ios 입문 강의

7강 상속과 옵셔널

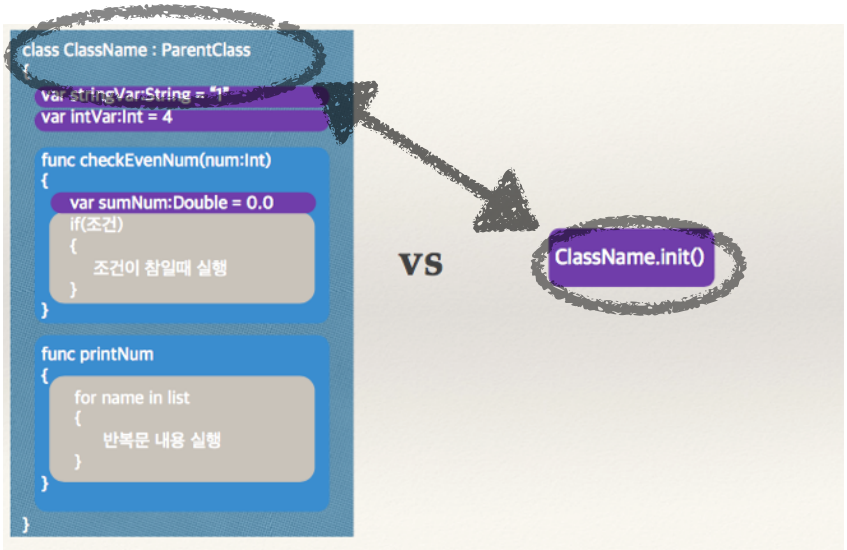
1. 지난 수업 복습

(1) 객체 지향 프로그래밍

* 클래스
  * 특정 기능, 값 등을 하나의 구조로 만들어놓은 객체의 틀
  * 일반적으로 파일 하나당 클래스 하나로 씀

* 객체
  * 클래스를 메모리에 적재시켜 실질적으로 사용하는 상태
  * 도면의 기능을 하는 파일 자체가 클래스 / 그것을 메모리 위에 적재 시킨 것이 객체

* 클래스 vs 객체



var person:Person = Person()

person은 Person의 주소값을 가지고 있는 구조

*** 객체 만들기**

* var calssObj:ClassName = ClassName.init()

* .init -> 초기화

1. 변수 생성: 초기화한 객체를 담을 그릇을 만든다. *여기서 변수 타입은 ClassName이다.
2. 인스터스&초기화(initialize): **클래스 내용을 메모리 할당**하고 **초기화**한다.
3. 초기화한 객체를 변수에 넣고 사용한다.

※ 객체 안에는 속성(변수)과 행위(메소드) -> .을 통해서 사용

classObj.변수

classObj.메소드()

※ 객체란, 실제로 작동이 되고 잇는 상태 <-> 클래스 와는 다르다

** 기본적으로는 객체는 변수가 사라질 때 사라진다. (모두 그런 것은 아님)

*** 초기화**

var classObj:Person = Person( )

클래스 이름 ( )
= 객체로 초기화 한 것

classObj.name = “joo”

classObj.age = 23

* 객체를 만들 때 필요한 데이터가 있다면?

* NSObject를 상속받아서 만든 class이다.
* NSObject는 object-c, swift 통틀어 가장 최상위

```
class Person: NSObject {

    var name:String?
    var age:Int?

    init(name:String, age:Int) {

        self.name = name
        self.age = age

    }
}
```

```
var classObj:Person = Person(name:”Joo”, age:3)
```

Person 클래스를 객체로 만들어 사용할 때

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

실습 1

예제 : 초기화 함수를 포함 클래스 제작

1. Person Class 만들기
- 변수 : 이름, 성별, 나이, 키, 몸무게, 직업, 연봉
- 초기화 메소드 : 이름, 성별을 받는 초기화 함수 작성
- 메소드 : 월급 계산기(연봉 /12),비만율 계산 함수 등

* BMI 지수 = 몸무게(kg) ÷ (신장(m) × 신장(m))
BMI 지수가 18.5 이하 저체중,
18.5~23 정상, 23~25는 과체중,
25~30은 비만, 30~35는 고도비만,
35 이상은 초고도비만으로 분류

2. ViewController에서 친구객체 리스트 만들기- Person Class사용

3. 추가 메소드 만들기
- 친구 이름 입력시 해당 친구의 월급 출력
- 친구 이름 입력시 해당 친구의 비만도 출력

———————— 예제 답안 ————————

최상위 클래스인 NSObject를 상속받은 Person클래스 생성

Person Class : new file 생성하여 만들기

```
class Person: NSObject {

    /* name과 gender은 옵셔널없이 아래의 custom초기화 함수로 초기화 시킨
        다. */
    var name:String
    var gender:String

    // 나머지 변수들은 옵셔널로 초기값 정해주었다.
    var height:Double?
    var age:Int?
    var weight:Double?
    var job:String?
    var salary:Int?

    /* name과 gender의 custom 초기화 함수 */
    init(name:String, gender:String) {
        self.name = name
        self.gender = gender
    }

    /* height, age, weight 값을 세팅해주는 메소드 */
    func setInforWithHeight(height:Double, age:Int, weight:
        Double) {
        self.height = height
        self.age = age
        self.weight = weight
    }

    func printValue() {
        print("name은 \(self.name)입니다.")
        print("gender은 \(self.gender)입니다.")
        print("height은 \(self.height)입니다.")
        print("age은 \(self.age)입니다.")
    }
```

Person 클래스

변수 생성 : 이름, 성별, 나이, 키, 몸무게, 직업, 연봉

1

2

많으셨습니다 :) IOS 입문반 화이팅 ❤️

3

```
func printBMI() {
    var bmi:Double = weight! / ((height!*0.01) * (height!
        *0.01))
    if (bmi <= 18.5) {
        print ("저체중")
    }
    else if (18.5 < bmi && bmi <= 23) {
        print("정상")
    }
    else if (23 < bmi && bmi <= 25) {
        print("과체중")
    }
    else if (25 < bmi && bmi <= 30) {
        print("비만")
    }
    else if (30 < bmi && bmi <= 35) {
        print("고도비만")
    }
    else
    {
        print("초고도비만")
    }
}
```

1 까지 만들고 <viewDidLoad 에서 사용할 때 - >

```
var person1:Person = Person(name: "주영민", gender: "남")
person1.printValue()

var person2:Person = Person(name: "설현", gender: "여")
person2.printValue()
```

1까지 만들고 ViewDidLoad에서 다음과 같이 실행하면,
초기화 때 세팅이 되는 name과 gender외에는 값이 나오지 않습니다.

```
------------*****-------------
name은 주영민입니다.
name은 남입니다.
name은 nil입니다.
name은 nil입니다.
name은 설현입니다.
name은 여입니다.
name은 nil입니다.
name은 nil입니다.
------------*****-------------
```

2 **Person클래스에 printValue 함수를 추가하고** <viewDidLoad 에서 사용함 >

```
var person1:Person = Person(name: "주영민", gender: "남")
person1.setInforWithHeight(160, age: 11, weight:50)
person1.printValue()
```

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

setInforWithHeight 메소드로서 나머지 값을 세팅하면 nil로 나왔던 값들이 세팅이 됩니다.

```
----------*****----------
name은 주영민입니다.
name은 남입니다.
name은 Optional(160.0)입니다.
name은 Optional(11)입니다.
```

3 Person클래스에 printBMI 함수를 추가하고 <viewDidLoad 에서 사용함 >

```
var person1:Person = Person(name: "주영민", gender: "남")
person1.setInforWithHeight(160, age: 11, weight:50)
person1.printValue()
person1.printBMI()
```

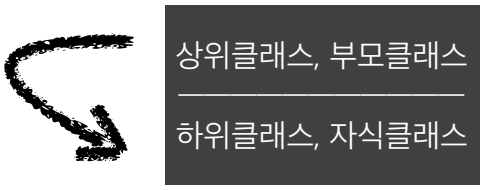
```
name은 주영민입니다.
gender은 남입니다.
height은 Optional(160.0)입니다.
age은 Optional(11)입니다.
정상
```

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

2. 상속

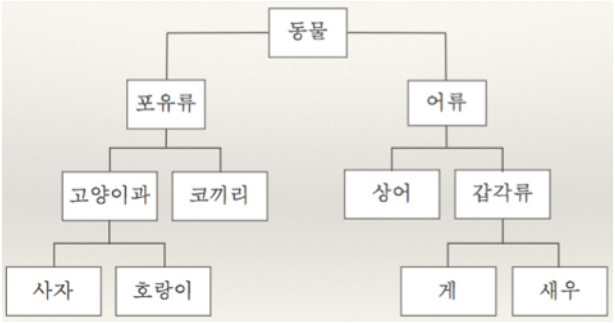
(1) 상속이란?

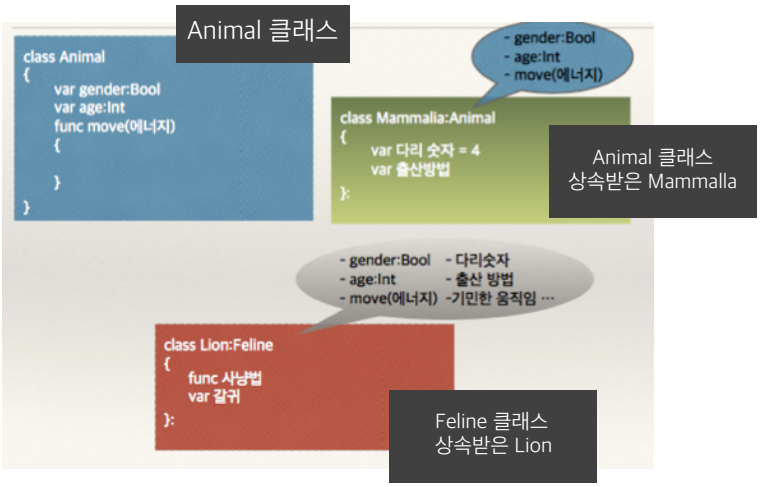
상속은 **새로운 클래스가 기존의 클래스의 자료와 연산을 이용할 수 있게 하는 기능**이다. 상속을 받는 새로운 클래스를 부클래스, 파생 클래스, 하위 클래스, 자식 클래스라고 하며 새로운 클래스가 상속하는 기존의 클래스를 기반 클래스, 상위 클래스, 부모 클래스라고 한다. 상속을 통해서 기존의 클래스를 상속받은 하위 클래스를 이용해 프로그램의 요구에 맞추어 클래스를 수정할 수 있고 클래스 간의 종속관계를 형성함으로써 객체를 조직화 할 수 있다.



* 상속 -> 부모가 자식에게 무언가 주는 것
* 상속은 새로운 클래스가 기존 클래스의 자료(속성)와 연산(메소드)을 이용할 수 있게 하는 기능

(2) 객체지향에서의 상속





* 부모가 할 수 있는 건 자식도 다 할 수 있다.
* 자식이 내려갈 수록 할 수 있는 것이 많아진다.
* 동물의 특징은 아래의 모든 동물들에게 영향을 준다
  * <—> 그러나 사자의 특징이 모든 동물에 영향을 주는 건 아니다.
    그저, 사자는 동물의 특징을 상속받은 것 중 하나일 뿐.

* IOS에 적용시켜 예를 들면, UIView를 상속받은 UILabel이 있다.
  UIView는 보여지는 화면에 대한 객체이고, UILabel은 글자를 올려줘서 보여주는 객체이다.

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

UILabel은 화면에 무언가를 보여주는 UIView의 속성을 가지지면서 글자를 올려줘 보여준다는 특성이 있는 것.

* Swift에서의 상속 특징
  * 다중상속 불가. 오직 하나만 상속 가능. + 꼭 하나는 상속을 받아야 한다.
  * 그래서 모든 클래스는 최상위 NsObject를 상속 받는다.

(3) 오버라이드(override) - 함수 재정의

```
class Animal
{
    var gender:Bool
    var age:Int
    func move(에너지)
    {
      //움직인다.
    }
}
```

```
class Lion:Animal
{
    override  func move(에너지)
    {
      //조용히 움직인다.
    }
}
```

* 예를 들어, 동물에서의 move는 그냥 움직임이라면, 사자의 move 는 '조용히' 움직임이라는 특징이 있다.
* 그러나, 비슷한 속성의 메소드를 매 번 두번 따로 만들기엔 해깔림

* 때문에 override라는 키워드를 붙여서 함수를 재정의한다.
  * 기존 부모의 move 함수를 재정의하여 사용한다.
    * ※ 참고 - viewDidLoad에 override가 붙어있다. UIViewController에서 override 한 것.

* **super** - 상위 클래스의 변수, 메소드를 지칭
* **self** - 자기 자신 클래스의 변수, 메소드를 지칭

  * 기존 부모의 move 함수의 속성을 함께 쓰고 싶을 때에는 super를 사용

```
class Animal
{
    var gender:Bool
    var age:Int
    func move(에너지)
    {
        print ("움직인다")
    }
}
```

```
class Lion:Animal
{
    override  func move(에너지)
    {
        super.move ( )
        print ("조용조용")
    }
}
```

이렇게 하면 Animal의 move 함수를 재정의한 Loin의 move 함수는

super.move()를 통해 Animal의 move 함수 내용 —> print (" 움직인다 ")

Loin의 move 함수 안에 추가하여 쓰여진 내용 ——-> print ("조용조용")

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

두 가지 다 사용할 수 있게 되는 것이다.

실습 2

예제 : 상속클래스만들기

1. 부모 클래스 만들기 : Person Class
(지난 실습 과제 하신 분은 사용가능 - 추가 func work())

2. 자식클래스만들기:Developer
{ override func work()}

* 나만의 커스텀 클래스 만들기
Person Class를 상속받아서 나만의 클래스를 만들어 보세요.

3. 자식객체생성

———————— 예제 답안 ————————

부모클래스 (Person)

```
/* 부모클래스의 init */
init(name:String, gender:String) {
    self.name = name
    self.gender = gender
}

func work()
{
    print("놀지말고 일해")
}
```

ViewController에서의 사용할 때

부모클래스 객체

자식클래스 객체

```
let friend2:Person = Person(name: "야곰", gender:"M")
friend2.setAge(32, height: 176, weight: 80)
friend2.salary = 3500
friends!.append(friend2)

//상속 실습
let friend3:Developer = Developer(name: "강한아", gender: "G")
friend3.setAge(45, height: 160, weight: 65)
friend3.salary = 1200
friends!.append(friend3)

//override 실습
friend1.work()
friend2.work()
friend3.work()
```

부모클래스 객체 -> 결과) "놀지말고 일해"

자식클래스 객체 -> 결과) "코딩"

```
class Developer: Person {

    override init(name: String, gender: String) {
        super.init(name: name, gender: gender)
        self.job = Job(name: "Developer")
    }

    //상속 - override실습 - Developer 자식 클래스
    override func work() {
        print("코딩")

    }

    func isDevelop()
    {
        print("i'm develop")
    }

}
```

자식 클래스 (Developer)

자식 클래스는

부모에서 정의되었던 메소드들,
설정들을 쓸 수도 있고,

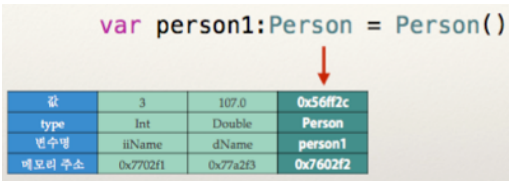
재정의 된 것은 재정의된 내용으로 쓰인다.

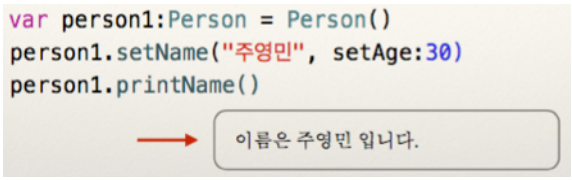
다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

3. 옵셔널

(1) nil 이란?

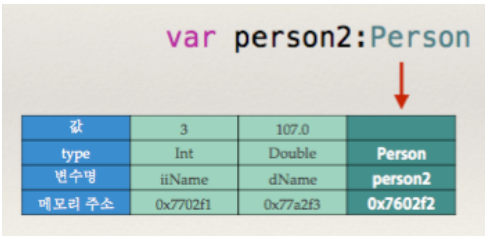
* 값이 없다.
* 0도 아님. 아예 값이 없는 것.
* var person1:Person = Person( )

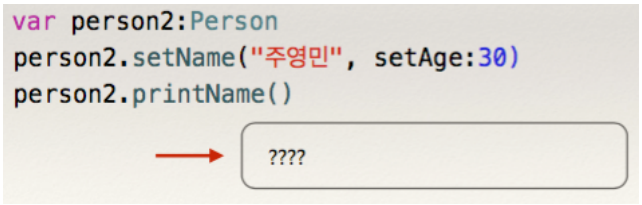




* 메모리 안에 객체를 만드는 것
* ( )안하고 .을 통해 함수를 부르면 에러가 난다.

* 만약, 선언만 한 상태라면?





* var person2:Person ==> nil: 변수 선언은 되었지만 값이 없는 상태

(2) 옵셔널 타입

* swift의 특징 중 하나인 안정성을 위해 만들었다.
* ?와 !가 핵심
* swift의 변수 또는 상수에서 값을 꺼낼 때 값이 nil인 경우를 위해 만들었다.
* 값을 꺼내올 때 nil체크를 한다.
* 옵셔널 변수(상수)는 해당 값에 nil이 들어가 있을 가능성이 있음을 명시하는 것.

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

(3) 옵셔널 타입 (! , ?)

* person! / person?
    * 옵셔널 person타입으로 지정되는 것
    * ! —> 절대 nil아니다 (이 경우는 보통 옵셔널을 안하고 값을 넣어줌)
    * ? —> nil일 수도 있고, 아닐 수도 있다.

* ! 일때, 값이 넣어지지 않았음 —> 프로그램이 멈춘다 1
* ? —> 초기화가 안되어있다면 run이 실행이 안되고, 밑에 코드로 넘어간다. (크러쉬가 나진 x) 2
* crush가 나는 것을 개발자 책임으로 넘김

[예제] 옵셔널 타입 테스트

일반 변수 vs 옵셔널 변수

? 와 ! 의 차이

```
var aaa:Int?
var bbb:Int!
var ccc:Int = 0


override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a
        nib.


    aaa = 3
    var ddd:Int = aaa!  //aaa 가 값이 있다는 것! ?를 쓰려면 var ddd:Int? = aaa?
    print(aaa)  Initialization of variable 'ddd' was never used; consider replacing with assignment to '_' or r...
    print(ccc)
    print(bbb)      // nil이 아니라해놓고 nil이어서 crush
                    Thread 1: EXC_BAD_INSTRUCTION (code=EXC_I386_INVOP, subcode=0x0)
```

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

(4) nil 검사

* 위 예제와 같은 충돌을 막기 위해, 함수 안에서 사용하는 변수가 nil인지 확인하는 방법

* nil직접 검사

```
func testFuc(list:[String]?)
{
    if list != nil
    {
        list![0]
    }
}
```

* nil 검사를 하긴 하지만 옵셔널을 계속 사용해야 한다.

* if문을 써서 검사하는 방법
* swift에선 이 방법보단 아래 방법이 더 많이 쓰임

* 옵셔널 바인딩

- nil값인지 확인 후 nil이 아니면 새로운 변수에 넣는다.
- nil값 확인 후 사용하는 것으로 안전하다.

```
func testFuc(list:[String]?)
{
    if let newList = list
    {
        newList[0]
        newList[1]
    }

}
```

* 옵셔널 변수를 새로운 변수에 넣는다.
* let 상수에 어떤 변수를 집어넣는다 -> 상수는 바꿀 수 없는 값이기 때문에 초기화할 때 값을 집어넣어야 한다 -> 초기화해서 넣는 값이 옵셔널이면 false가 되고, 값이 있으면 true가 된다

* Grard문

```
guard 조건값 else
{
  //조건값이 거짓일때 실행
}
```

* if의 반대 개념
* 조건이 거짓일 때 실행

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️

* Early Exit

```
func testFuc(list:[String]?)
{
    guard let newList = list  else
    {
        return
    }
    newList[0]
    newList[1]
}
```

거짓일 때, 실행

참일 때, 실행

* list가 nil 아니면 새로운 변수에 집어넣고, nil이면 끝내라 (return. 함수 종료)

다들 고생 많으셨습니다 :) IOS 입문반 화이팅 ❤️