는 붕어빵틀과 같은 역할이므로, 원칙적으로는 붕어빵틀이 저장속성을 찍어낼 수 있다는 관점에서 생각하면 됨. 다만, 구조체의 경우 상속과 관련된 부분에서, 자유롭기 때문에 생성자도 구현 가능)