# 구조체와 클래스

## 구조체

### 구조체는 struct키워드로 정의

구조체를 정의한다는 것은 새로운 타입을 생성해주는 것이다. 그러므로 기본타입 이름처럼 대문자 카멜케이스로 이름을 지워야 한다.

Struct 구조체 이름 {
    프로퍼티와 메서드들
}

```
2
3  struct BasicInformation {
4      var name : String //저장 프로퍼티
5      var age : Int
6  }
7
```

구조체 정의를 마친 후 인스턴스를 생성하고 초기화할때 기본 생성된 멤버와이즈 이니셜라이저를 사용, 사용자 정의 이니셜라이저도 구현 가능

멤버와이즈 이니셜라이저 : 프로퍼티의 이름으로 매개변수를 갖는 이니셜라이저

```
7
8  var personInfo : BasicInformation = BasicInformation(name: "kim", age: 99)
9  personInfo.age = 11
10 personInfo.name = "yam"
11
12 let otherInfo : BasicInformation = BasicInformation(name: "bo", age: 22)
13 otherInfo.age = 2                  Cannot assign to property: 'otherInfo' is a 'let
14 otherInfo.name = "ki"               Cannot assign to property: 'otherInfo' is a 'let
15
```

Ex 매개변수 name, age

13번줄, 14번 줄은 struct는 값 타입이므로 상수를 선언하는 let을 썼을때 값을 변경하는게 불가능하여 오류가 뜬다.

## 클래스

클래스를 정의할 때는 class키워드로 정의

클래스를 정의한다는 것은 새로운 타입을 생성해주는 것이다. 그러므로 기본타입 이름처럼 대문자 카멜케이스로 이름을 지워야 한다.

클래스를 정의 후, 인스턴스를 생성하고 초기화할 때는 기본적인 이니셜라이저를 사용

```
class Person {
    var height : Float = 0.0
    var weight : Float = 0.0
}

var kim : Person = Person()
kim.height = 111.1
kim.weight = 123.1

let yam : Person = Person()
yam.weight = 123.3
yam.height = 333.3

```

클래스는 참조타입이므로 let으로 선언을 하여도 내부 프로퍼티 값을 변경할 수 있다.

## 클래스 인스턴스의 소멸

클래스는 참조 타입이므로 참조가 필요가 없을때 메모리에서 해체된다.
이 과정을 거치기 전에 deinit이라는 디이니셜라이저 메소드가 호출된다.(자동 호출)

### 디이니셜라이저

- 클래스 당 하나만 구현
- 매개변수와 반환 값을 가질 수 없음
- 매개변수를 위한 소괄호를 적어주지 않음

```swift

class Person {
    var height : Float = 0.0
    var weight : Float = 0.0

    deinit {
        print("Person 클래스의 인스턴스가 소멸")
    }
}

var kim : Person? = Person()
kim = nil // 인스턴스 소멸

```

### 용도
소멸 전에 데이터를 저장하거나 다른 객체에 인스턴스 소멸을 알려할때 사용

### 구조체와 클래스의 차이

### 같은점
- 값을 저장하기 위해 프로퍼티를 정의 가능
- 기능 실행을 위해 메서드 정의 가능
- 초기화될 때의 상태를 지정하기 위해 이니셜라이저 정의 가능
- 초기구현과 더불어 새로운 기능 추가를 위한 extension으로 확장 가능
- 특정 기능을 위해 특정 프로토콜을 준수할 수 있음

### 다른점
- 디이니셜라이저는 클래스의 인스턴스에만 활용가능
- 구조체는 상속할 수 없다.
- 전달되는 값이 다르다.

**If 함수의 전달인자로 값을 넘길 때**
**참조타입 -> 참조(주소)가 전달**
**값 타입 -> 전달될 값이 복사되어 전달**

```swift
struct BasicInformation {
    let name : String
    var age : Int
}

var kimInfo : BasicInformation = BasicInformation(name: "kim", age: 23)
print(kimInfo.age) //23

var yamInfo = kimInfo // kim값을 복사하여 할당
print(yamInfo.age) //23

yamInfo.age = 24

print(kimInfo.age) // 23
print(yamInfo.age) // 24 값을 복사하여 별개의 값을 갖는다.

```

```swift
class BasicInformation {
    let height : Float = 0.0
    var weight : Float = 0.0
}

var kimInfo : BasicInformation = BasicInformation()
var yamInfo : BasicInformation = kimInfo //kim의 주솟값 할당

print(kimInfo.weight) // 0.0
print(yamInfo.weight) // 0.0

yamInfo.weight = 98.0

print(kimInfo.weight) // 98.0
print(yamInfo.weight) // 98.0 참조하는 곳이 동일하여 값이 같음

```

### 식별 연산자

클래스의 인스턴스끼리 참조가 같은지 확인 할 때 사용

```swift
class BasicInformation {
    let height : Float = 0.0
    var weight : Float = 0.0
}

var kimInfo : BasicInformation = BasicInformation()
var yamInfo : BasicInformation = kimInfo //kim의 주솟값 할당
var otherInfo : BasicInformation = BasicInformation()

print(kimInfo === yamInfo) // true
print(kimInfo === otherInfo) //false
print(yamInfo === otherInfo) //false

```

### 구조체와 클래스 선택사용

애플 가이드라인에서 조건 중에 하나 이상 해당한다면 구조체 사용을 권장

- 연관된 간단한 값의 집합을 캡슐화하는 것만이 목적일 때
- 캡슐화한 값을 참조하는 것보다 복사하는 것이 합당할 때
- 구조체에 저장된 프로퍼티가 값 타입이며 참조하는 것보다 복사하는 것이 합당할 때
- 다른 타입으로부터 상속받거나 자신을 상속할 필요가 없을 때