

iOS 9주차 레포트

|  |  |
| --- | --- |
| 과목명 : iOS 프로그래밍 | 담당교수 : 한성현 교수님 |
| 학과 : 컴퓨터소프트웨어학과 | 학년 : 2학년 |
| 학번 : 201912047 | 이름 : 정정욱 |
| 제출일 : 2022.10.24 |  |

9주차 레포트 과제를 아래

<https://aloe-study.tistory.com/182>

블로그에 좀더 보기 좋게 정리하였습니다.

**기본 클레스 소스코드 틀**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**class** **func** cM(){

print("cM은 클래스 메서드입니다.")

}

**static** **func** scM(){

print("scM은 클래스 메서드(static)")

}

}

**var** kim : Man = Man()

kim.display() *//인스턴스 메서드는 인스턴스가 호출*

Man.cM() *//클래스 메서드는 클래스가 호출*

Man.scM() *//클래스 메서드는 클래스가 호출*

*// 나이=1, 몸무게=3.5*

*// cM은 클래스 메서드입니다.*

*// scM은 클래스 메서드(static)*

**기본 클레스 소스코드 틀 2**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age *//프로퍼티 = 매개변수*

**self**.weight = weight

}

}

**var** kim : Man = Man(age:10, weight:20.5)

kim.display()

*// 나이=10, 몸무게=20.5*

computed property를 만드는 1단계

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**var** manAge : Int {

}

*// computed property 만드는 1 단계 중괄호 묶어주기*

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age *//프로퍼티 = 매개변수*

**self**.weight = weight

}

}

**var** kim : Man = Man(age:10, weight:20.5)

kim.display()

*// 나이=10, 몸무게=20.5*

**2단계**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**var** manAge : Int {

**get**{

**return** age-1

}

}

*// computed property 만드는 2 단계 getter 사용*

*// 따로 외부에서 사용 하는게 아니라 어떻게 사용하겠다 만들어주는*

*// 것임 return 필수로 해줘야 함*

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age *//프로퍼티 = 매개변수*

**self**.weight = weight

}

}

**var** kim : Man = Man(age:10, weight:20.5)

kim.display()

print(kim.manAge)

*// 나이=10, 몸무게=20.5*

*// 9 // 넣어준 적 없지만 잘 나옴*

**getter 생략**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**var** manAge : Int {

*// get{*

**return** age-1

*// }*

}

*//setter가 없으면 get{ }는 생략할 수 있으며 변경하지 않더라도 var로 선언해야 함*

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age *//프로퍼티 = 매개변수*

**self**.weight = weight

}

}

**var** kim : Man = Man(age:10, weight:20.5)

kim.display()

print(kim.manAge)

*// 나이=10, 몸무게=20.5*

*// 9 // 넣어준 적 없지만 잘 나옴*

**getter와 setter를 갖는 computed property**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**var** manAge : Int{

**get**{

**return** age-1

}

**set**(USAAge){

age = USAAge + 1

}

}

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age

**self**.weight = weight

}

}

**var** kim : Man = Man(age:10, weight:20.5)

kim.display()

print(kim.manAge) *//9, getter호출*

print(kim.age) *//10*

kim.manAge = 3 *//setter호출*

print(kim.age) *//4*

**Shorthand Setter Declaration**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**var** manAge : Int{

**get**{ **return** age-1 }

**set**{ age = newValue + 1 }

}

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age

**self**.weight = weight

}

}

**var** kim : Man = Man(age:10, weight:20.5)

kim.display()

print(kim.manAge) *//9, getter호출*

print(kim.age) *//10*

kim.manAge = 3 *//setter호출*

print(kim.age) *//4*

*// setter가 있으면 get{ }는 생략할 수 없음*

*// 매개변수명은 newValue가 기본*

*// set(newValue){*

*// age = newValue + 1*

*// }*

*// Shorthand Setter Declaration*

*// setter의 매개변수명이 newValue인 경우에만 이렇게 “(newValue)” 생략 가능*

*// set{*

*// age = newValue + 1*

*// }*

[**https://ko.wikipedia.org/wiki/%ED%95%A8%EC%88%98\_%EC%98%A4%EB%B2%84%EB%A1%9C%EB%93%9C**](https://ko.wikipedia.org/wiki/%ED%95%A8%EC%88%98_%EC%98%A4%EB%B2%84%EB%A1%9C%EB%93%9C)

**method overloading : 생성자 중첩 및 일반 메서드 중첩 (주의사항 까지)**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**func** display(age:Int){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

*// 매개변수의 개수와 자료형이 달라야 오류 안남*

*// func display(){*

*// print("나이=\(age), 몸무게=\(weight)")*

*// }*

**init**(age: Int, weight : Double){ *//1*

**self**.age = age

**self**.weight = weight

}

**init**(age: Int){ *//2*

**self**.age = age

}

}

**var** kim : Man = Man(age:10, weight:20.5) *//1*

**var** kim1 : Man = Man(age:10) *//2*

*//var kim2 : Man = Man()*

*//init가 없다면 인스턴스 만드는 방법*

kim.display()

kim1.display()

kim1.display(age:3)

*// 나이=10, 몸무게=20.5*

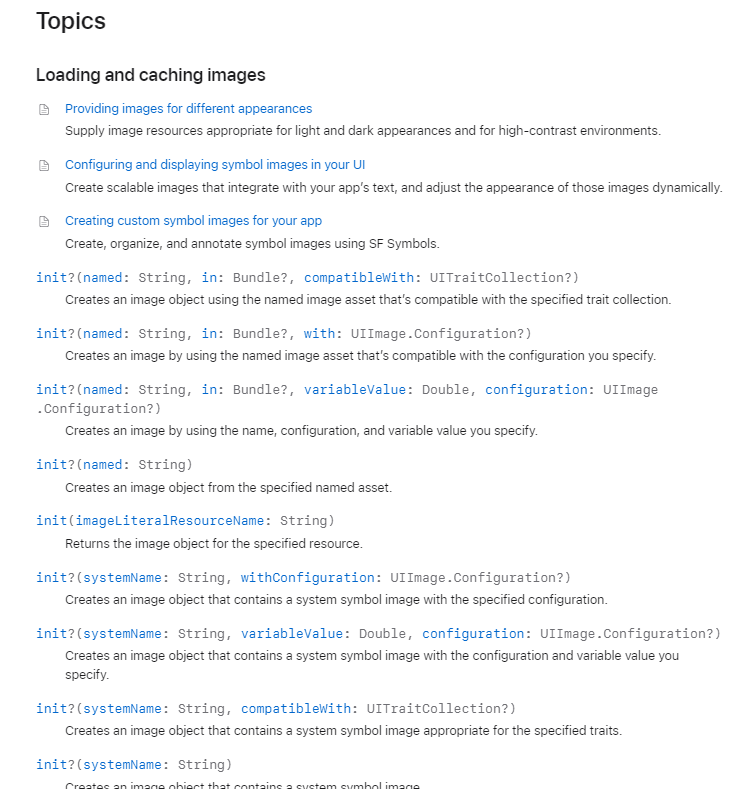
*// 나이=10, 몸무게=3.5*

*// 나이=3, 몸무게=3.5*

[**https://developer.apple.com/documentation/uikit/uiimage**](https://developer.apple.com/documentation/uikit/uiimage)

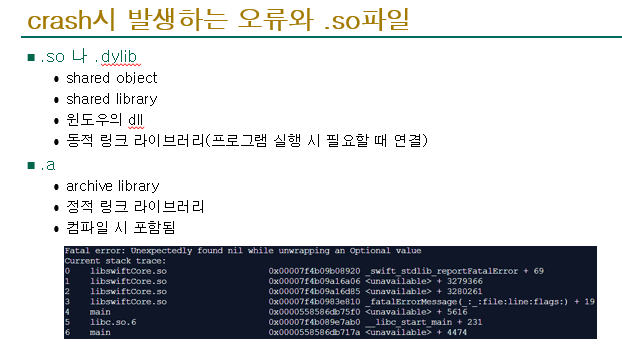
**swift 최상 단 클레스의 있는 NS 는 next step 약자임**

**문서에 다양한 생성자들을 사용할 수 있게 오버라이드 되어있음**



이렇게 사용하는 이유는 개발자들이 자신이 만든 클레스가 아니기 때문에 어려움을 느낄 수 있어

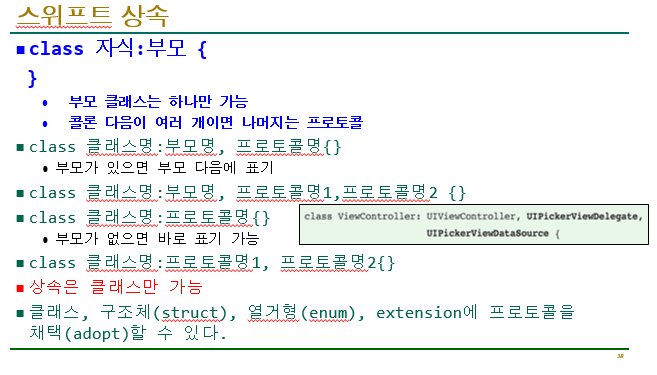
객체지향언어의 가장 큰 목적인 재사용에 부합하기 위해 즉 여러 개발자가 내가 만든 클레스가 아닌 데도 불구하고 인스턴스를 생성하기 편하게 위함이다.



**앱 개발 시 crash가 날 때 봐야할 파일들**

UIColor 예제

[**http://seorenn.blogspot.com/2015/02/swift-hex-uicolor.html**](http://seorenn.blogspot.com/2015/02/swift-hex-uicolor.html)



**C++을 제외한 대부분 언어 단일 상속임 그렇기 때문에 부모는 하나만 올 수 있음**

용어정리  
부모 클레스를 자식 클레스가 상속받았다고 함  
  
조금 이따 배울 프로토콜은 자바의 인터페이스와 비슷한 개념이며  
여러 개를 채택할 수 있음 프로토콜은 상속이 아니라 채택한다고 함

**단일 상속 (single inheritance)**

* 1. **Swift에서 하위 클래스는 단 하나의 부모 클래스만 상속받을 수 있음**

**상속**

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age

**self**.weight = weight

}

}

**class** **Student** : Man {

*//비어 있지만 Man의 모든 것을 가지고 있음*

}

*//var kim : Man = Man.init(age:10, weight:20.5) 생성자도 함수라 이렇게 사용할 수 있음 여담*

**var** kim : Man = Man(age:10, weight:20.5)

kim.display()

**var** lee : Student = Student(age:20,weight:65.2)

lee.display()

print(lee.age)

*// 나이=10, 몸무게=20.5*

*// 나이=20, 몸무게=65.2*

*// 20*

1. **상속된 클래스는 부모 클래스의 모든 기능을 상속받으며, 자신만의 기능을 추가**

**super : 부모 메서드 호출 시 사용**

**class** **Man**{

**var** age : Int

**var** weight : Double

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age

**self**.weight = weight

}

}

**class** **Student** : Man {

**var** name : String

**func** displayS() {

print("이름=**\(**name**)**, 나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double, name : String){

**self**.name = name

**super**.**init**(age:age, weight:weight) *//과제: 이 줄을 안쓰면?*

}*//error:'super.init' isn't called on all paths before returning from initializer*

}

*// 부모에 프로퍼티에 초기 값이 없어서 이렇게 안해주면 오류가 남 그렇기 때문에 이렇게*

*// super 키워드로 부모 생성자에 접근*

**var** lee : Student = Student(age:20,weight:65.2,name:"홍길동")

lee.displayS()

lee.display()

*// 이름=홍길동, 나이=20, 몸무게=65.2*

*// 나이=20, 몸무�=65.2*

*// init(age1: Int, weight1 : Double, name : String){*

*// super.init(age:age1, weight:weight1)*

*// self.name = name*

*// }*

*// }*

*// var lee : Student = Student(age1:20,weight1:65.2,name:"홍길동")*

**// 즉 넘어오는 걸 부모 쪽으로 바로 주는 것임 이해를 돕기 위한 소스**

용어정리

부모 클레스를 자식 클레스가 상속받았다고 함

조금 이따 배울 프로토콜은 자바의 인터페이스와 비슷한 개념이며  
여러 개를 채택할 수 있음 프로토콜은 상속이 아니라 채택한다고 함

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){

print("나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double){

**self**.age = age

**self**.weight = weight

}

}

**class** **Student** : Man {

**var** name : String = "김소프"

**override** **func** display() {

print("이름=**\(**name**)**, 나이=**\(**age**)**, 몸무게=**\(**weight**)**")

}

**init**(age: Int, weight : Double, name : String){

**super**.**init**(age:age, weight:weight)

**self**.name = name

}

}

**var** lee : Student = Student(age:20,weight:65.2,name:"홍길동")

lee.display()

*// 이름=홍길동, 나이=20, 몸무게=65.2*

1. **부모와 자식에 display()라는 메서드가 있어서 Student클래스는 display() 메서드가 두 개임**
2. **Student클래스의 인스턴스 lee가 display()를 호출할 때, 자식클래스가 새로 만든 display() 메서드가 우선적으로 호출되려면 func 앞에 override키워드 씀**

**내가 만든 걸 최우선 적용해줘라 : override**

**애플 사용자 인터페이스 디자인 가이드라인 사이트**

1. [**https://developer.apple.com/design/human-interface-guidelines/components/all-components**](https://developer.apple.com/design/human-interface-guidelines/)
2. [**https://developer.apple.com/design/human-interface-guidelines/**](https://developer.apple.com/design/human-interface-guidelines/)

**extension**

1. **class, struct, enum, protocol에 새로운 기능을 추가**
2. [**https://developer.apple.com/documentation/swift/double**](https://developer.apple.com/documentation/swift/double)

**현제 더블형에 없는 기능을 확장 (기존 기능을 건드는 게 아님)**

**extension** **Double** {

**var** squared : Double { *//기존 Double형 구조체에 계산 프로퍼티 추가*

**return** **self** \* **self**

}

**var** up : Double { *//기존 Double형 구조체에 계산 프로퍼티 추가*

**return** **self** + 1.0

}

}

**let** myValue: Double = 3.0

print(myValue.squared)

print(2.0.squared) *//Double형 값에도 .으로 바로 사용 가능, 과제*

**let** myValue2: Double = 3.0

print(myValue2.up)

print(2.0.up) *//Double형 값에도 .으로 바로 사용 가능*

*// 9.0*

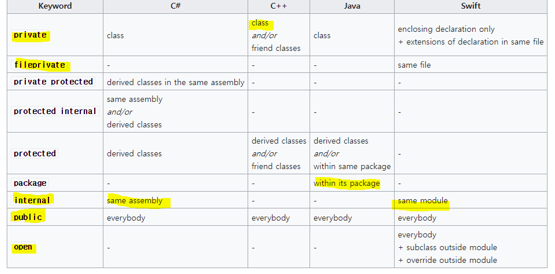
*// 4.0*

*// 4.0*

*// 3.0*

**access modifier(or access specifiers)**

1. **접근 속성(접근 수정자, 액세스 수정자, 액세스 지정자 )는 클래스, 메서드, 멤버의 접근 가능성을 설정하는 객체 지향 언어의 키워드**



Swift는 현제 5 가지가 디폴트 접근 속성임

**public** **class** MyClass{

*// 모듈의 모든 소스 파일 내에서 접근+정의한 모듈을 가져오는 다른 모듈의 소스파일에서도 접근 가능*

fileprivate **var** name : String = "Kim"

*//현재 소스 파일 내에서만 사용 가능*

**private** **func** play() {}

*//현재 블럭 내에서만 사용 가능*

**func** display(){} *//internal은 디폴트 속성으로 생략됨*

*//해당 모듈의 모든 소스 파일 내에서 사용*

}

* 1. **모듈은 코드 배포(code distribution)의 단일 유닛**
     1. **앱, Framework(UIKit 등),외부라이브러리**
     2. **import Alamofire**
  2. **open 접근 및 public 접근을 통해 모듈의 모든 소스 파일 내에서 사용할 수 있으며, 정의한 모듈을 가져오는 다른 모듈의 소스파일에서도 사용할 수 있다. 일반적으로 Framework에 공용 인터페이스를 지정할 때는 open 또는 public 접근을 사용한다.** 
     1. **open 접근은 클래스 및 클래스 멤버에만 적용**
  3. **internal 접근은 해당 모듈의 모든 소스 파일 내에서 사용되지만, 해당 모듈 외부의 소스파일에서는 사용되지 않도록 한다. 일반적으로 App이나 Framework의 내부 구조를 정의할 때 internal 접근을 사용한다.**
     1. **접근 제어를 생략하면 internal이 기본**
  4. **fileprivate 접근은 해당 소스 파일 내에서만 사용가능**
  5. **private접근은 블록과 동일한 파일에 있는 해당 선언의 extension으로 제한한다.**

**Type Property**

**class** **SClass** {

**var** storedProperty = 2

**static** **var** storedTypeProperty = 1

**static** **var** computedTypeProperty: Int {

**return** 10

}

**class** **var** overrideableComputedTypeProperty: Int {

**return** 100

}

}

**var** x = SClass()

print(x.storedProperty) *// 인스턴스가 가지고 놈*

print(SClass.storedTypeProperty)

print(SClass.computedTypeProperty)

print(SClass.overrideableComputedTypeProperty)

*// 2*

*// 1*

*// 10*

*// 100*

*// static, class 붙은 애들은 클레스가 가지고 노는 것임*

*Class 붙은 애들은 상속이 가능함!!!!!!*

***프로토콜(protocol)***

1. *자바, C#의 interface*

미리 정의해준다고 생각 이러이러한 거 필요하니까 잊지 말고 사용해라

**protocol** **Runnable** { *//대리하고 싶은 함수 목록 작성*

**var** x : Int {**get** **set**} *//읽기와 쓰기 가능 프로퍼티,{get}은 읽기 전용*

*//property in protocol must have explicit { get } or { get set } specifier*

**func** run() *//메서드는 선언만 있음*

}

**class** **Man** : Runnable { *//채택, adopt*

**var** x : Int = 1 *//준수, conform*

**func** run(){print("달린다~")} *//준수, conform*

}

**let** woogie = Man()

woogie.run()

*// class Man에 x, run()정의 없다면*

*// type 'Man' does not conform to protocol 'Runnable‘*

*// 즉 준수 해줘야 함 프로퍼티에는 값을 넣어줘야 하고 메서드에는 기능을 넣어야 함*

*// 달린다~*

*과제 문제풀기*

*// 다음과 같은 기능을 전체 하나의 소스로 작성.*

*// bb()라는 메소드가 하나 있는 프로토콜(protocol) B을만드시오.*

*// bb()메서드는 Int형 값을 매개변수로 받아 Int형으로 리턴 하는 메서드이다.*

*// 클래스 A는 클래스 C라는 부모를 갖는다(C로부터 상속받는다).*

*// 클래스 A에서 프로토콜 B를 채택하고, 준수하는 소스를 작성.*

*// bb()메서드는 Int형 값을 매개변수로 받아 두배 한 값을 Int형으로 리턴 하는 메서드이다.*

*// 클래스 A의 인스턴스 a를 만들고 bb를 호출(argument로 3을 입력)하여 결과를 확인하세요*

**protocol** **B** {

**func** bb(x: Int) -> Int

}

**class** **C** { *//채택, adopt*

}

**class** **A** : C, B {

**func** bb(x : Int) -> Int{

**return** x \* 2

}

}

**var** a : A = A()

print(a.bb(x:3))

*// 6*