값 타입 / 참조 타입

yagom

Class

- 전통적인 OOP 관점에서의 클래스
- 단일상속
- (인스턴스/타입) 메서드
- (인스턴스/타입) 프로퍼티
- **참조 타입**
- Apple 프레임워크의 대부분의 큰 뼈대는 모두 클래스로 구성

Struct

- C 언어 등의 구조체보다 다양한 기능
- 상속 불가
- (인스턴스/타입) 메서드
- (인스턴스/타입) 프로퍼티
- **값 타입**
- Swift의 대부분의 큰 뼈대는 모두 구조체로 구성

Enum

- 다른 언어의 열거형과는 많이 다른 존재
- 상속 불가
- (인스턴스/타입) 메서드
- (인스턴스/타입) 연산 프로퍼티
- **값 타입**

Enum

- Enumeration
- 유사한 종류의 여러 값을 유의미한 이름으로 한 곳에 모아 정의

  예) 요일, 상태값, 월(Month) 등
- **열거형 자체가 하나의 데이터 타입**
  열거형의 case 하나하나 전부 하나의 유의미한 값으로 취급
- 선언 키워드 - enum

Class / Struct / Enum

	Class	Struct	Enum
Type	Reference	Value	Value
Subclassing	○	X	X
Extension	○	○	○

구조체는 언제 사용하나?

- 연관된 몇몇의 값들을 모아서
  하나의 데이터타입으로 표현하고 싶을 때
- 다른 객체 또는 함수 등으로 전달될 때
  참조가 아닌 복사를 원할 때
- 자신을 상속할 필요가 없거나,
  자신이 다른 타입을 **상속받을 필요가 없을 때**
- Apple 프레임워크에서 프로그래밍을 할 때에는 주로 클래스를 많이 사용

Value vs Reference

- **Value**
  - 데이터를 전달할 때 값을 복사하여 전달
- **Reference**
  - 데이터를 전달할 때 값의 메모리 위치를 전달

```swift
struct SomeStruct {
    var someProperty: String = "Property"
}

var someStructInstance: SomeStruct = SomeStruct()

func someFunction(structInstance: SomeStruct) {
    var localVar: SomeStruct = structInstance
    localVar.someProperty = "ABC"
}

someFunction(someStructInstance)
print(someStructInstance.someProperty)
```

```swift
class SomeClass {
    var someProperty: String = "Property"
}

var someClassInstance: SomeClass = SomeClass()

func someFunction(classInstance: SomeClass) {
    var localVar: SomeClass = classInstance
    localVar.someProperty = "ABC"
}

someFunction(someClassInstance)
print(someClassInstance.someProperty)
```

Data types in Swift

```
public struct Int

public struct Double

public struct String

public struct Dictionary<Key : Hashable, Value>

public struct Array<Element>

public struct Set<Element : Hashable>
```

Swift LOVEs Struct

- 스위프트는 구조체, 열거형 사용을 선호
- Apple 프레임워크는 대부분 클래스 사용
- Apple 프레임워크 사용시 구조체/클래스 선택은 우리의 몫