iOS 프로그래밍(D)

2주차 과제

박필준 교수님

빅데이터트랙

2091096

오현석

1. iOS에서 사용할 수 있는 자료형의 유형 및 값의 범위 정리하기

```
//2091096 오현석
// 상수의 선언
let name : type = value

// 변수의 선언
var name : type = value

// Integer Types

// 64bit 기준: -2^63 ~ 2^63 - 1
let intValue: Int = 42

// 64bit 기준: 0 ~ 2^64 - 1
let uintValue: UInt = 100

// -128 ~ 127
let int8Value: Int8 = -128

// 0 ~ 255
let uint8Value: UInt8 = 255

// -32,768 ~ 32,767
let int16Value: Int16 = 32767

// 0 ~ 65,535
let uint16Value: UInt16 = 65535

// -2,147,483,648 ~ 2,147,483,647
let int32Value: Int32 = 2147483647

// 0 ~ 4,294,967,295
let uint32Value: UInt32 = 4294967295
```

```swift
// Floating-Point Types

// 32bit: 약 -3.4 × 10³⁸ ~ 3.4 × 10³⁸ (6자리 정밀도)
let floatValue: Float = 3.14159

// 64bit: 약 -1.7 × 10³⁰⁸ ~ 1.7 × 10³⁰⁸ (15자리 정밀도)
let doubleValue: Double = 3.14159265359

// Boolean Type
// true 또는 false
let boolValue: Bool = true

// String & Character Types
// 유니코드 문자열 (범위 제한 없음)
let stringValue: String = "Hello, iOS"
// 단일 유니코드 문자
let charValue: Character = "A"

// Collection Types

// 순서 있는 요소 집합
let arrayValue: [Int] = [1, 2, 3]

// 키-값 쌍
let dictValue: [String: Int] = ["age": 25]

// 고유한 요소 집합
let setValue: Set<String> = ["apple", "banana"]
```

2. 옵셔널 변수에 대한 개념과 안전하게 옵셔널 변수를 처리하는 방법

```swift
// 옵셔널 (Optional) 변수의 개념
//Swift의 옵셔널(Optional)은 변수가 값이 있거나 nil(없음)일 수 있는 상태를 나타냄
//값의 부재를 안전하게 처리하기 위한 메커니즘으로, ?를 사용해 선언
var optionalValue: String? = "2091096오현석"
optionalValue = nil //nil 허용

//안전하게 옵셔널 변수 처리하는 방법

//#옵셔널 바인딩 (Optional Binding)
//if let 또는 guard let으로 값이 있는 경우에만 안전하게 사용
if let value = optionalValue {
print(value) // 값이 있을 때만 실행

//#강제 언래핑 (Force Unwrapping)
//!로 강제로 값을 꺼냄. nil이면 오류 발생
print(optionalValue!) // 값이 확실할 때만 사용

//#nil 병합 연산자 (Nil-Coalescing Operator)
//??로 nil일 때 기본값 제공
let result = optionalValue ?? "Default"

//#옵셔널 체이닝 (Optional Chaining)
//?.로 옵셔널 객체의 속성 접근. nil이면 결과도 nil.
let name = person?.dog?.name
```

3. 2주차 이미지 뷰 프로그램 만들기 실습의 버그 수정

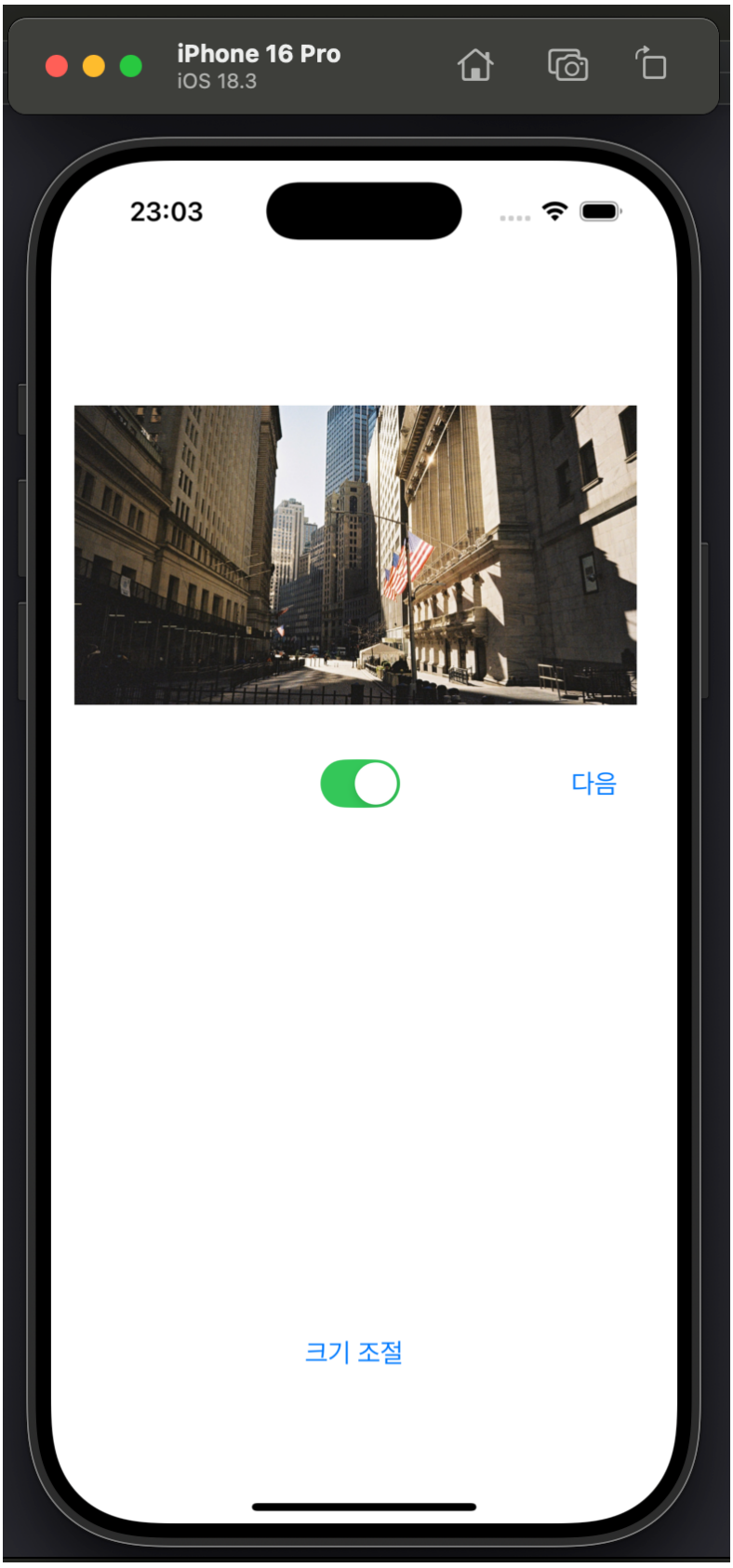
3.1 소스코드

```swift
import UIKit

class ViewController: UIViewController {
    var isZoom = true //true 값으로 수정하여 화면 밖으로 벗어나는 이미지 버그 수정

    //이미지 배열 추가
    var currentImageIndex = 0
    let images = [UIImage(named: "swift.jpeg")!,
                  UIImage(named: "35836.JPG")!,
                  UIImage(named: "950A705A-DDCD-48E9-8B54-8789C505EE4F.jpeg")!,
                  UIImage(named: "726FDE33-9228-45B1-AA8C-5EAF00EB03D0.jpeg")!,
                  UIImage(named: "02493592-2F68-49B5-8840-CA020B36884E.jpeg")!,
                  UIImage(named: "33668.JPG")!,
                  UIImage(named: "3C59A070-DDA7-440E-9B8C-582BE3A7E036.jpeg")!
    ]
    var imgOn: UIImage?
    @IBOutlet weak var imgView: UIImageView!
    @IBOutlet weak var buttonResize: UIButton!
    @IBOutlet weak var buttonShift: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

        //이미지 불러오기 및 초기화
        imgOn = UIImage(named: "swift.jpeg")
        imgView.image = imgOn
    }
```

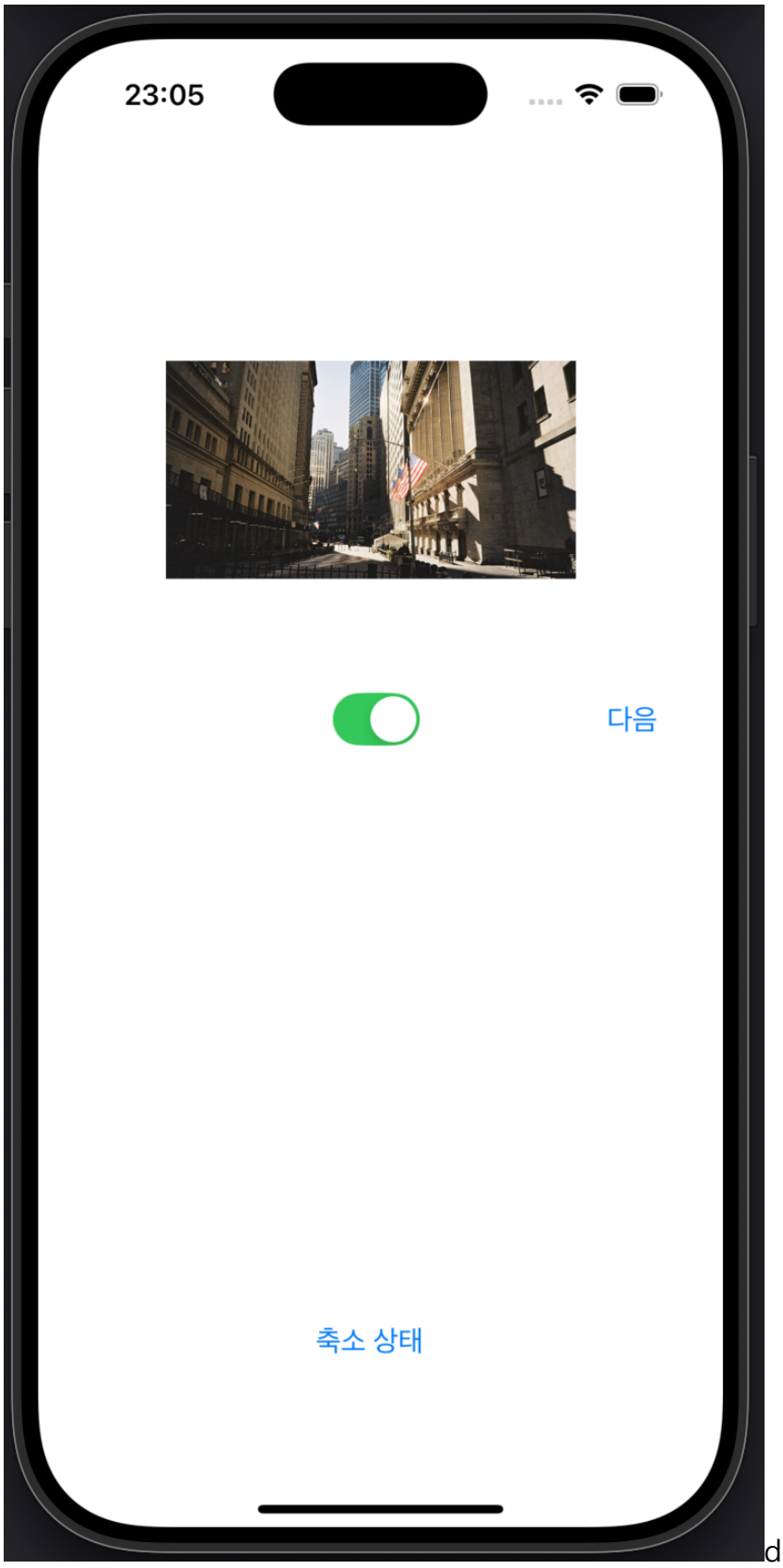
```swift
    @IBAction func buttonResize(_ sender: UIButton) {
        let scale: CGFloat = 1.5
        var newWidth: CGFloat, newHeight: CGFloat

        // 현재 중심점 저장
        let currentCenter = imgView.center

        if(isZoom) {
            newWidth = imgView.frame.width / scale
            newHeight = imgView.frame.height / scale
            buttonResize.setTitle( "축소 상태", for: .normal)
        }//UX를 고려하여 버튼명을 출력된 이미지 상태 표기로 변경

        else {
            newWidth = imgView.frame.width * scale
            newHeight = imgView.frame.height * scale
            buttonResize.setTitle( "확대 상태", for: .normal)
        }//UX를 고려하여 버튼명을 출력된 이미지 상태 표기로 변경

        imgView.frame.size = CGSize(width: newWidth, height: newHeight)
        //이미지 확대 시 정렬을 벗어나는 버그 이미지 중심점을 유지하여 수정
        imgView.center = currentCenter
        isZoom = !isZoom
    }
```

```swift
    //이미지 표기 스위치
    @IBAction func UISwitch(_ sender: UISwitch) {
        if sender.isOn {
            imgView.image = images[currentImageIndex]
        }
        else{
            imgView.image = nil
        }
    }

    //이미지 전환 버튼
    @IBAction func buttonImgSwitch(_ sender: UIButton) {
        currentImageIndex = (currentImageIndex + 1) % images.count
            imgView.image = images[currentImageIndex]
            sender.setTitle("다음", for: .normal)
    }
}
```

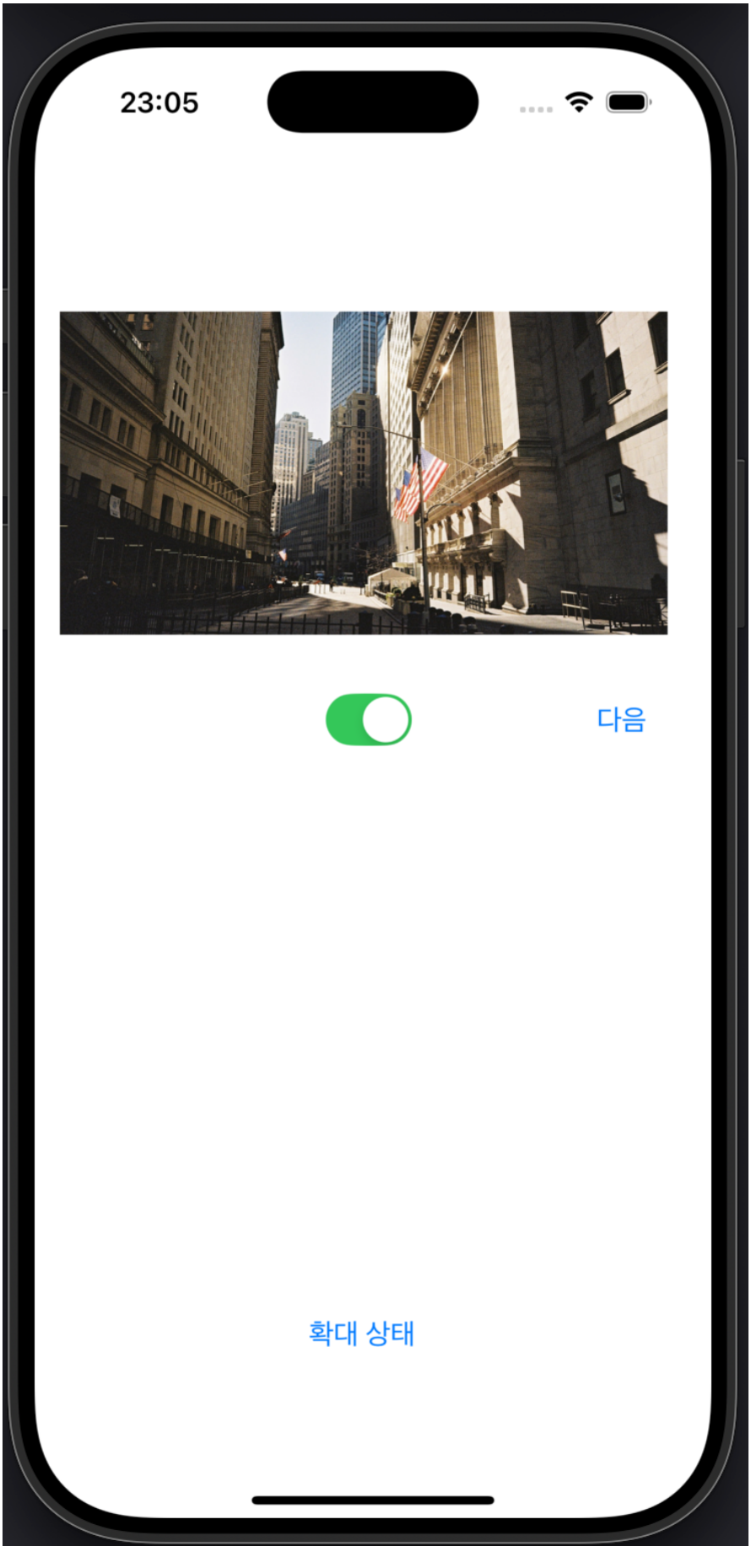
3.2 Simulator 초기 화면



3.3 "크기 조절" 버튼 클릭



d


23:05
다음
확대 상태

3.4 "다음" 버튼 클릭 시 (이미지 전환 기능 추가)