화살이(가) 표시된 사진

자동 생성된 설명

iOS 7주차 레포트

|  |  |
| --- | --- |
| 과목명 : iOS 프로그래밍 | 담당교수 : 한성현 교수님 |
| 학과 : 컴퓨터소프트웨어학과 | 학년 : 2학년 |
| 학번 : 201912047 | 이름 : 정정욱 |
| 제출일 : 2022.10.18 |  |

**let** weight = 60.0

**let** height = 170.0

**let** bmi = weight / (height\*height\*0.0001) *// kg/m\*m*

**var** body = ""

**if** bmi >= 40 {

body = "3단계 비만"

} **else** **if** bmi >= 30 && bmi < 40 {

body = "2단계 비만"

} **else** **if** bmi >= 25 && bmi < 30 {

body = "1단계 비만"

} **else** **if** bmi >= 18.5 && bmi < 25 {

body = "정상"

} **else** {

body = "저체중"

}

print("BMI:**\(**bmi**)**, 판정:**\(**body**)**")

*// BMI:20.761245674740483, 판정:정상*

함수로 만들기

// 함수 만드는거 알고 있어야함

**import** **Foundation**

**func** calcBMI(weight : Double, height : Double) -> String{

**let** bmi = weight / (height\*height\*0.0001) *// kg/m\*m*

**let** shortenedBmi = String(format: "%.1f", bmi)

**var** body = ""

**if** bmi >= 40{

body = "3단계 비만"

} **else** **if** bmi >= 30 && bmi < 40 {

body = "2단계 비만"

} **else** **if** bmi >= 25 && bmi < 30 {

body = "1단계 비만"

} **else** **if** bmi >= 18.5 && bmi < 25 {

body = "정상"

} **else** {

body = "저체중"

}

**return** "BMI:**\(**shortenedBmi**)**, 판정:**\(**body**)**"

}

print(calcBMI(weight:62.5, height: 172.3))

*//BMI:21.1, 판정:정상*

*// 함수 만드는거 알고 있어야함*

1급 객체, 1급 시민

**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)) *// 레이블 사용하면 안됨 오류남*

**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

}

}

**var** r = decideFun(x:**true**) *// let r = toUp*

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

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

r = decideFun(x:**false**)

print(type(of:r))

print(r(10))

*// 11*

*// 11*

*// 결과 = 11*

*// 결과 = 9*

*// (Int) -> Int*

*// 11*

*// (Int) -> Int*

*// 9*

// 클로저 : 30점 나옴

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

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

*// 레이블 보내지 않음 1급 객체 1번 조건 똑같음*

print(type(of:add1))

*// 30*

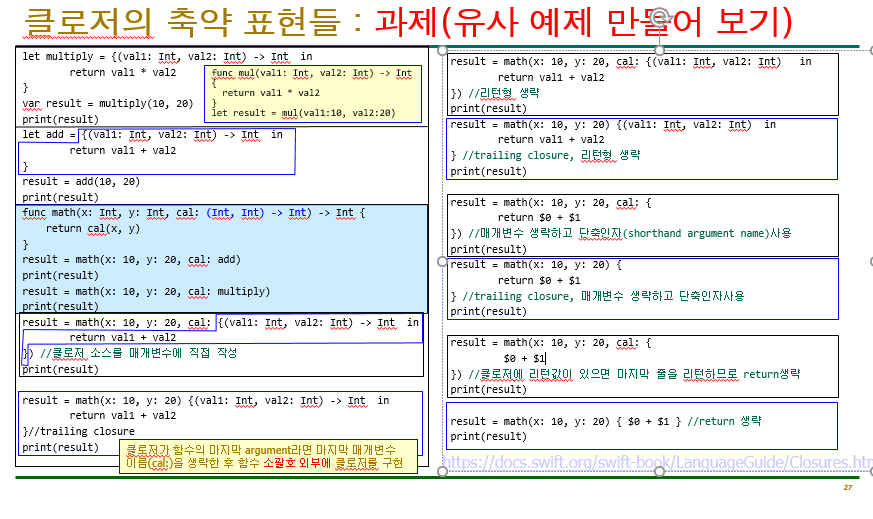
*// 30*

*// 30*

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

*// print(type(of:add1))*

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



*//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)

}

**var** 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)

*//리턴형 (반환 타입) 생략*

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

**return** val1 + val2

})

print(result)

*//trailing closure, 리턴형 (반환 타입)생략*

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

**return** val1 + val2

}

print(result)

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

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

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

})

print(result)

*//trailing closure, 매개변수 생략하고 단축인자사용 단축인자 시험*

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

**return** $0 + $1

}

print(result)

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

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

$0 + $1

})

print(result)

*//trailing closure //return 생략*

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

print(result)

*// 결과 모두 30*

**텍스트이(가) 표시된 사진

자동 생성된 설명**

*//stored property 초기값 주는 방법은 3가지가 있음*

**class** **Man1**{

**var** age : Int = 1 *//stored property는 초기값이 있어야 함*

**var** weight : Double = 3.5 *// 값 직접 주는 방법 1*

}

*// 닐 즉 옵셔널로 줘도 됨 2*

**class** **Man2**{

**var** age : Int? *//stored property는 초기값이 있어야 함, nil*

**var** weight : Double!

}

**3번째 init 을 이용한 초기화 방법 (원래 우리가 사용하고 있지만 숨겨져서 안보였던 것 우리가 정의 해서 초기화 시켜줄 수 있음)**

*// 인스턴스 만들고 메서드와 프로퍼티 접근*

*// var 인스턴스명 = 클래스명()*

*// 클래스명 다음의 괄호는 눈에 보이지 않는 default initializer를 나타냄*

*// var Kim : Man = Man() //:Man은 생략 가능*

**class** **Man**{

**var** age : Int = 1

**var** weight : Double = 3.5

**func** display(){

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

}

*//init() 보이지 않고 숨겨져 있음*

}

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

kim.display() *//인스턴스 메서드는 인스턴스가 호출*

print(kim.age)

*// 나이=1, 몸무게=3.5*

*// 1*

*생성자 정의해보자*

**class** **Man**{

**var** age : Int

*// = 1 생성자 있어서 생략 가능 프로퍼티 초기화 3가지 조건 중 하나만 만족 하면 됨*

**var** weight : Double *// = 3.5*

**func** display(){

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

}

**init**(yourAge: Int, yourWeight : Double){

age = yourAge //self.age = yourAge 현재 self 생략가능 이름이 다르기 때문에

weight = yourWeight

} *//designated initializer*

}

*//var kim : Man = Man() //오류*

*//init()을 하나라도 직접 만들면 default initializer는 사라짐*

**var** kim : Man = Man(yourAge:10, yourWeight:20.5) *// 자동으로 initializer 호출 됨*

kim.display()

*// 나이=10, 몸무게=20.5*

*매개변수 프로퍼티 이름이 같을 때*

*// 현재 클래스 내 메서드나 프로퍼티를 가리킬 때 메서드나 프로퍼티 앞에 self.을 붙임*

*// 다른 언어의 this와 같음(다른 포인터를 사용하지 않는 언어 ) 포인터 관련해서 this 키워드가 정의 되어 있기 때문에 다른 언어와 차이를 둠*

**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*

**타입 메서드 만들기**

**class** **Man** {

**var** age : Int = 1

**var** weight : Double = 3.5

*// 인스턴스 메서드*

**func** display(){

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

}

*// 타입 메서드*

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

print("cM은 클래스 메서드입니다.") *// 클래스 메서드는 안에 프로퍼티 age 같은거 가지고 못 놈*

}

**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*

*// class키워드로 만든 클래스 메서드는 자식 클래스에서 override가능 함*

*// 나이=1, 몸무게=3.5*

*// cM은 클래스 메서드입니다.*

*// scM은 클래스 메서드(static)*