

과 목 명 : iOS프로그래밍실무

학 과 : 컴퓨터소프트웨어

학 번 : 201822048

     반 : A

이 름 : 황재하

제 출 일 : 2023.04.03

담당교수 : 한성현 교수님

실습.1

// 부모 클래스 Human 생성

class Human {

// 프로퍼티 생성

var gender : String

var age : Int

// 메소드 생성

func showInfo(){

print("성별 : \(gender), 나이 : \(age)")

}

// designated initializer 생성

init(gender: String, age: Int) {

// self가 앞에 붙으면 클래스의 프로퍼티라는 뜻임

self.gender = gender

self.age = age

}

}

// 자식 클래스 Man 생성

class Man: Human { // Human 상속받음

// 프로퍼티 생성

var name: String

var weight: Double

// 부모의 메소드를 상속받아 재정의해 사용하고 이름이 똑같으며 자식클래스 메소드가 우선순위가 높음

override func showInfo() {

print("이름 : \(name), 성별 : \(gender), 나이 : \(age), 몸무게 : \(weight)")

}

// 그럼 오버로딩이란?

// 함수의 이름은 같으나 매개변수, 리턴형이 다른 함수름 여려개 정의 하는 것!

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

self.name = name

self.weight = weight

super.init(gender: gender, age: age) // 이 줄을 안쓰면?

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

// 서브 클래스에서 Designated Initializers를 작성할 경우, 상위 클래스의 생성자를 호출해야 한다.

// 서브 클래스에서 Designated Initializers를 작성할 경우 부모의 init()이 상속되지 않기 때문이다.

// 순서 또한 중요하다 서브 클래스의 프로퍼티를 초기화해 준 이후, 상위 클래스의 생성자를 호출해 초기화해야 한다.

}

}

var hwang: Man = Man(gender: "남자", age: 26, name: "재하", weight: 83.4)

hwang.showInfo() // 이름 : 재하, 성별 : 남자, 나이 : 26, 몸무게 : 83.4

실습.2

class Woman{

var age : Int

var weight : Double

func display(){

print("나이=\(age), 몸무게=\(weight)")

}

// init()을 overloading해 사용

// 매개변수에 따라 맞는 자료형의 init()이 자동으로 호출된다.

init(){ // 1, designated initializer

age = 25

weight = 59.5

}

init(age: Int, weight : Double){ // 2, designated initializer

self.age = age

self.weight = weight

} //designated initializer

}

var cho1: Woman = Woman()

cho1.display() // 나이=25, 몸무게=59.5

var cho2: Woman = Woman(age: 26, weight: 56.2)

cho2.display() // 나이=26, 몸무게=56.2

실습.3

class Male{

var weight: Double

var height: Int

func show(){

print("몸무게 : \(weight), 키 : \(height)")

}

init(){

self.weight = 0.0

self.height = 0

}

// designated initializer

init(weight: Double, height: Int){

self.weight = weight

self.height = height

}

// convenience initializer 1

// 일부 프로퍼티만 초기화 할 수 있고 여러개 만들 수 있습니다.

convenience init(weight: Double){ // 매개변수 1개

// 다른 생성자를 이용해 초기화하기 때문에 initializer delegation을 먼저해주고 자신의 프로퍼티를 초기화해야함!

self.init() // initializer delegation

// 자신의 프로퍼티 초기화

self.weight = weight

}

// convenience initializer 2

convenience init(height: Int){

self.init(weight: 83.5, height: height)

}

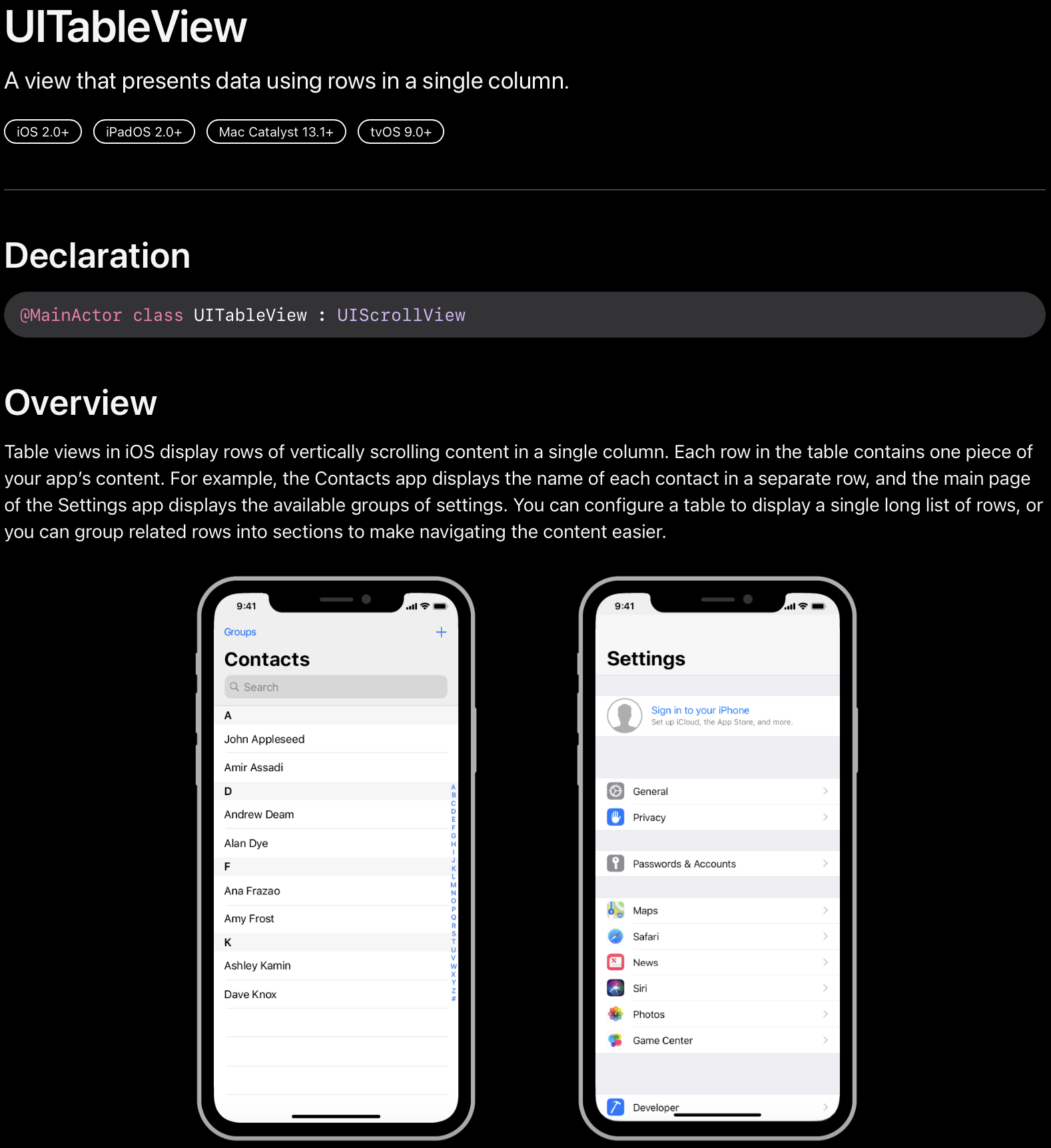
}

var jaeha1: Male = Male(weight: 82.4)

jaeha1.show() // 몸무게 : 82.4, 키 : 0

var jaeha2: Male = Male(height: 178)

jaeha2.show() // 무게 : 83.5, 키 : 178

실습.4 UITableView 간단 도움말

**공식 문서에는 "단일 열에 배열된 행을 사용하여 데이터를 표시하는 뷰" 라고 되어있다.**

**또한 UIScrollView 에 상속받고 있으므로 Scroll이 당연히 가능하고 Scroll 관련된 메소드들도 함께 사용할 수 있다.**

**pasted-image.tiff**

**기본적으로 필요한 데이터소스 메소드는 numberOfSections 와 numberOfRowsInSection 가 있습니다.**

**numberOfSections 은 섹션의 개수를 numberOfRowsInSection 은 섹션안의 행의 갯수를 정하는 메소드 입니다.**

pasted-image.tiff

**행 하나하나는 TableViewCell로 이루어져있고. 이것은 cellForRowAt 이라는 메소드로 행을 구성할 테이블뷰 셀을 정해주면 된다**