

과 목 명 : iOS프로그래밍실무

학 과 : 컴퓨터소프트웨어

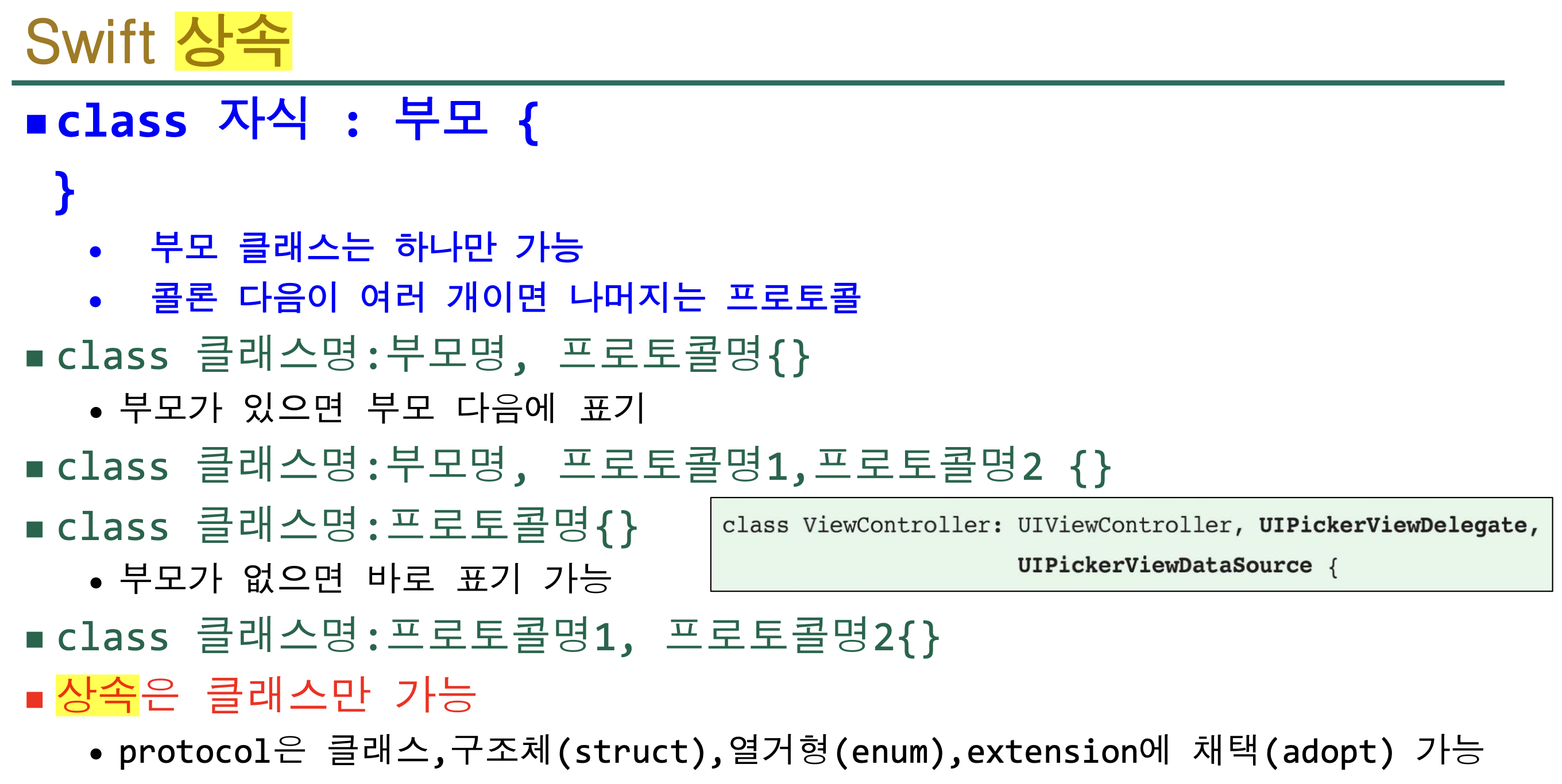
학 번 : 201822048

     반 : A

이 름 : 황재하

제 출 일 : 2023.04.05

담당교수 : 한성현 교수님

실습.1

실습.2

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 {}

//비어있지만 Man의 모든 것을 가지고 있음 }

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

실습.3

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) //이 줄을 안쓰면? => 상위 클래스의 초기화 메서드(init)를 호출해야 하위 클래스의 초기화가 완료되기 때문에, 초기화 메서드에서 super.init을 호출하는 것이 필요합니다.

}

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

} // 자식 클래스에서 designated initializer를 만들면 부모 init()상속 안됨!!

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

lee.displayS()

lee.display()

//이름=홍길동, 나이=20, 몸무게=65.2

//나이=20, 몸무게=65.2

실습.4

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() { //같은 이름의 메서드가 부모에도 있음

// override란?

// 최우선의 뜻으로 부모에 있는 같은 이름의 메소드보다 자식에 있는 메소드를 우선으로 호출된다.

// 부모클래스에 있는 메소드를 자식클래스에서 이름을 같게하여 기능을 재정의해 사용하는 것

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

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

self.name = name

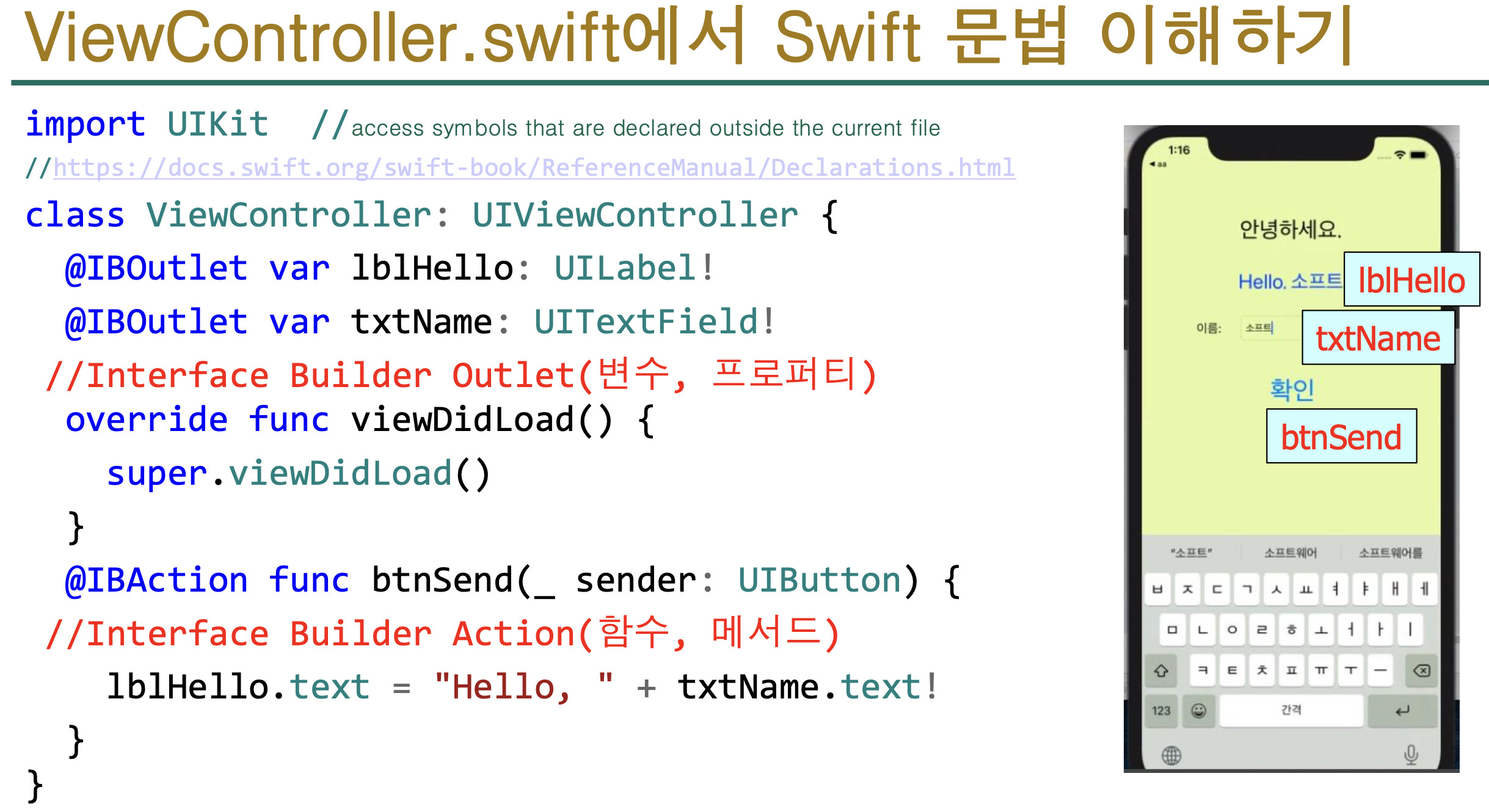
super.init(age:age, weight:weight)

}

}

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

lee.display() //이름=홍길동, 나이=20, 몸무게=65.2

실습.5

실습.6

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

}

}

실습.7

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

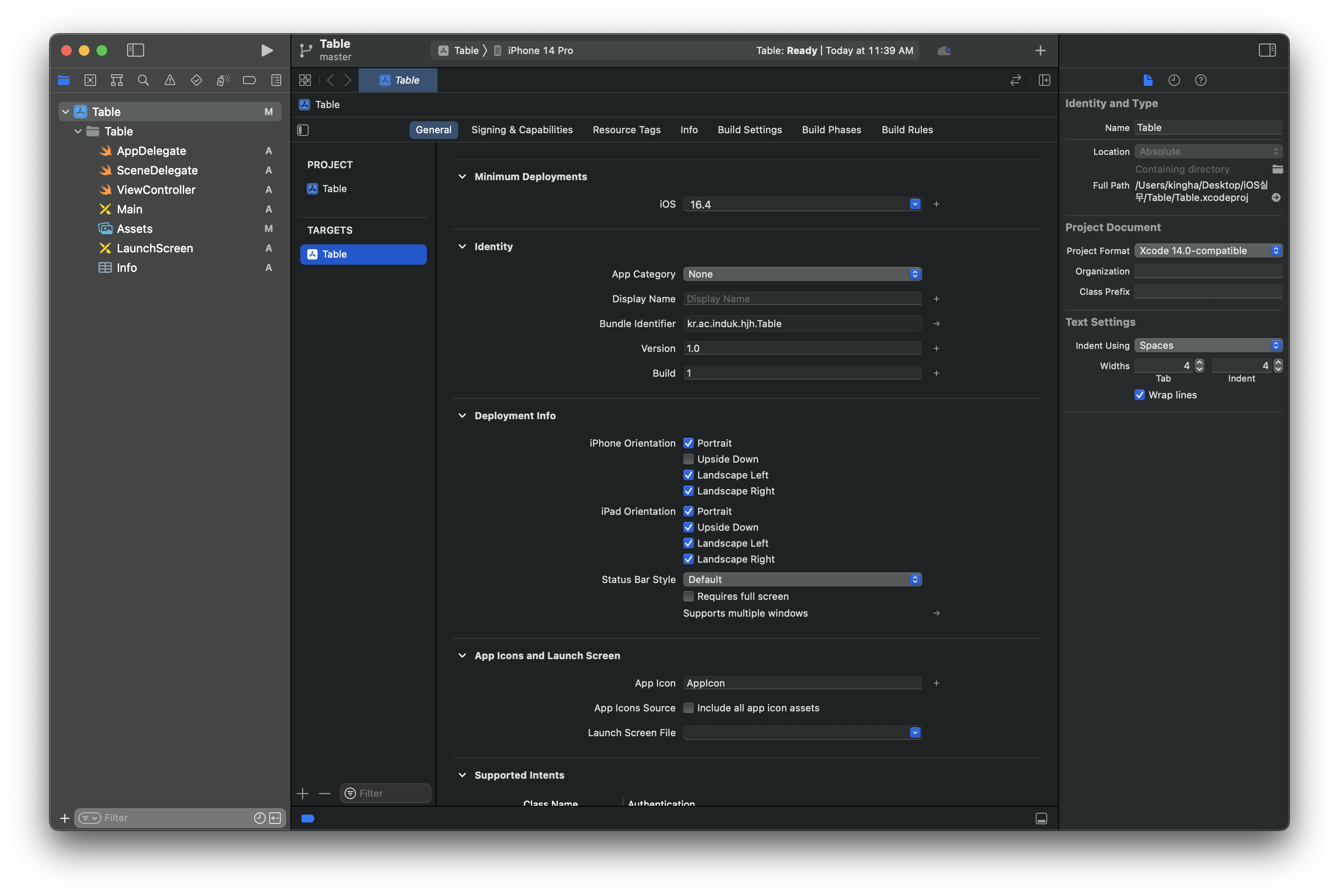
var x : Int = 1 //준수, conform

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

// 준수, conform 가 안되면? => 프로토콜을 채택한 후에 해당 프로토콜의 요구사항을 준수하지 않는다면, 컴파일 오류가 발생합니다. 즉, 프로토콜의 요구사항을 충족시키지 않은 타입은 컴파일이 되지 않습니다.

}

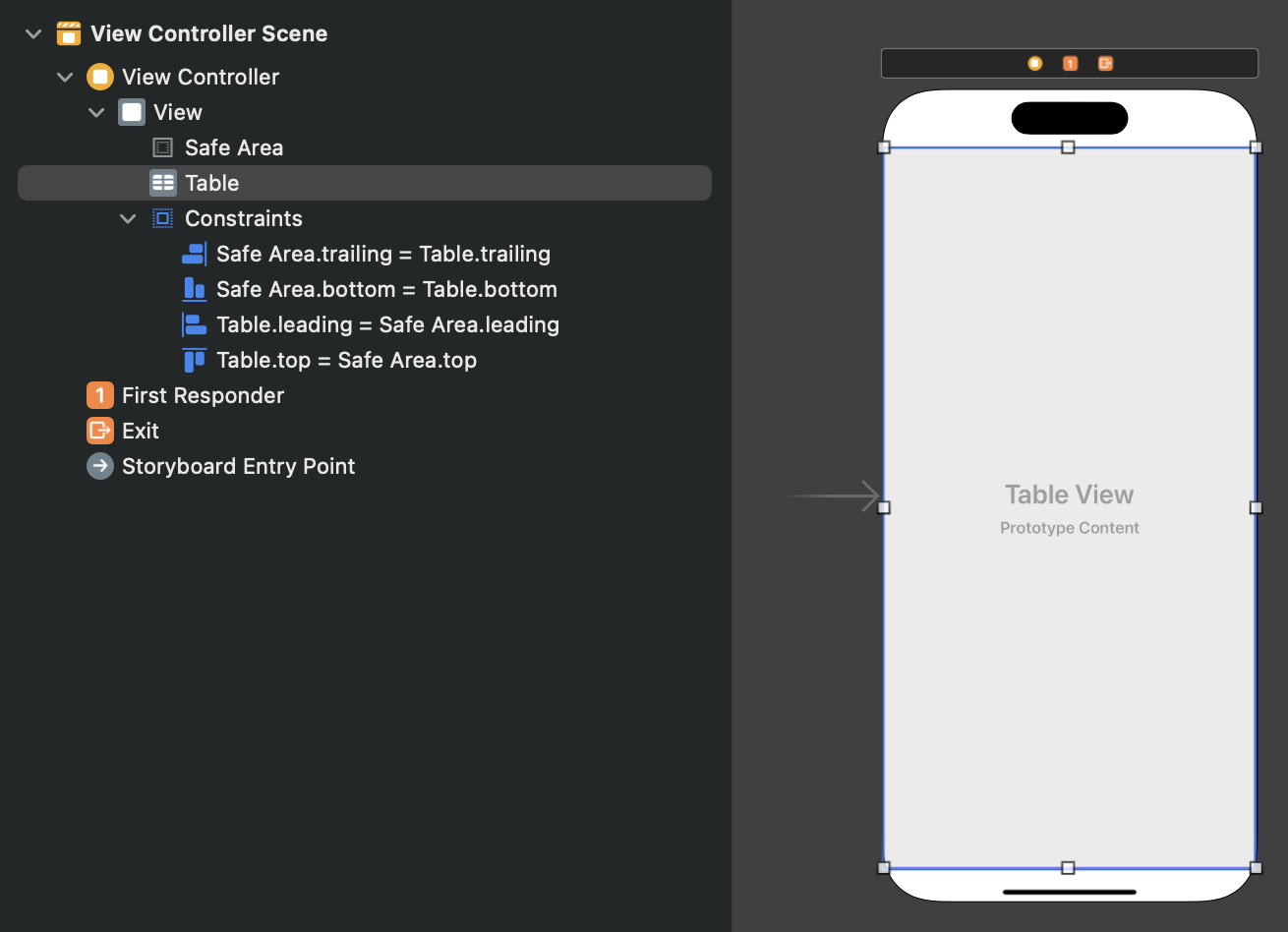
}

실습.8

중요!

Minimun Deployments : 앱이 돌아가기 위한 최소 요구 버전

Bundle identifier : 고유하게 만들어야 앱스토어에 등록 가능

실습.9

실습.10