🔥🚊 Day 18

swift struct 에서 mutating 에 대해 한번 알아보았습니다.

```swift
//
// 맴버 변수 name을 가지는 struct
// struct 는 참조인 class 와 다르기 때문에
// struct 구조의 맴버 변수 값을 변경하려면
// mutating 키워드가 필요
struct Friend {
  var name : String
  // mutaing 키워드로 맴버 변수의
  // 값을 변경하는 메소드
  mutating func changeName(){
    self.name = "hello! " + self.name
  }

}

var myFriend = Friend(name: "jeff")

print(myFriend.name)

myFriend.changeName()

// 맴버 변수가 변경된걸 확인 할 수 있다.
print(myFriend.name)
```

참조 (주소)개념

Class (클래스) → 상속 가능

- 참조 타입입니다.
- ARC로 메모리를 관리합니다.
- 같은 클래스 인스턴스를 여러 개의 변수에 할당한 뒤 값을 변경시키면 할당한 모든 변수에 영향을 줍니다. (메모리만 복사)
- 상속이 가능합니다.
- 타입 캐스팅을 통해 런타임에서 클래스 인스턴스의 타입을 확인할 수 있습니다.
- deinit을 사용하여 클래스 인스턴스의 메모리 할당을 해제할 수 있습니다.

값

Struct (구조체) → 상속 불가능

- 값 타입입니다.
- 구조체 변수를 새로운 변수에 할당할 때마다 새로운 구조체가 할당됩니다.
- 즉 같은 구조체를 여러 개의 변수에 할당한 뒤 값을 변경시키더라도 다른 변수에 영향을 주지 않습니다. (값 자체를 복사)

Class, Struct의 공통점

- 값을 저장할 프로퍼티를 선언할 수 있습니다.
- 함수적 기능을 하는 메서드를 선언 할 수 있습니다.
- 내부 값에. 을 사용하여 접근할 수 있습니다.
- 생성자를 사용해 초기 상태를 설정할 수 있습니다.
- extension을 사용하여 기능을 확장할 수 있습니다.
- Protocol을 채택하여 기능을 설정할 수 있습니다.

* Swift 의 생성자는 init( ) 형태로 작성된다.

```swift
class Friend {
    var name : String

    func changeName(newName: String){
        self.name = newName
    }
    init(_ name: String){
        self.name = name
    }
}

var myFriend = Friend("쩡대리")

myFriend.changeName(newName: "개발하는 쩡대리")

myFriend.name
```

```
Friend
Friend
Friend
"개발하는 쩡대리"
```

name : 쩡대리
name : 개발하는 쩡대리
값이 바뀜.

```swift
myFriend.name

struct BestFriend {

    var name : String

    func changeName(newName: String){
        self.name = newName
//        print(
    }

//    init(_ name
//        self.name = name
```

Cannot assign to property: 'self' is immutable

Mark method 'mutating' to make 'self' mutable — Fix

struct의 경우 임의로 값의 변경이 불가능함을 알 수 있다.

값이 바뀌지 않는다.