화살이(가) 표시된 사진

자동 생성된 설명

iOS 6주차 레포트

|  |  |
| --- | --- |
| 과목명 : iOS 프로그래밍 | 담당교수 : 한성현 교수님 |
| 학과 : 컴퓨터소프트웨어학과 | 학년 : 2학년 |
| 학번 : 201912047 | 이름 : 정정욱 |
| 제출일 : 2022.09.29 |  |

텍스트이(가) 표시된 사진

자동 생성된 설명

**first class object : (1) 함수를 변수에 저장 가능**

*// Swift는 함수를 데이터 타입처럼 처리할 수 있음*

*// 다음과 같이 함수를 상수 또는 변수에 할당하는 것이 가능*

**func** inchesToFeet (inches: Float) -> Float {

**return** inches \* 0.0833333

}

**let** toFeet = inchesToFeet *//함수를 자료형처럼 사용*

*//함수를 호출하려면 원래의 함수 이름 대신에 상수 이름을 이용하여 호출 가능*

print(inchesToFeet(inches:10))

print(type(of:toFeet))

*print(toFeet(10)) //주의 : 매개변수명(inches:) 안씀 : 레이블 없이 넘겨주면 됨c 처럼*

*// 0.833333*

*// (Float) -> Float 함수의 자료형과 똑같음 그렇기 때문에 함수의 자료형 잘 확인할 수 있어야함*

*// 0.833333*

텍스트이(가) 표시된 사진

자동 생성된 설명

**first class object : (2) 함수를 매개변수로 사용**

*// Swift는 함수를 데이터 타입처럼 처리할 수 있음*

*// 다음과 같이 함수를 상수 또는 변수에 할당하는 것이 가능*

**func** inchesToFeet (inches: Float) -> Float {

**return** inches \* 0.0833333

}

**let** toFeet = inchesToFeet *//함수를 자료형처럼 사용*

*//함수를 호출하려면 원래의 함수 이름 대신에 상수 이름을 이용하여 호출 가능*

print(inchesToFeet(inches:10))

print(type(of:toFeet))

print(toFeet(10)) *//주의 : 매개변수명(inches:) 안 씀 : 레이블 없이 넘겨주면 됨c 처럼*

**func** outputConversion(converterFunc: (Float) -> Float, value: Float) {*//함수를 매개변수로 사용*

*//첫번째 매개변수를 함수를 받음 (Float) -> Float 이런 형의 함수를 받겠다*

**let** result = converterFunc(value) *//toFeet(10)*

print("Result = **\(**result**)**")

}

outputConversion(converterFunc:toFeet, value: 10) *// 피트로 변환하는 inchesToFeet함수 호출*

*// 0.833333*

*// (Float) -> Float*

*// 0.833333*

*// Result = 0.833333*

*// 즉 매개변수가 (Float) 형이고 리턴 값이 Float형인 함수를 넣어주겠다 라는 것임*

텍스트이(가) 표시된 사진

자동 생성된 설명

**first class object : (3) 함수를 리턴 값으로 사용**

**func** inchesToFeet (inches: Float) -> Float {

**return** inches \* 0.0833333

}

**func** inchesToYards (inches: Float) -> Float {

**return** inches \* 0.0277778

}

**let** toFeet = inchesToFeet

**let** toYards = inchesToYards

*//반환 타입으로 함수의 타입을 선언하면 함수도 반환될 수 있음*

*//다음 함수는 Boolean 매개변수의 값에 따라 toFeet 함수 또는 toYards 함수를 반환*

**func** decideFunction (feet: Bool) -> (Float) -> Float

{ *//매개변수형 리턴형이 함수형*

**if** feet {

**return** toFeet *//함수를 리턴*

} **else** {

**return** toYards

}

}

텍스트이(가) 표시된 사진

자동 생성된 설명

이 3가지를 다 만족하면 그 언어의 함수는 1급 객체라고 부를 수 있는 것임

그래서 스위프트 함수는 일급 객체임 아니면 일급 시민이다 first class object, first class citizen 다 가능

텍스트이(가) 표시된 사진

자동 생성된 설명

위 소스 코드는 1급 객체를 만족하는 3가지 조건을 해당 사항에 맞게 번호를 마킹 한 것이다.

**func** up(num: Int) -> Int {

**return** num + 1

}

**func** down(num: Int) -> Int {

**return** num - 1

}

**let** toUp = up

print(up(num:10))

print(toUp(10))

**let** toDown = down

**func** upDown(Fun: (Int) -> Int, value: Int) {

**let** result = Fun(value)

print("결과 = **\(**result**)**")

}

upDown(Fun:toUp, value: 10) *//toUp(10)*

upDown(Fun:toDown, value: 10) *//toDown(10)*

**func** decideFun(x: Bool) -> (Int) -> Int {

*//매개변수형 리턴형이 함수형*

**if** x {

**return** toUp

} **else** {

**return** toDown

}

}

**let** r = decideFun(x:**true**) *// let r = toUp : 함수를 리턴해서 변수에 저장*

print(type(of:r)) *//(Int) -> Int*

print(r(10)) *// toUp(10)*

*// 11*

*// 11*

*// 결과 = 11*

*// 결과 = 9*

*// (Int) -> Int*

*// 11*

함수 : 일급 객체 응용해보기

**func** up(name: String, age: Int) -> String {

**return** "**\(**name**)**는 **\(**age + 1**)**"

}

**func** down(name: String, age: Int) -> String {

**return** "**\(**name**)**는 **\(**age - 1**)**"

}

**let** toUp = up

print(up(name:"우기", age:23))

print(toUp("부기", 22))

**let** toDown = down

**func** upDown(Fun: (String, Int) -> String, name: String, age: Int) {

**let** result = Fun(name, age)

print("결과 = **\(**result**)**")

}

upDown(Fun:toUp, name:"우기", age:23) *//toUp(우기,23)*

upDown(Fun:toDown, name:"우기", age:23) *//toDown(우기,23)*

**func** decideFun(x: Bool) -> (String, Int) -> String {

*//매개변수형 리턴형이 함수형*

**if** x {

**return** toUp

} **else** {

**return** toDown

}

}

**let** r = decideFun(x:**true**) *// let r = toUp : 함수를 리턴해서 변수에 저장*

print(type(of:r))

print(r("징니", 23)) *// toUp(10)*

*// 우기는 24*

*// 부기는 23*

*// 결과 = 우기는 24*

*// 결과 = 우기는 22*

*// (String, Int) -> String*

*// 징니는 24*

**함수를 클로저로 바꿔보기 실습**

**func** add(x: Int, y: Int) -> Int {

**return**(x+y)

}

print(add(x:10, y:20))

**let** add1 = { (x: Int, y: Int) -> Int **in**

**return**(x+y)

}

*//print(add1(x:10, y:20)) //주의 error: extraneous(관련 없는) argument labels 'x:y:' in call*

print(add1(10, 20)) *//OK*

print(type(of:add1)) *//과제*

*// 함수를 클로저로 바꿀때는 앞에 fun 키워드와 이름을 지우고*

*// 블락을 땡겨 전체로 묶은 다음 in 키워드를 사용해줌*

*// 일종의 함수이니 변수에 넣을수 있음 : 즉 일급 객체이기 때문에 대입 가능*

*// 클로저를 만드는 이유는 한번만 호출될거 같을때 함수를 따로 만들 필요 없이 그 자리에 기능을 넣어주겠다 할때 많이 사용함*

*// 30*

*// 30*

*// (Int, Int) -> Int*

**func** mul(val1: Int, val2: Int) -> Int

{

**return** val1 \* val2

}

**let** result = mul(val1:10, val2:20)

print(result)

**let** multiply = {(val1: Int, val2: Int) -> Int **in**

*//매개변수 리턴형*

**return** val1 \* val2

}*// 여기서 multiply의 자료형은 (Int, Int) -> Int*

**let** result2 = multiply(10, 20) *//상수를 함수처럼 호출,200*

print(result2)

print(type(of:multiply))

*// 200*

*// 200*

*// (Int, Int) -> Int*

텍스트이(가) 표시된 사진

자동 생성된 설명

즉 어떠한 함수에서 매개변수로 함수를 받을 때 클로저를 넘겨줄 수 있음

중간에 작성한 코드처럼 저렇게 넘기는 방법이 기본적인 방법이지만

가독성을 위해서 클로저가 함수의 마지막 argument라면 마지막 매개변수명을 생략한후 소괄호 뒤에 클로저를 작성해 넘겨줄 수 있음

**+**

**나중에 배울 클래스 문법에서 사용할**

**아래와 같은 함수를 보면**

텍스트이(가) 표시된 사진

자동 생성된 설명

저렇게 마지막 매개변수로 함수를 받게 되어있음

텍스트이(가) 표시된 사진

자동 생성된 설명

일반적으로 위 코드를 활용한다고 할 때 저렇게 클로저를 사용해 넘겨줄 수 있지만 클로저가 함수의 마지막 argument 이기 때문에

텍스트이(가) 표시된 사진

자동 생성된 설명

이렇게 클로저를 함수 뒤쪽에 따로 빼서 넘겨줄 수 있다는 것임

이것을 “후행 클로저”라고 함

참조 문헌

<https://developer.apple.com/documentation/uikit/uialertaction/1620097-init>

소스가 길어지니까 가독성을 위한 것

이것을 알아야

나중에 앱 개발을 하거나 문서를 볼 때 당황하지 않고 무엇인지 알 수 있음

**클로저의 축약 표현들**

*/\**

*아래와 같은 함수를 클로저로 바꿔 보고 다양한 클로저 축약 표현들을 익혀보자*

*func mul(val1: Int, val2: Int) -> Int*

*{*

*return val1 \* val2*

*}*

*let result = mul(val1:10, val2:20)*

*\*/*

**let** multiply = {(val1: Int, val2: Int) -> Int **in**

**return** val1 \* val2

}

**var** result = multiply(10, 20)

print(result)

**let** add = {(val1: Int, val2: Int) -> Int **in**

**return** val1 + val2

}

result = add(10, 20)

print(result)

*// 함수 대신 클로저를 활용하여 상수에 값을 저장하기*

**func** math(x: Int, y: Int, cal: (Int, Int) -> Int) -> Int { *// 일급 시민 함수를 마지막 매개변수로 받기*

**return** cal(x, y)

}

result = math(x: 10, y: 20, cal: add)

print(result)

result = math(x: 10, y: 20, cal: multiply)

print(result)

result = math(x: 10, y: 20, cal: {(val1: Int, val2: Int) -> Int **in**

**return** val1 + val2

}) *//클로저 소스를 매개변수에 직접 작성*

print(result)

*// 클로저가 함수의 마지막 argument라 매개 변수 이름을 생략후 : 후행 클로저로 변환*

result = math(x: 10, y: 20) {(val1: Int, val2: Int) -> Int **in**

**return** val1 + val2

}*//trailing closure*

print(result)

*// 클로저의 다양한 축약 표현들*

result = math(x: 10, y: 20, cal: {(val1: Int, val2: Int) **in** *// -> Int*

**return** val1 + val2

}) *//리턴형 (반환 타입) 생략*

print(result)

result = math(x: 10, y: 20) {(val1: Int, val2: Int) **in** *// -> Int*

**return** val1 + val2

} *//trailing closure, 리턴형 (반환 타입)생략*

print(result)

result = math(x: 10, y: 20, cal: {

**return** $0 + $1 *//$0 + $1 숫자 증가에 따라 순서대로 들어오는 매개변수 순서가 됨*

}) *//매개변수 생략하고 단축인자(shorthand argument name)사용*

print(result)

result = math(x: 10, y: 20) {

**return** $0 + $1

} *//trailing closure, 매개변수 생략하고 단축인자사용*

print(result)

result = math(x: 10, y: 20, cal: {

$0 + $1

}) *//클로저에 리턴 값이 있으면 마지막 줄을 리턴 하므로 return생략*

print(result)

result = math(x: 10, y: 20) { $0 + $1 } *//return 생략*

print(result)

*// 200*

*// 30*

*// 30*

*// 200*

*// 30*

*// 30*

*// 30*

*// 30*

*// 30*

*// 30*

*// 30*

*// 30*

*// 클로저 사람마다 생략(축약)하는게 다 다름 이렇게 다양하게 축약해서 사용 할 수 있다는 것을 알아야함*

*// 다른 사람 코드를 보더라도 문제없이 어떤 건지 알 수 있음*

*// 너무 많이 축약하면 오히려 협업 시 가독성이 떨어질 수 있음(역설적으로) 그렇기 때문에 적당히 줄여서 사용 해야함*

*// 이 모든 게 다 같은 소스라는 걸 알고 있어야함*

참조 문서

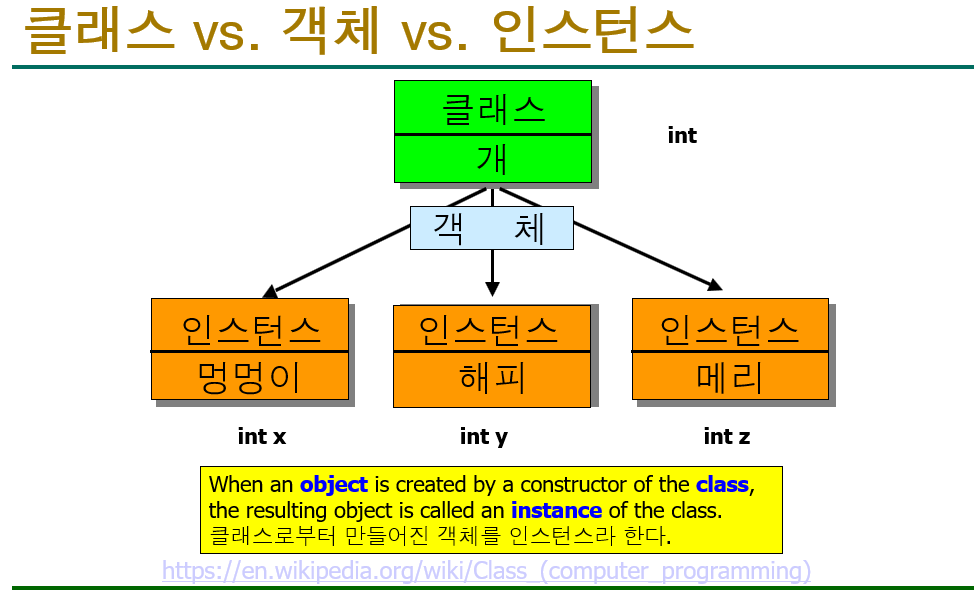
<https://docs.swift.org/swift-book/LanguageGuide/Closures.html>

추가 공부 자료

<https://aloe-study.tistory.com/121>

<https://aloe-study.tistory.com/122>

**클래스 vs. 객체 vs. 인스턴스**



<https://en.wikipedia.org/wiki/Class_(computer_programming)>

우리나라에서는 인스턴스라는 말 잘 사용 x

영어권에서는 인스턴스라는 말을 많이 사용

객체 지향 언어의 핵심은 재사용임

덩치가 커지면 재사용하기가 쉽지가 않음 코드가 어디부터 어디까지 인가? 라는 의문도 들고 그래서 대부분 객체지향 언어의서 재사용 단위가 클래스임

이런 것 들이 모여 있는 것을 SDK라고 하며

그게 커지고 어떤 틀 안에서 내가 작업하는 것을 프레임 워크라고 함

<https://aloe-study.tistory.com/158>

+ 프레임 워크 용어 정리

테이블이(가) 표시된 사진

자동 생성된 설명

객체지향 언어 별 용어를 비교하면 위와 같은 형태로 나타나는 것을 알 수 있음 일반 적으로 클래스 안에 선언되어 있는 변수를 Swift에서 프로퍼티라고 하며 클래스 안에 선언되어 있는 함수를 메소드라고 부름

텍스트이(가) 표시된 사진

자동 생성된 설명

* [**https://docs.swift.org/swift-book/LanguageGuide/ClassesAndStructures.html**](https://docs.swift.org/swift-book/LanguageGuide/ClassesAndStructures.html)

스위프트에서 구조체와 클래스 많이 사용하는데 이 두개가 어떤 차이가 있는지 설명할 수 있어야함

문법 구조는 비슷하지만 상속 같은 기능을 이용하겠다 하면 클래스를, 간단하게 데이터를 가지고 놀겠다 하면 구조체를 사용하면 됨 정답은 없음

텍스트이(가) 표시된 사진

자동 생성된 설명

가장 큰 특징을 비교하면 구조체, 열거형 둘 다 일반적인 값 타입이고

클래스는 참조(주소) 타입임

<https://aloe-study.tistory.com/120> 추가 공부 자료

**그렇다면 클래스란 무엇인가?**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

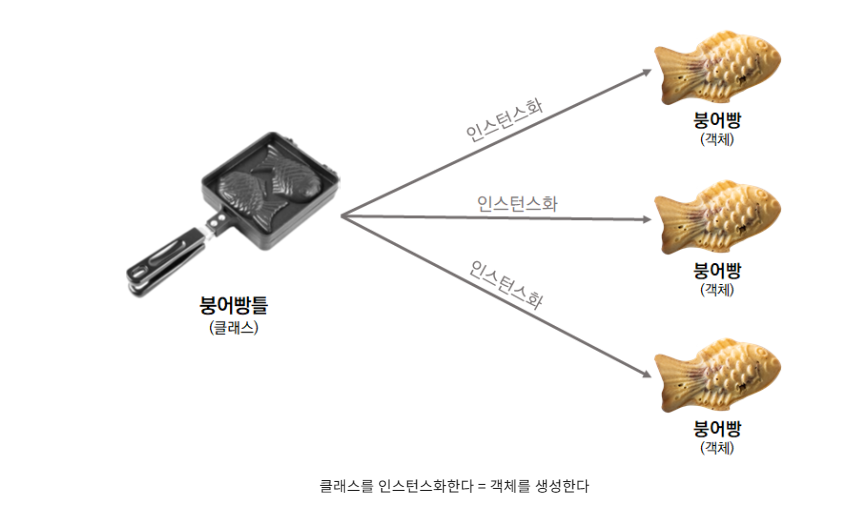
//쉽게 새로운 자료형을 만드는 것이라고 생각하자

**+ 클래스를 통해 생성한 인스턴스는 무엇인가?**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**객체의 구성을 보면 클래스에서 정의한 변수, 메서드를 사용할 수 있음(찍어낸 것이라)**

****

[**https://blog.hexabrain.net/104**](https://blog.hexabrain.net/104)

**참고 자료 (자바와 문법 비교)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

상속을 받을 때만: 부모 클래스 해주고

그냥 독자적인 클래스면 필요 없음

ex 자바에서는: 대신 extends

클래스에서 메서드는 두 가지임 (인스턴스, 타입) 메서드

**프로퍼티(property) : Swift 에서 제공하는 프로퍼티는 2가지로 나뉨**

* [**https://docs.swift.org/swift-book/LanguageGuide/Properties.html**](https://docs.swift.org/swift-book/LanguageGuide/Properties.html)

**텍스트이(가) 표시된 사진

자동 생성된 설명**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

*// 프로퍼티는*

*// 초기값이 있거나*

*// init을 이용해서 초기화하거나*

*// 옵셔널 변수(상수)로 선언(자동으로 nil로 초기화)*

**class** **Man**{

**var** age : Int

**var** weight : Double

} *//오류*

*// 초기 값이 없어서 에러가 남*

*// main.swift:1:7: error: class 'Man' has no initializers*

*// class Man{*

*// ^*

*// main.swift:2:6: note: stored property 'age' without initial value prevents synthesized initializers*

*// var age : Int*

**var** a : Int

*// 일반 변수는 초기 값이 있어도 없어도 되지만*

*// 클래스에 저장 프로퍼티는 초기 값이 없으면 안됨*

*// 1, 2, 3 조건을 따라야함*

텍스트이(가) 표시된 사진

자동 생성된 설명

**class** **Man**{

**var** age : Int? *// var age : Int? = nil 이렇게 자동으로 nil 값 들어 감*

**var** weight : Double!

}

*//이렇게 옵셔널 변수로 만드는 것임*

**class** **Man2**{

**var** age : Int = 1

**var** weight : Double = 3.5

}

텍스트이(가) 표시된 사진

자동 생성된 설명

자바에 static 메소드처럼 타입 메소드가 있음

이렇게 func 앞에 아무것도 없으면 일반적인 인스턴스 메서드

저 앞에 class or type이 들어가면 타입 메서드임 (클래스가 가지고 노는)

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){ *//인스턴스 메서드*

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

}

}

**var** kim : Man

print(kim.age)

*// 이러한 에러가 남*

*// 주의 할 점 위와 같이 코드를 작성하면 제대로 인스턴스가 만들어 지지 않음*

*// 인스턴스 만들 때 조심 해야함*

*// main.swift:13:7: error: variable 'kim' used before being initialized*

*// print(kim.age)*

*// ^*

*// main.swift:12:5: note: variable defined here*

*// var kim : Man*

*// ^*

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){ *//인스턴스 메서드*

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

}

}

**var** x : Int = 3

*//var kim : Man*

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

**var** kim2 = Man() *// 이렇게 생략 가능 함*

print(kim.age) *// 1*

*// Swift 변수를 만들 때 는 그냥 저렇게 만들어도 되지만*

*// 클래스의 인스턴스 객체를 만들 때 는*

*// 반드시 = 클래스 () 이렇게 작성 해야함*

*// initialize 함수때문에 그럼*

*// 자바처럼 클래스를 만들 때 자동으로 생성해 주는 함수인데*

*// Swift에서도 반드시 객체 만들 때 는 initialize 함수 이용해서 만들어야 함*

텍스트이(가) 표시된 사진

자동 생성된 설명