



뷰와 뷰 계층 구조





학습목표

- UINavigationController과 UIView 사이의 관계를 이해한다.
- UINavigationController의 라이브 사이클을 이해하고 활용할 수 있다.
- viewDidLoad() 함수의 호출 시점을 이해하고 이 함수에서 해야하는 작업을 이해한다.
- UIResponder 객체를 이해하고 키보드를 제어할 수 있다.
- Delegation을 이해하고 사용할 수 있다.



뷰 기본 지식

■ 뷰: 사용자에게 보일 수 있는 객체

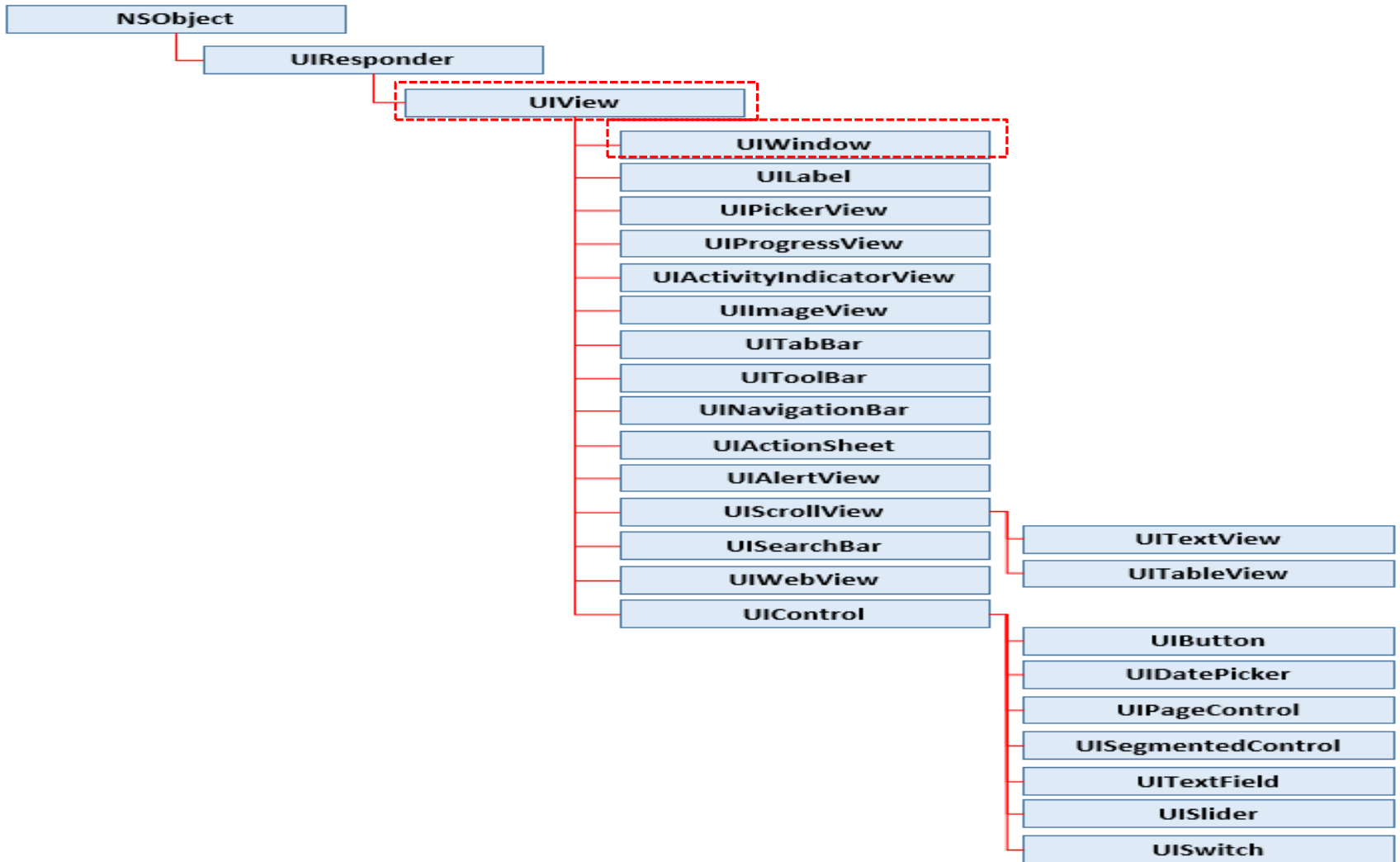
- 뷰는 UIView의 인스턴스이거나 UIView 하위 클래스의 인스턴스이다.
 - 통상 클래스 계층에서 단말 뷰를 컨트롤 또는 위젯이라 함
- 뷰는 자신을 어떻게 그리는지 알고(뷰 자신이) 있다
- 뷰는 터치와 같은 이벤트를 처리할 수 있다.
- 뷰 인스턴스는 뷰 계층 구조상에 존재한다. 뷰 계층 구조의 루트는 앱의 윈도우이다.





뷰 계층 구조

클래스 계층도



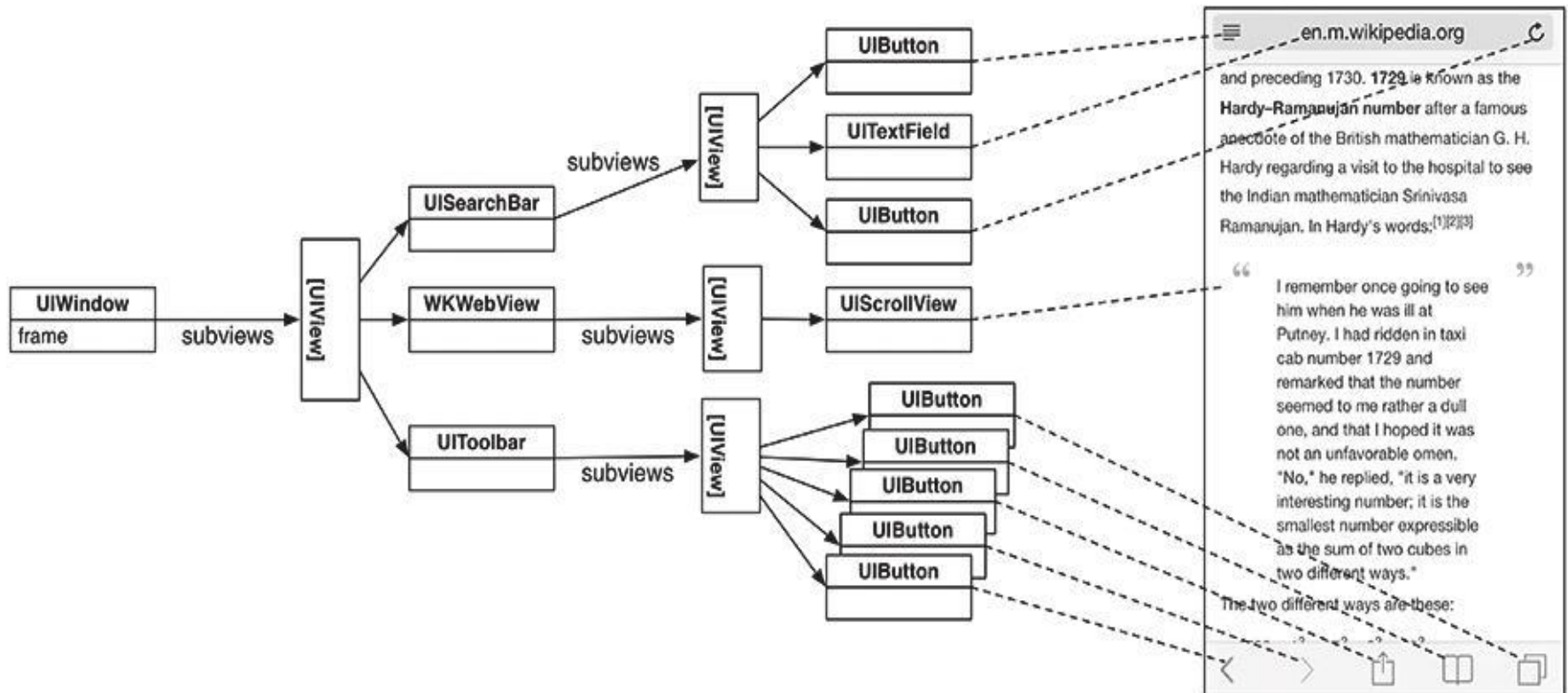


뷰 인스턴스의 계층 구조

■ UIWindow

- 모든 앱은 하나의 UIWindow 인스턴스를 가진다.
- 앱의 여러 뷰들은 이 UIWindow 인스턴스 내 포함된다.

■ 앱내의 view 인스턴스 포함 관계

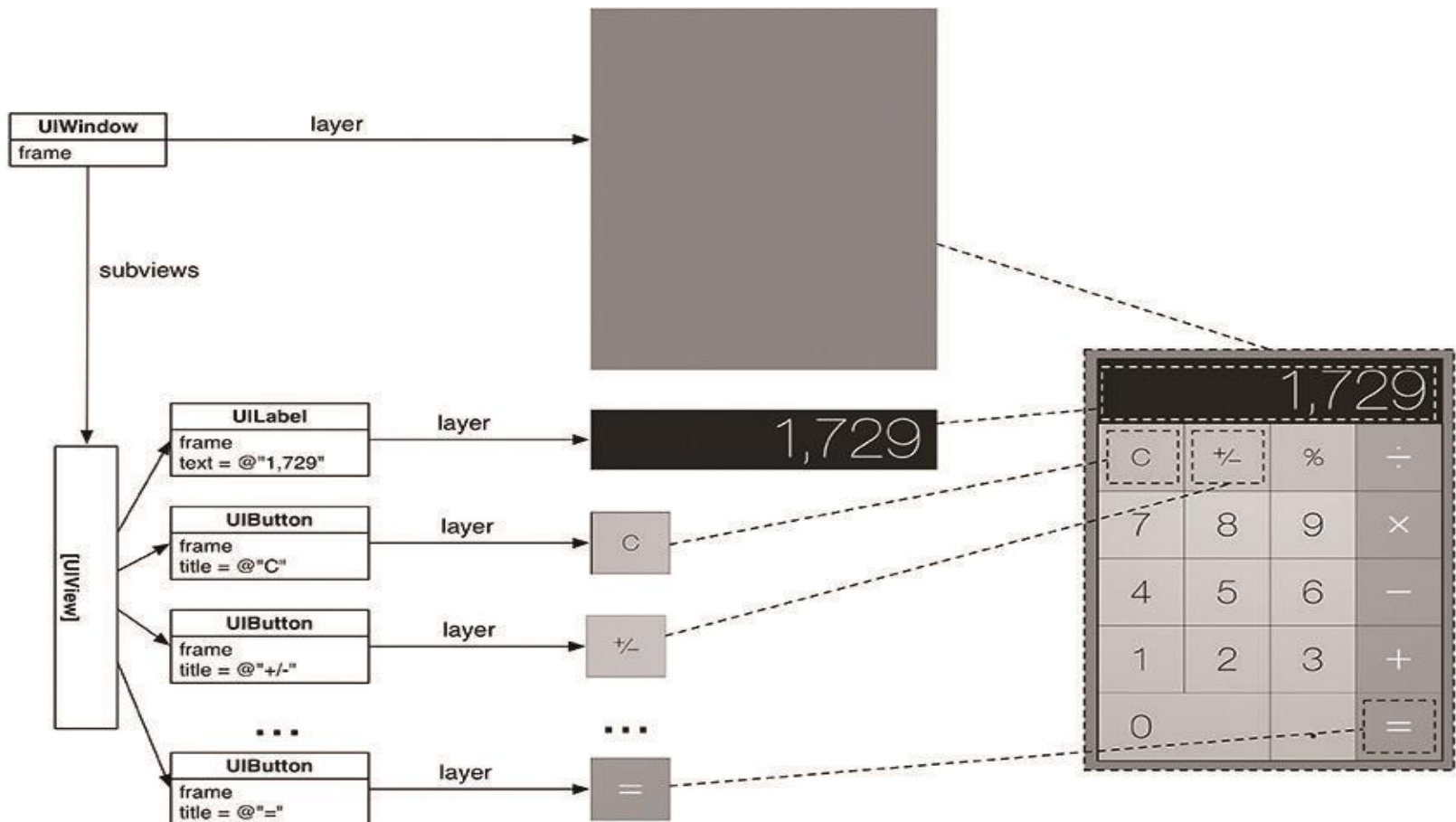




뷰 인스턴스의 계층 구조

■ 화면이 그려지는 과정

- 윈도우를 포함한 각 계층 구조의 뷰는 자신을 레이어(CALayer)에 그린다.
- 모든 뷰의 레이어들은 전체 화면에 합성된다.





새로운 프로젝트 만들기

- file->New->Project
 - Application -> Single View Application
 - Project Name: Conversion

Choose options for your new project:

Product Name:

Team:

Organization Identifier:

Bundle Identifier:

Interface:

Language:

Storage:

☐ Host in CloudKit

☐ Include Tests

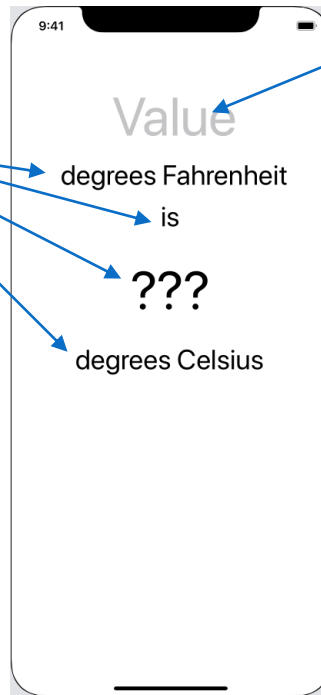


■ Main.storyboard를 이용한 뷰의 계층 구조 만들기

- Main.storyboard를 open하라
- 도큐먼트 아웃라인이나 인터페이스 위에 있는 노랑 원을 클릭하여 View Controller 선택

UILabel
fontSize: 36 or 70/Bold
Alignment: Center

배치는 적절히 중앙에 놓아라



UITextField
fontSize: 70/Bold
Alignment: Center
Placeholder: Value
Text Input Traits - Keyboard: Decimal Pad
Border Style: dotted line(첫번째)

적절히 높이를 변경할 수 있도록 해준다



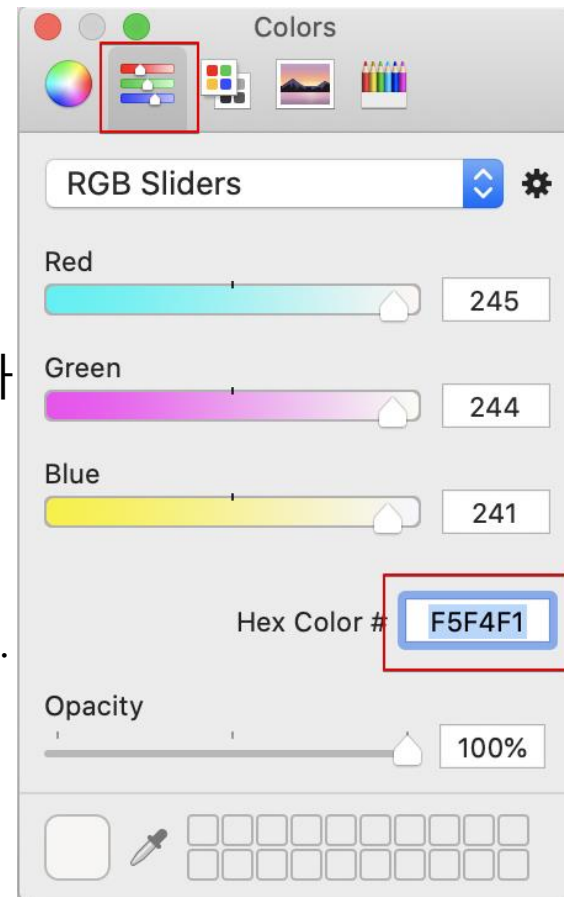
UI Design

■ 배경색 변경

- Main.storyboard에서 뷰(Document Outline에서 View)를 선택한다
- 속성 인스펙터(Attribute inspector)를 열고 새로운 배경색을 선택한다.
 - Background를 드롭 다운 하여 **custom**를 선택한다.
 - 두 번째 탭을 선택하고 **RGB Sliders**를 선택한다
 - Hex Color 값으로 **F5F4F1**을 입력하라.
 - 따뜻한 회색으로 바뀔 것이다.

■ Label 색상 변경

- "degrees ..", "is", "degree.." Label들을 동시에 선택하라
- 속성 inspector에서 Color를 선택하라
- 위와 동일한 방법으로
 - Hex Color값으로 **E15829** 값을 입력
 - 3가지 Label의 글자 색이 약간 붉은 글씨로 보일 것이다.



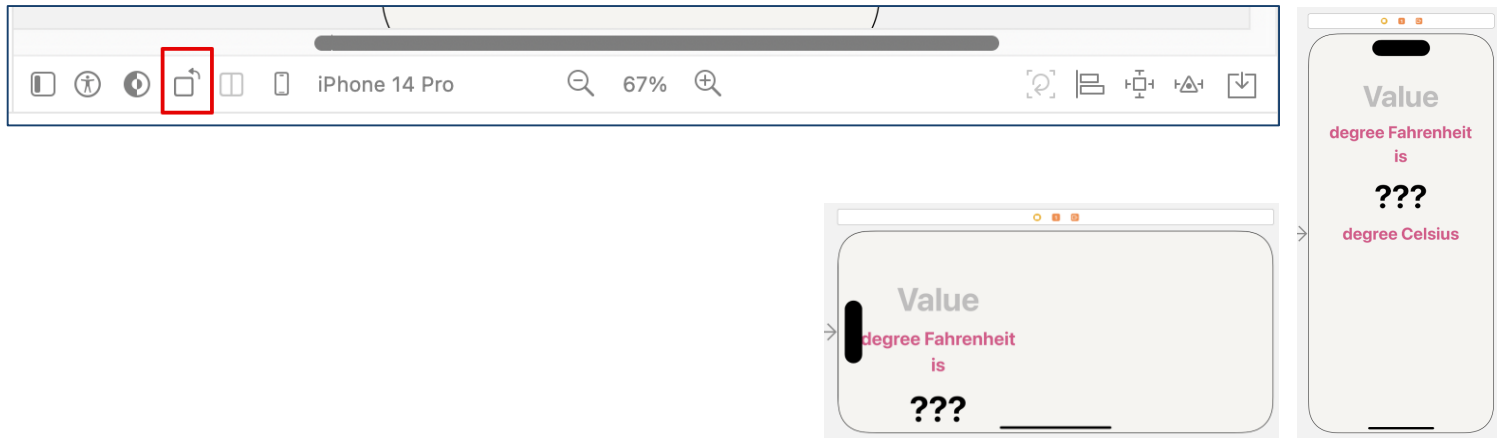


UI Design

■ 테스트1

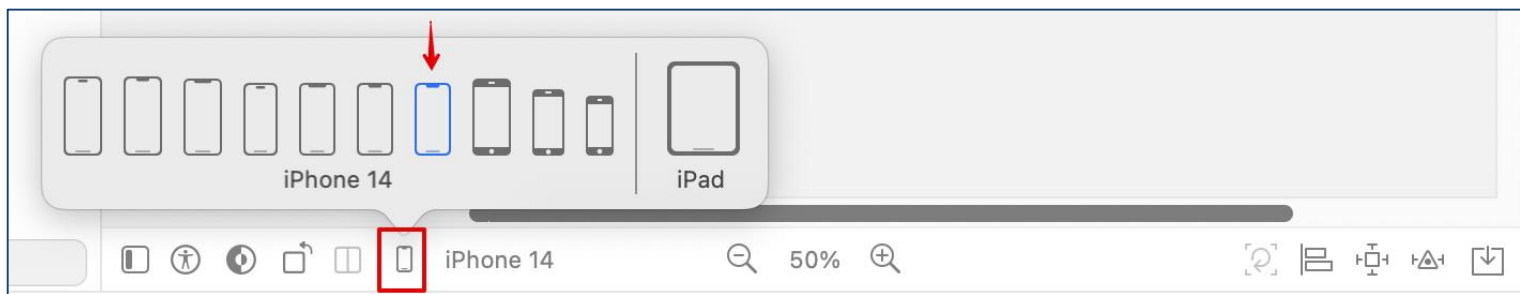
– 스토리보드에서

- Editor 하단에서 아래 버튼을 클릭으로 landscape, portrait 화면보기



- 다양한 폰으로 변경하기

– 아래 그림과 같이 메뉴에서 폰을 선택하고 좌우로 원하는 폰을 선택





UI Design

■ 테스트2

– 시뮬레이터 또는 실제 폰으로

ch03-JaeMoonLee-C...
main

ch03-Ja...Convert > iPhone 14 Pro Att (1) 클릭한다 Lee-

(3) 클릭하여 실행한다

(4) 클릭하여 중지한다

Filter

Mac

My Mac (Designed for iPad)

Build

Any iOS Device (arm64)

Any iOS Simulator Device (arm64, x86_64)

iOS Simulators

iPad (10th generation)

iPad Air (5th generation)

iPad Pro (11-inch) (4th generation)

iPad Pro (12.9-inch) (6th generation)

iPad mini (6th generation)

iPhone 14 Plus

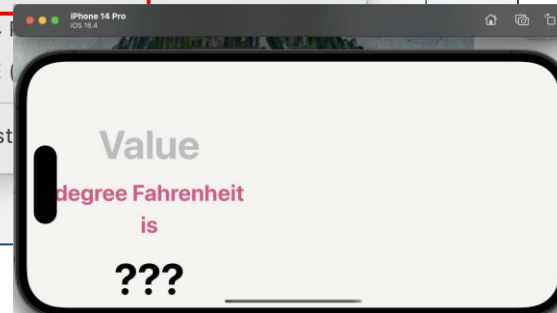
✓ iPhone 14 Pro

iPhone 14

iPhone SE

Manage Run Dest

클릭하면 회전한다



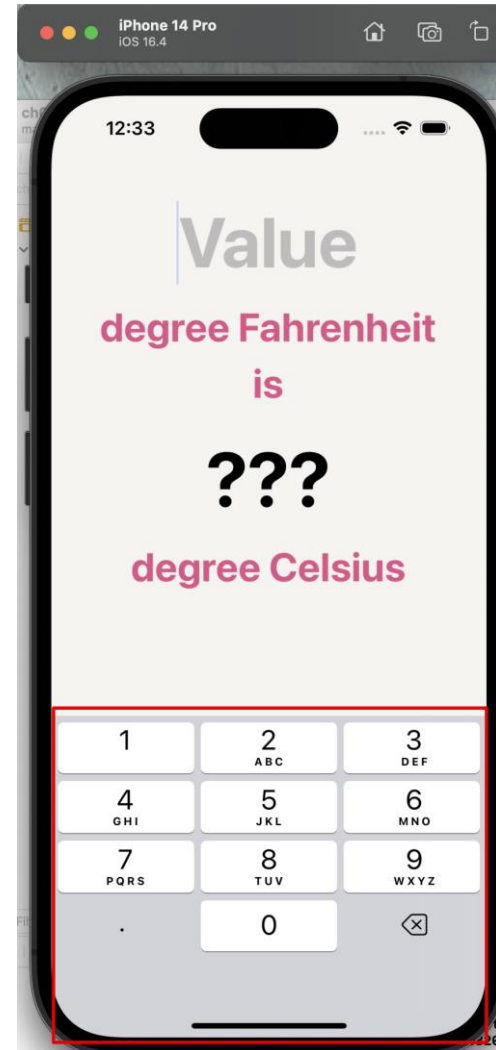


UI Design

■ 테스트2

– 키보드 입력하기

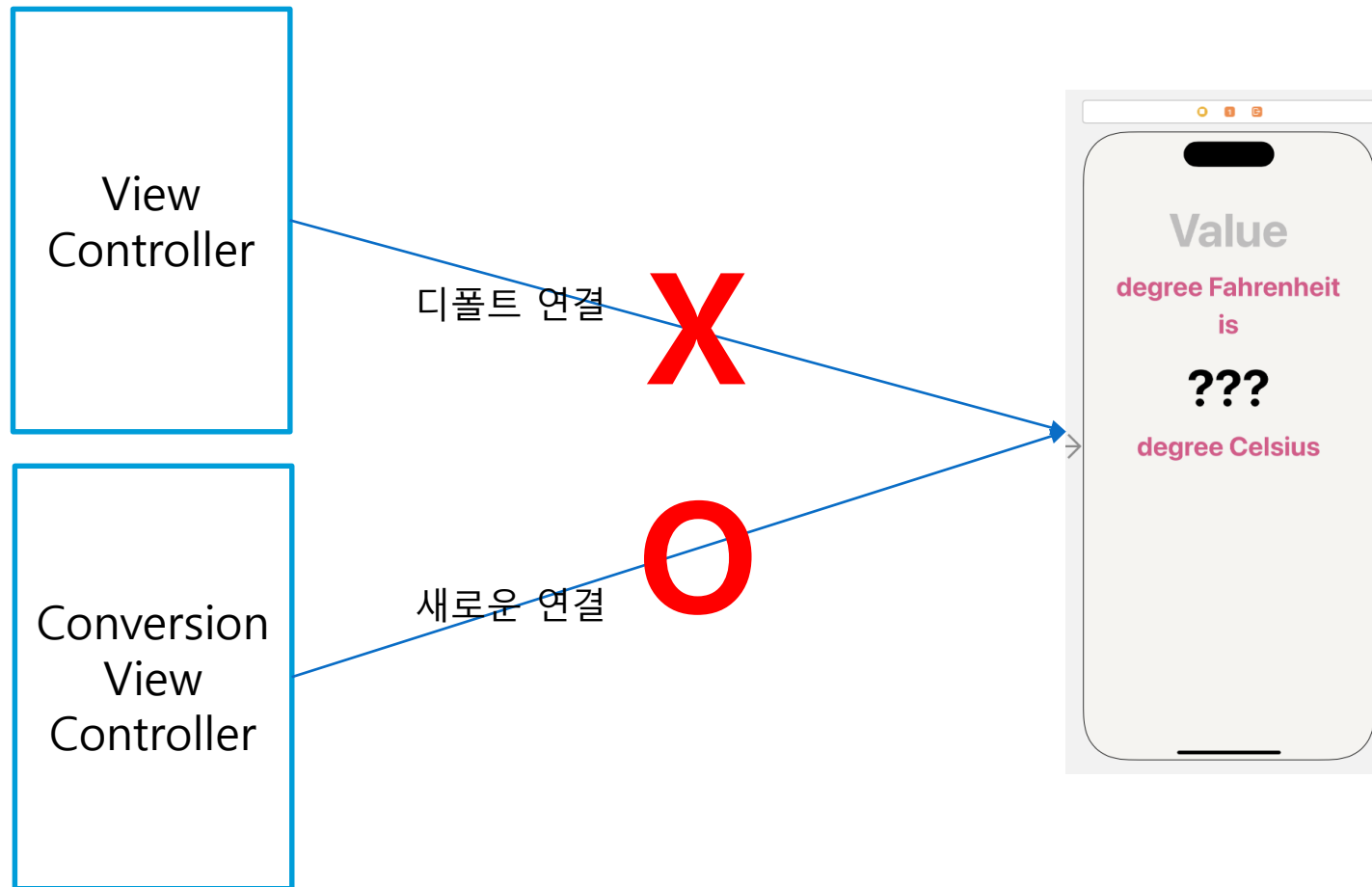
- 에뮬레이터를 선택한다
- TextField(Value)를 클릭한다.
- 메뉴에서 I/O -> Keyboard -> Toggle Software Keyboard 선택





ViewController 교체

- 새로운 ViewController로 교체





새로운 UINavigationController

■ UINavigationController 생성

- File->New->File->Cocoa Touch Class
- UINavigationController.swift 생성

Class: ConversionViewController

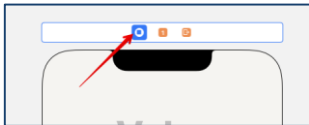
Subclass of: UIViewController

☐ Also create XIB file

Language: Swift

■ Main.storyboard의 ViewController과 UINavigationController의 연결

- 스토리보드에서 ViewController 선택

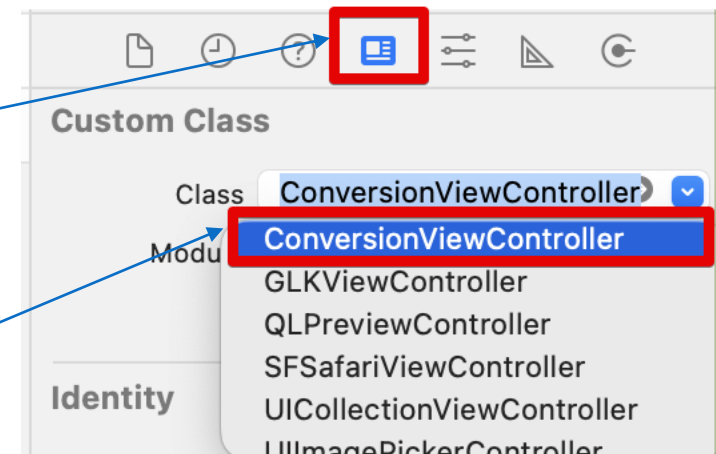


- Identity Inspector 선택(4번째)

- Class에서

- UINavigationController 선택

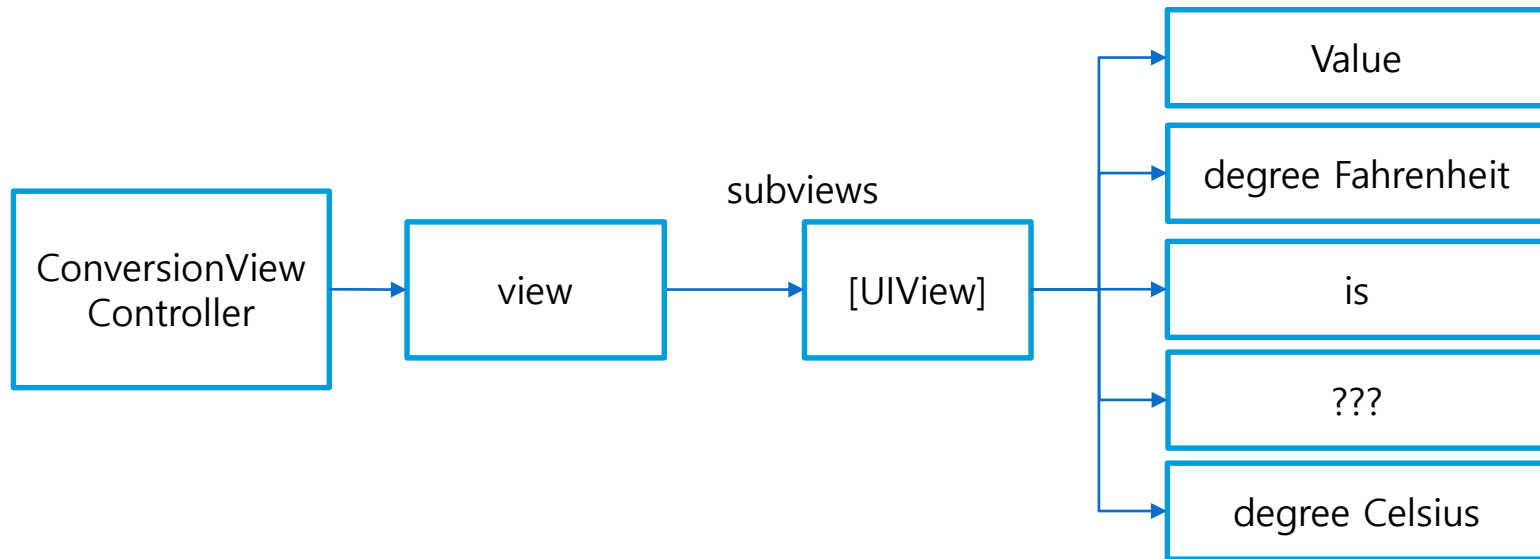
이제 Navigator에 있는
ViewController.swift는 삭제하여도 된다.





View Hierarchy

■ View & ViewController

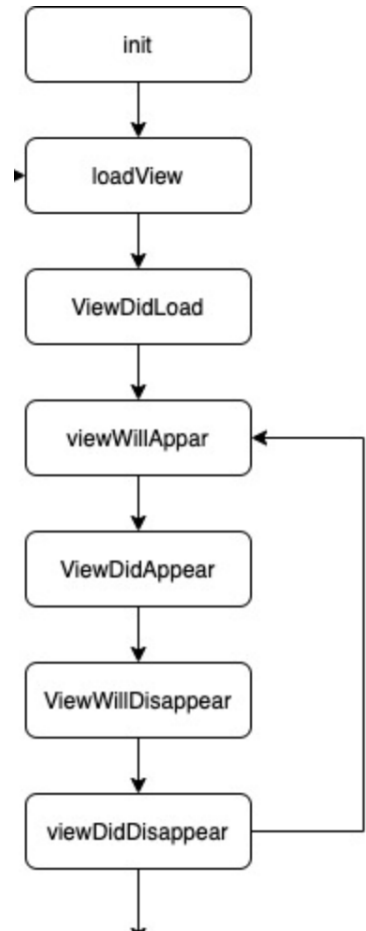




View Hierarchy

■ UIViewController Lifecycle

- init: 생성자
- loadView: 스토리보드 등을 이용하여 View Hierarchy를 완성, 대체로 코딩할 내용은 없음
- **viewDidLoad**: View가 화면에 나오기전에 초기화 등을 여기서 함, 여기서 많은 코딩이 일어남
- viewWillAppear: 화면에 보이기 직전에 호출되는 함수로 여기서 UI layout등을 조정
- viewDidAppear: 화면이 보이는 상태임, 여기서부터 사용자와 인터렉션이 가능
- viewWillDisappear: 화면이 보이지 않게 되는 시점에서 호출, 특별한 코딩은 없음
- **viewDidDisappear**: 화면이 사라진후에 호출됨, 통상적으로 저장되어야 하는 데이터가 있다면 저장

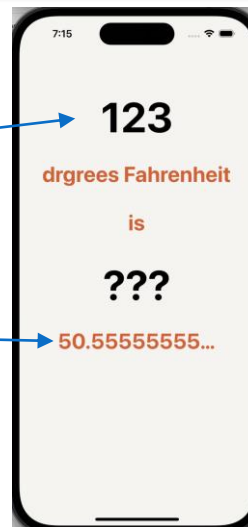
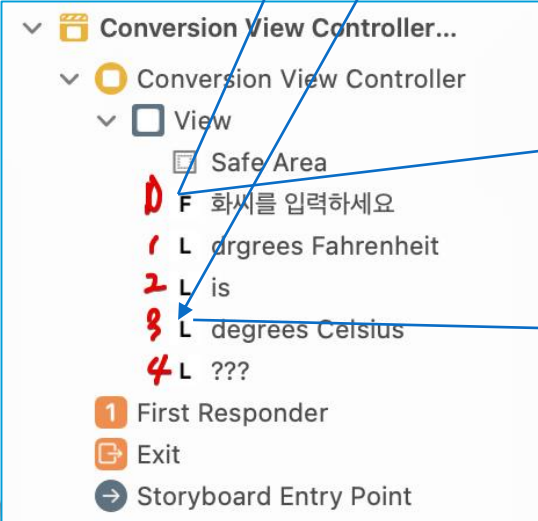
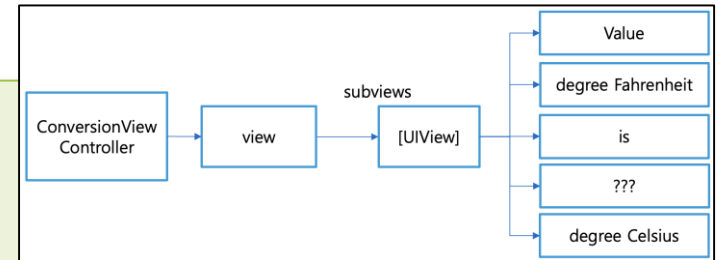




UIViewController에서 View Access

viewDidLoad() 함수에서 코딩 & 테스트

```
class ConversionViewController: UIViewController {  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        // Do any additional setup after loading the view.  
        if view.subviews[0] is UITextField{  
            (view.subviews[0] as? UITextField)?.text = "123"  
        }  
  
        let celsiusValue = 5.0 / 9.0 * (123.0 - 32)  
  
        if view.subviews[3] is UILabel{  
            (view.subviews[3] as! UILabel).text = String(celsiusValue)  
        }  
    }  
}
```



??? 대신에
50.555가 나와야 하는데
잘못됨
무엇을 고쳐야 하나?



UIViewController에서 View Access

■ 앞의 코드를 간략하게

- Optional Chain을 사용하여 간단하게 해보자
 - xxx?.yyy?.zzz?에서 하나라도 nil이면 전체가 nil임

```
class ConversionViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        // if view.subviews[0] is UITextField{
        //     (view.subviews[0] as? UITextField)?.text = "123"
        // }
        // 어차피 view.subviews[0]가 UITextField이 아니라면 "123"은 나타나지 않는다.
        (view.subviews[0] as? UITextField)?.text = "123"

        let celsiusValue = 5.0 / 9.0 * (123.0 - 32)

        // if view.subviews[3] is UILabel{
        //     (view.subviews[3] as! UILabel).text = String(celsiusValue)
        // }
        // 어차피 view.subviews[4]가 UILabel이 아니라면 "123"은 나타나지 않는다.
        (view.subviews[4] as? UILabel)?.text = String(celsiusValue)
    }
}
```



UIViewController에서 View Access

■ 앞의 코드를 더 간략하게

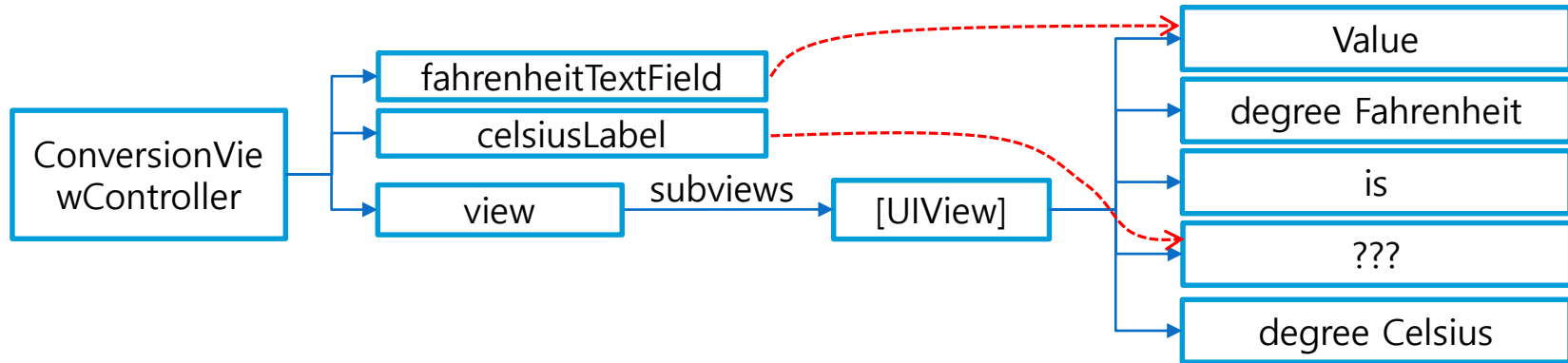
- Implicit Unrapping을 사용하여 ?를 최소화
 - 명확한 것은 ?보다 !를 사용

```
class ConversionViewController: UIViewController {  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        // Do any additional setup after loading the view.  
  
        // (view.subviews[0] as? UITextField)?.text = "123"  
        // view.subviews[0]가 명확히 UITextField이기 때문에 as!를 사용  
        // 따라서 두번째 ?를 제거할 수 있다.  
        (view.subviews[0] as! UITextField).text = "123"  
  
        let celsiusValue = 5.0 / 9.0 * (123.0 - 32)  
  
        // (view.subviews[4] as? UILabel)?.text = String(celsiusValue)  
        // view.subviews[4]가 명확히 UILabel이기 때문에 as!를 사용  
        // 따라서 두번째 ?를 제거할 수 있다.  
        (view.subviews[4] as! UILabel).text = String(celsiusValue)  
    }  
}
```



UIViewController에서 View Access

- 지금과 같은 방법으로 너무 불편, 다음과 같은 방법으로 개선



```
class ConversionViewController: UIViewController {
    var fahrenheitTextField: UITextField!
    var celsiusLabel: UILabel?           // !와 ?의 차이가 무엇인지 아래에서 파악하라

    override func viewDidLoad() {
        super.viewDidLoad()

        fahrenheitTextField = view.subviews[0] as! UITextField
        celsiusLabel = view.subviews[4] as? UILabel

        //(view.subviews[0] as! UITextField).text = "123"
        fahrenheitTextField.text = "123"

        let celsiusValue = 5.0 / 9.0 * (123.0 - 32)
        //(view.subviews[4] as! UILabel).text = String(celsiusValue)
        celsiusLabel?.text = String(celsiusValue)
    }
}
```



UIViewController에서 View Access

■ 앞 방법의 문제점

- Document Outline에서 뷰의 순서를 바꾸면 ViewController에서 배열의 인덱스를 항상 재설정하여야 함
- 안드로이드처럼(findViewById()) 일일이 별도의 프로그래밍을 하여야 함


■ xcode에서 다소 편리한 기능제공

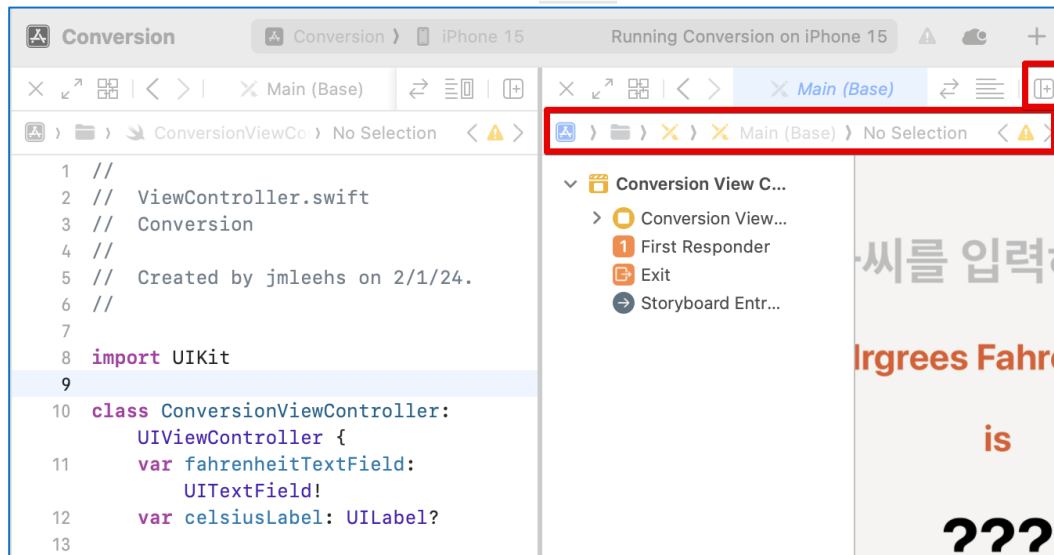
- IBOutlet
 - 스토리보드의 UIView객체에 대하여 자동으로 ViewController의 변수와 연결, Document Outline에서 위치가 바뀌어도 문제 없음
- IBAction
 - 스토리보드의 UIView객체의 이벤트 핸들러 함수를 자동으로 ViewController의 함수와 연결, Document Outline에서 위치가 바뀌어도 문제 없음



UIViewController에서 View Access

■ xcode의 editor에 여러 문서 동시 open

- main.storyboard와 ConversionViewController동시 오픈
- 방법1
 - main.storyboard(ConversionViewController)또는 가 오픈된 상태에서
 - **옵션키**를 누른상태에서 ConversionViewController.swift(또는 main.storyboard)를 클릭하라.
- 방법2
 - Editor의 오른쪽 상단의  을 클릭하라.



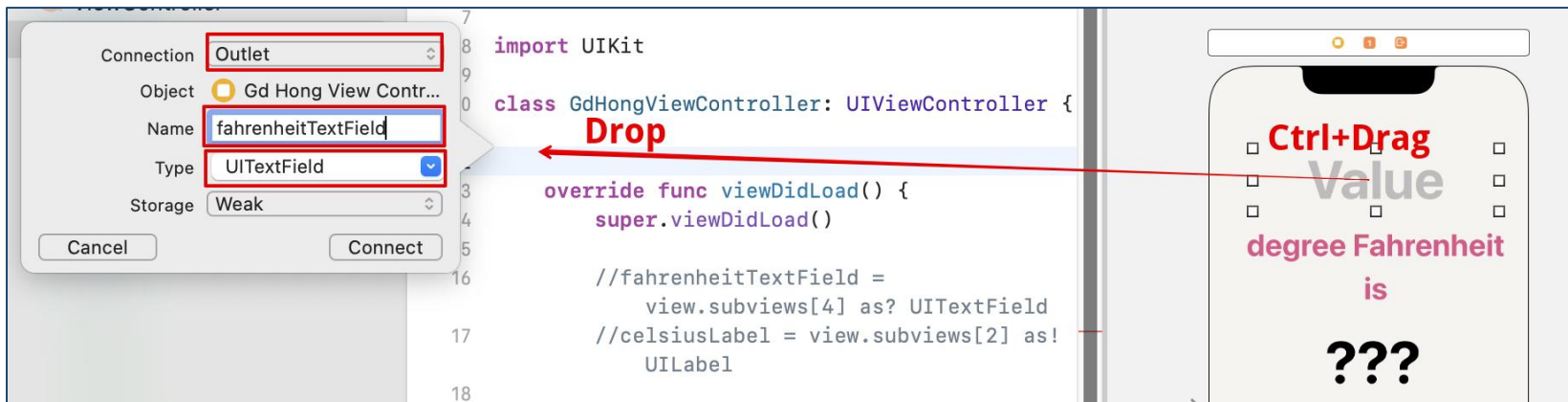
오픈하기를 원하는 파일 선택



UIViewController에서 View Access

■ 두 파일에서 View객체와 변수간 연결 방법1

- 앞과 달리 어떠한 변수도 만들지 말라
- 원하는 View 객체를 선언한 후 **Ctrl+Drag**하여 ConversionViewController의 클래스 변수위치에서 Drop한다
- 그러면 그림과 같은 팝업윈도우가 뜨고 outlet를 선택한 후 변수이름을 넣으면 된다.



- "???"에 대해서도 동일하게 하면 connect를 클릭하면 다음과 같이 outlet변수가 생긴다.

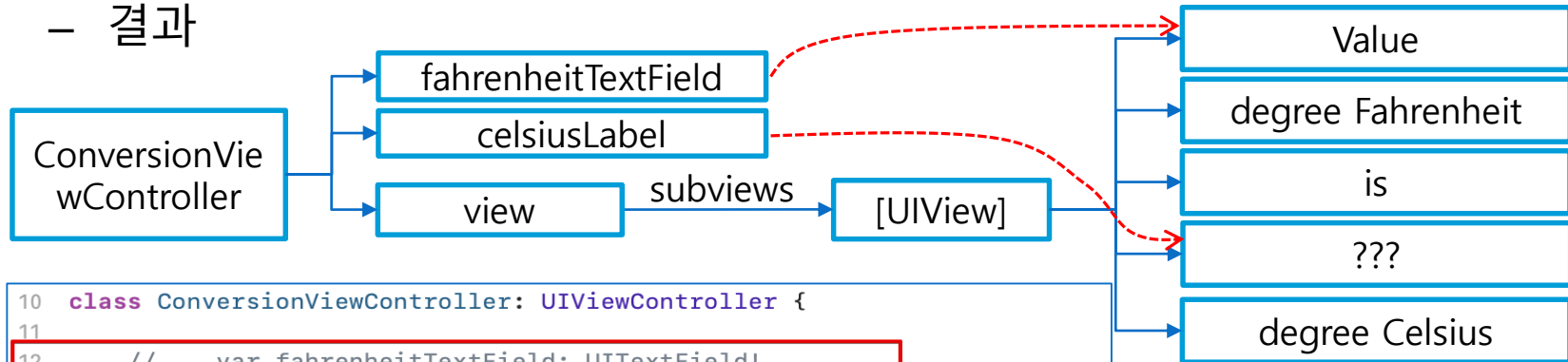
```
4 class GdHongViewController: UIViewController {  
    @IBOutlet weak var fahrenheitTextField: UITextField!  
    @IBOutlet weak var celsiousLabel: UILabel!  
    override func viewDidLoad() {  
        super.viewDidLoad()  
    }  
}
```



UIViewController에서 View Access

■ 두 파일에서 View객체와 변수간 연결 방법1

- 결과



```
10 class ConversionViewController: UIViewController {
11
12     // var fahrenheitTextField: UITextField!
13     // var celsiusLabel: UILabel?
14     @IBOutlet weak var celsiusLabel: UILabel!
15     @IBOutlet weak var fahrenheitTextField: UITextField!
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         // Do any additional setup after loading the view.
20
21         //fahrenheitTextField = view.subviews[0] as! UITextField
22         //celsiusLabel = view.subviews[4] as? UILabel
23
24         //(view.subviews[0] as! UITextField).text = "123"
25         fahrenheitTextField.text = "123"
26
27         let celsiusValue = 5.0 / 9.0 * (123.0 - 32)
28
29         //(view.subviews[4] as! UILabel).text = String(celsiusValue)
30         celsiusLabel.text = String(celsiusValue)
31     }
32 }
```




UIViewController에서 View Access

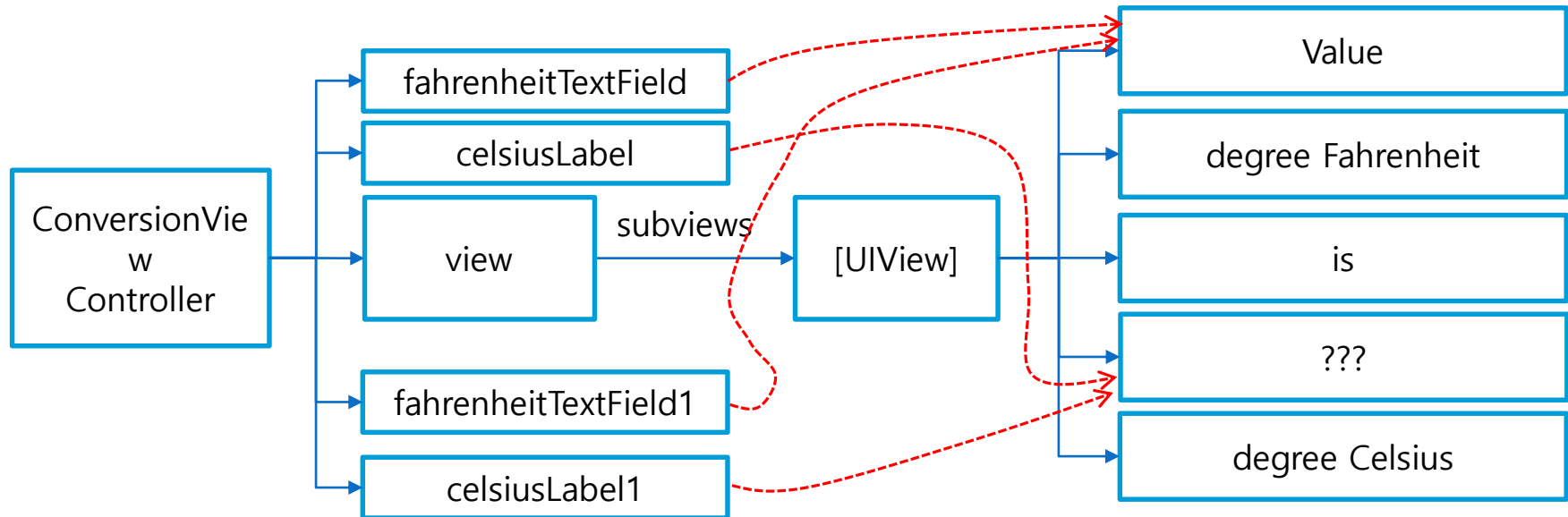
- 두 파일에서 View객체와 변수간 연결 방법2
 - 왼쪽과 같이 ConversionViewController에서 두개의 클래스 변수를 선언한다.
 - 그러면 맨앞에 그림의 16라인 같이 빈 원이 생긴다.
 - 그 원을 Drag(not Control+Drag)하여 원하는 View객체에 Drop한다

```
1 //
2 // ViewController.swift
3 // Conversion
4 //
5 // Created by jmleehs on 2/1/24.
6 //
7
8 import UIKit
9
10 class ConversionViewController:
    UIViewController {
11
12     @IBOutlet weak var celsiusLabel:
        UILabel!
13     @IBOutlet weak var
        fahrenheitTextField:
            UITextField!
14
15     @IBOutlet weak var celsiusLabel1:
        UILabel!
16     @IBOutlet weak var Drag(not Ctl+Drag)
        fahrenheitTextField1:
            UITextField!
17
18 override func viewDidLoad() {
19     super.viewDidLoad()
20     // Do any additional setup
        after loading the view.
21
22     fahrenheitTextField.text =
        "123"
23     let celsiusValue = 5.0 / 9.0
```



UIViewController에서 View Access

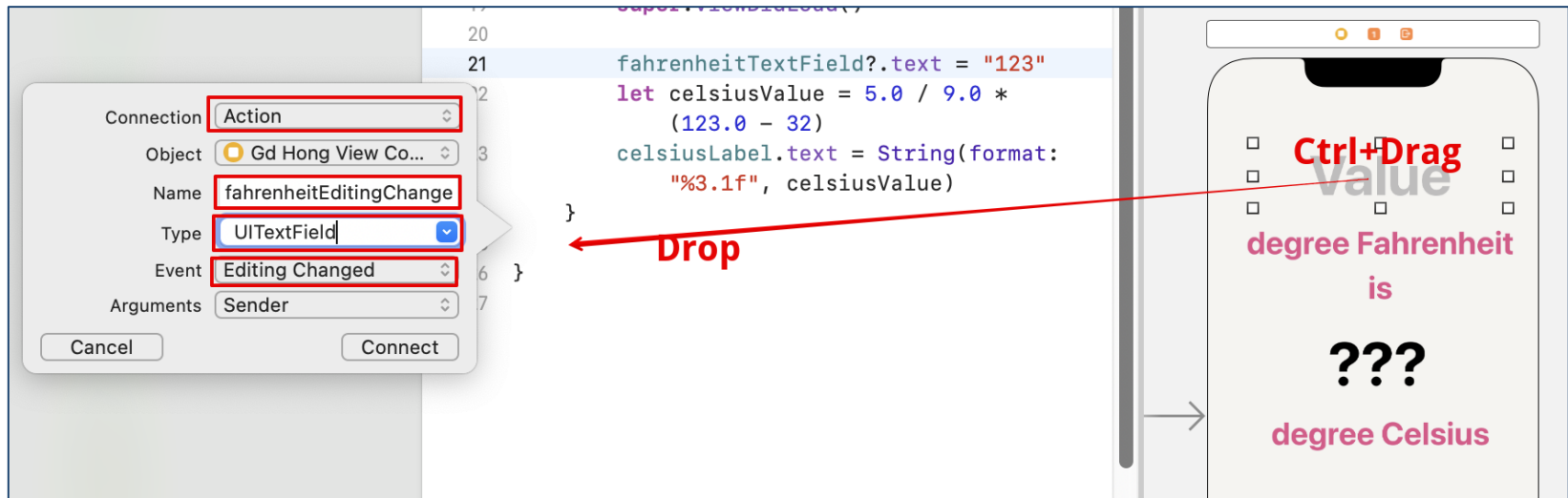
- 두 파일에서 View객체와 변수간 연결 방법2
 - 결과





UIViewController에서 View Access

- 두 파일에서 View객체와 ConversionViewController의 함수 연결 방법
 - 방법2는 생략
 - fahrenheitTextField의 내용이 바뀔때마다 fahrenheitEditingChanged() 함수 호출하도록 연결



- Value객체를 Ctrl+Drag하여 ConversionViewController내의 적절한 위치에 Drop
- 그림과 같은 팝업창이 뜨면
 - 반드시 "Action" 선택
 - 함수이름 적절히 입력(fahrenheitEditingChanged)
 - UITextField 선택
 - 여러 이벤트 핸들러중에서 "Editing Changed"를 선택



UIViewController에서 View Access

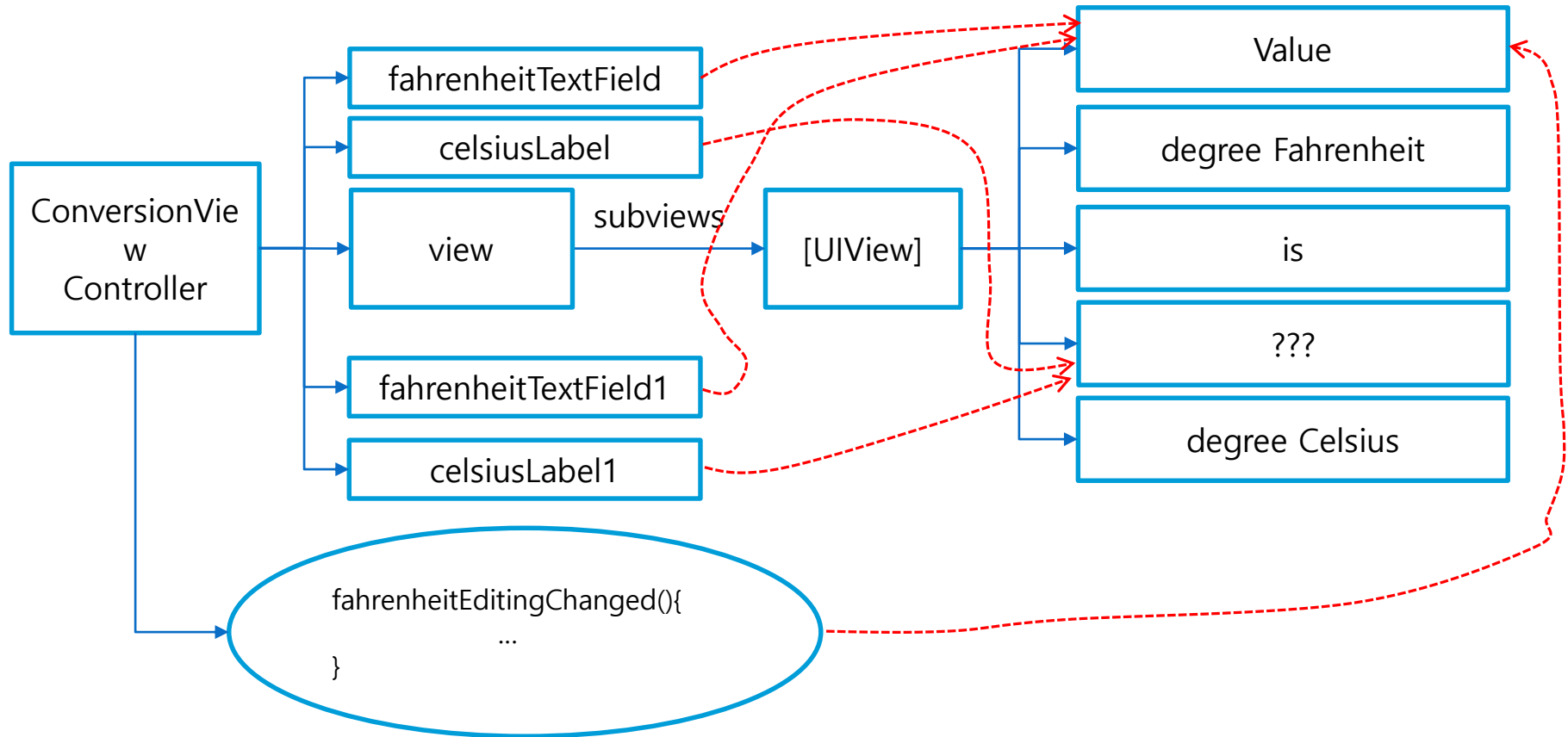
- 두 파일에서 View객체와 ConversionViewController의 함수 연결 방법
 - 결과 다음과 같은 함수가 생긴다.

```
class ConversionViewController: UIViewController {  
  
    @IBOutlet weak var celsiusLabel: UILabel!  
    @IBOutlet weak var fahrenheitTextField: UITextField!  
  
    @IBOutlet weak var celsiusLabel1: UILabel!  
    @IBOutlet weak var fahrenheitTextField1: UITextField!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        // Do any additional setup after loading the view.  
  
        fahrenheitTextField.text = "123"  
        let celsiusValue = 5.0 / 9.0 * (123.0 - 32)  
        celsiusLabel.text = String(celsiusValue)  
    }  
  
    @IBAction func fahrenheitEditingChanged(_ sender: UITextField) {  
    }  
}
```



UIViewController에서 View Access

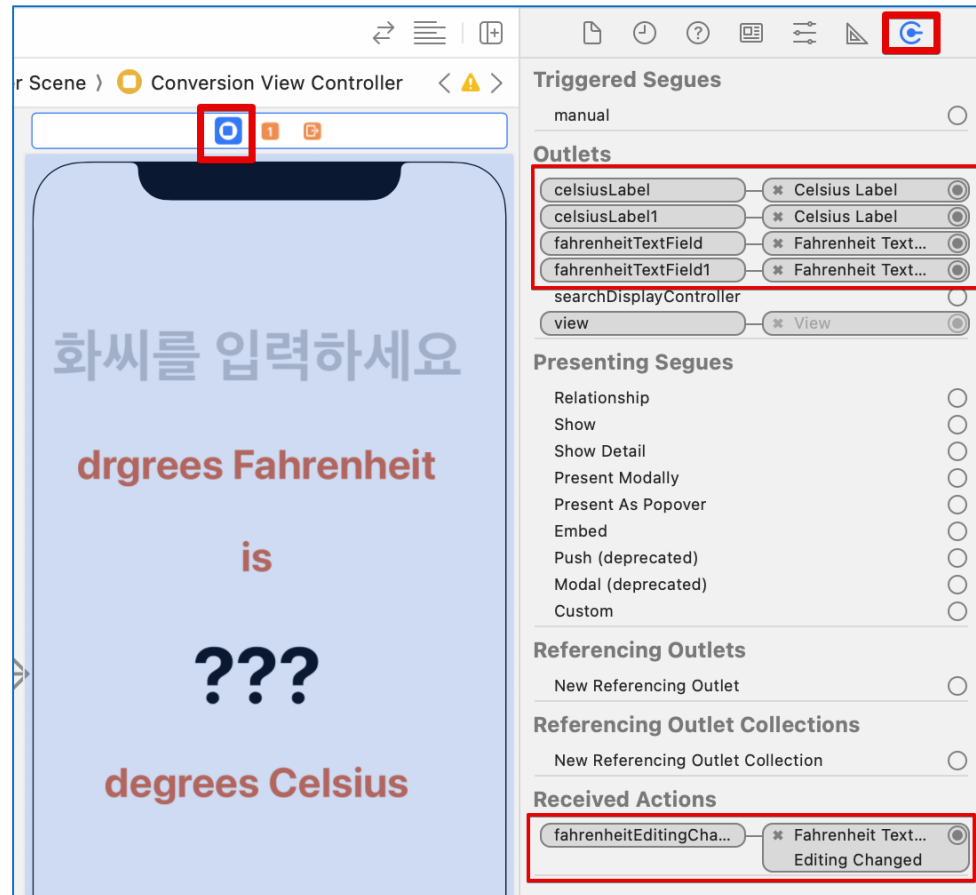
- 두 파일에서 View객체와 ConversionViewController의 함수간 연결 방법
 - 결과





UIViewController에서 View Access

- Connection 내용 확인하기(매우 중요)
 - 스토리보드에서 ViewController 선택
 - Inspector에서 마지막 탭 선택
 - 그러면 연결정보를 알 수 있다.





UIViewController에서 View Access

- **Connection 내용 확인하기(매우 중요)**
 - xcode의 초기 개발자의 90%이상이 직면하는 에러
 - 스토리보드의 View 객체와 ViewController 사이의 연결을 만든 후 ViewController에서 변수(또는 함수)를 지우는 경우
 - 현재 fahrenheitTextField1, celsiusLabel1 변수는 필요없다. 지우자

```
class ConversionViewController: UIViewController {
    @IBOutlet weak var celsiusLabel: UILabel!
    @IBOutlet weak var fahrenheitTextField: UITextField!

    // 아래 두 변수를 지웠다.
    //@IBOutlet weak var celsiusLabel1: UILabel!
    //@IBOutlet weak var fahrenheitTextField1: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        fahrenheitTextField.text = "123"
        let celsiusValue = 5.0 / 9.0 * (123.0 - 32)
        celsiusLabel.text = String(celsiusValue)
    }
    @IBAction func fahrenheitEditingChanged(_ sender: UITextField) {
    }
}
```



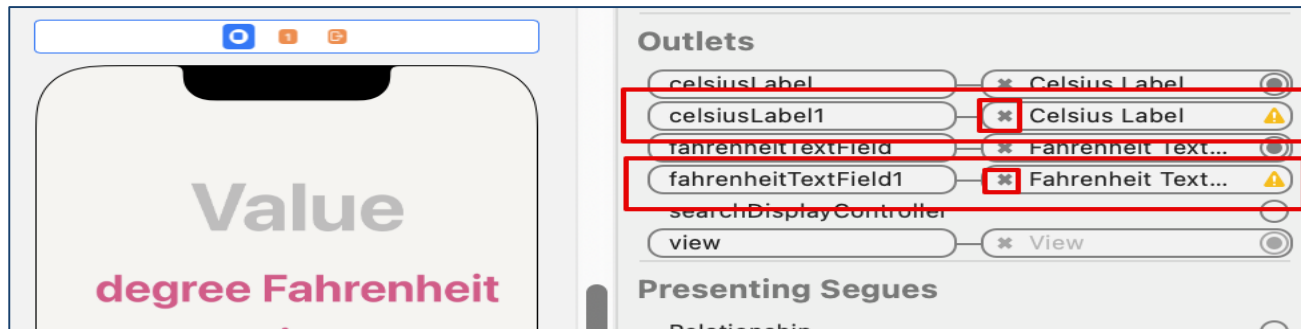
UIViewController에서 View Access

■ 테스트

- ConversionViewController.swift에서 지운후에 실행해 보자. 아래와 같은 에러를 직면한다

```
10 @main
11 class AppDelegate: UIResponder, UIApplicationDelegate { Thread 1: "[<Conversion.C...
12
13 func application(_ application: UIApplication, didFinishLaunchingWithOptions
    launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
```

- 이것은 변수는 지웠지만 연결정보는 아직 남아 있기 때문이다.
 - x를 클릭하여 지워야 한다



- 2개의 connection 정보를 지워라



코드 완성하기

■ 현재 코드 테스트

- 다음과 같이 간단한 코드를 넣자. 그리고 UITextField에 값을 입력해보라

```
@IBAction func fahrenheitEditingChanged(_ sender: UITextField) {  
    print(fahrenheitTextField.text!)  
}
```



```
1  
12  
123  
1234
```

- 값을 입력할때 마다 fahrenheitTextField의 내용이 출력됨을 알수 있다.



코드 완성하기

■ 코드 정리 및 완성하기

- 화씨를 섭씨로 바꾸는 함수 $f(\text{화씨}) = 5.0 / 9.0 * (\text{화씨} - 32.0)$

```
class ConversionViewController: UIViewController {

    @IBOutlet weak var celsiusLabel: UILabel!
    @IBOutlet weak var fahrenheitTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

    }

    @IBAction func fahrenheitEditingChanged(_ sender: UITextField) {
        // 여기서 sender이 fahrenheitTextField 객체이다.
        if let text = sender.text { // 그내용이 뭔가 있으면, 옵셔널 바인딩

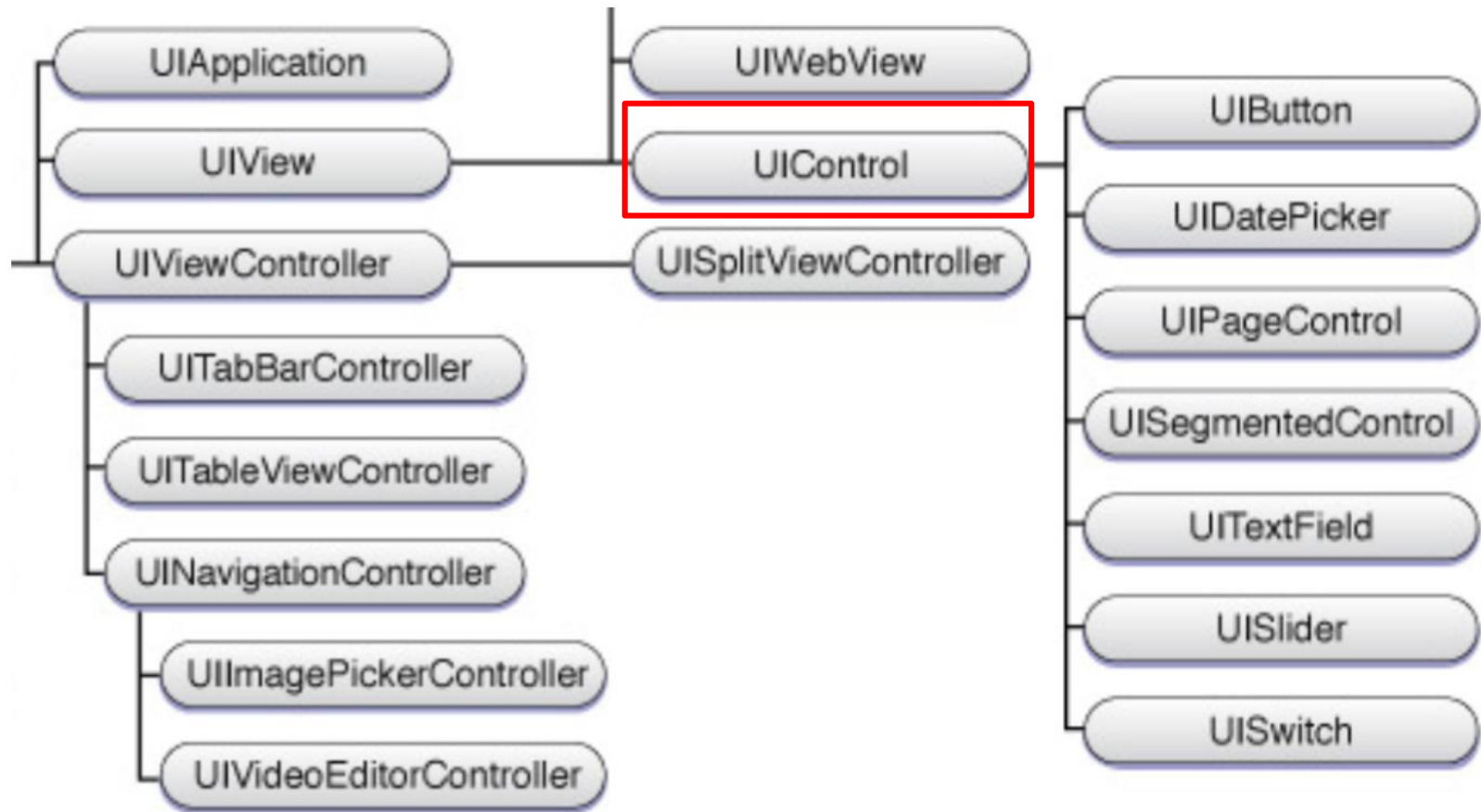
            // 문자열 -> Double 형 변환, Double(text)가 nil수도 있다
            if let fahrenheitValue = Double(text){
                let celsiusValue = 5.0/9.0*(fahrenheitValue - 32.0) // 변환
                celsiusLabel.text = String(format: "%.2f", celsiusValue)
            }else{
                celsiusLabel.text = "???"
            }
        }
    }
}
```



UIView의 이벤트 핸들러

■ UIControl

- 이벤트를 받아서 처리할 수 있는 UIView

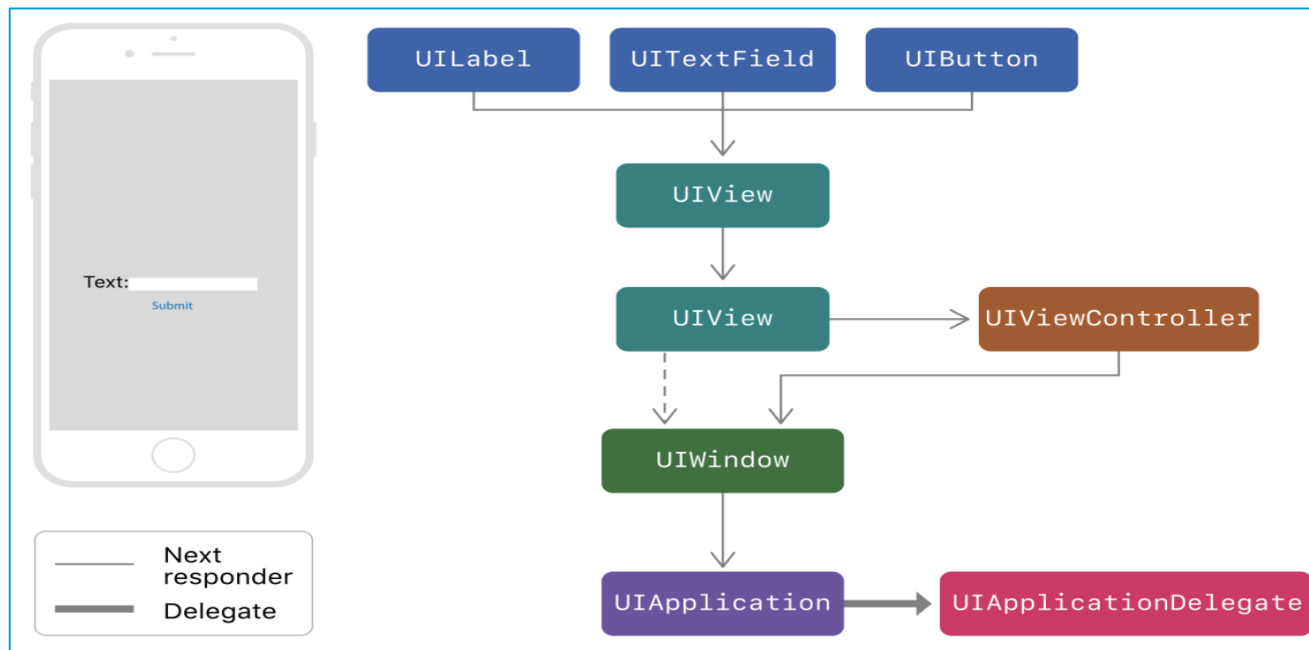




UIResponder

■ UIResponder 클래스

- UIView의 수퍼 클래스이다. → 모든 UI객체는 이 객체의 하위 객체임
 - 슬라이드4의 클래스 다이어그램을 보라
- 모든 이벤트에 대하여 응답하는 객체임
- Managing the Responder Chain(이벤트의 흐름)





First Responder

■ First Responder

- Responder Chain에서 가장 먼저 이벤트를 받는 UI 객체를 지정

이벤트 타입	First Responder
터치 이벤트	터치가 발생한 뷰
프레스 이벤트	포커스를 가진 뷰
흔들기 이벤트	사용자나 UIKit이 지정한 뷰
원격 이벤트	사용자나 UIKit이 지정한 뷰
편집 메뉴 이벤트	사용자나 UIKit이 지정한 뷰

■ First Responder 지정

- ios가 지정
- 이것으로 지정되면 해당 View의 becomeFirstResponder을 호출

■ First Responder의 해제

- resignFirstResponder호출로 해제



First Responder

- **First Responder로 UITextField**
 - becomeFirstResponder에서 키보드를 나타나도록 한다
 - resignFirstResponder에서 키보드를 사라지게 한다.
- **시뮬레이터에서 키보드 사용**
 - I/O-->Keyboard-->Toggle Software Keyboard
 - 입력 후 키보드를 없어지게 해보라

↑
없어지지 않는다



키보드 숨기기

■ View에 Tap 제스처로 해결

- TextField외 어느 곳이든 클릭(Tapping)하면 호출되는 함수 지정
- UITapGestureRecognizer 객체 생성 및 view에 부착

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    let tapGesture = UITapGestureRecognizer(target: self, action:  
                                           #selector(dismissKeyboard))  
    view.addGestureRecognizer(tapGesture)  
}
```

오브젝티브C의 호출 룰에 따르므로 @objc를 붙여야함

탭핑이 일어나는 경우
호출되는 Callback 함수

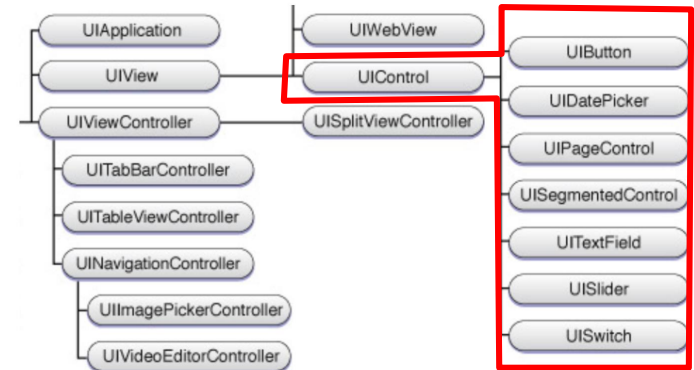
```
extension ConversionViewController{  
    @objc func dismissKeyboard(sender: UITapGestureRecognizer){  
        fahrenheitTextField.resignFirstResponder()  
    }  
}
```



Delegation

■ Delegation

- 특정 객체가 해야 하는 일을 대행하는 위임자
- 하나의 Delegate는 여러 개의 Callback 함수를 내포함



객체A → nil // 객체A가 이벤트를 받아도 아무런 행동을 하지 않는다.

delegate: Xdelegate 타입

객체A → // 객체A가 이벤트를 받으면
객체B의 적절한 Callback함수를 실행

객체B: XDelegate

- Callback1
- Callback2
- ...
- CallbackN

이런 것을 protocol이라 한다.
자바에서는 interface라 한다

객체A는 대부분
UIView이다

객체B는 대부분
UITableViewController이 된다



UITextField를 위한 Delegation

- UITextFieldDelegate 프로토콜
 - 프로토콜은 자바에서 인터페이스와 같다

```
public protocol UITextFieldDelegate : NSObjectProtocol {  
    @available(iOS 2.0, *)  
    optional public func textFieldShouldBeginEditing(_ textField: UITextField) -> Bool // return NO to disallow editing.  
    ...  
    @available(iOS 10.0, *)  
    optional public func textFieldDidEndEditing(_ textField: UITextField, reason: UITextFieldDidEndEditingReason)  
    // if implemented, called in place of textFieldDidEndEditing:  
    @available(iOS 2.0, *)  
    optional public func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange,  
    replacementString string: String) -> Bool // return NO to not change text  
    @available(iOS 2.0, *)  
    optional public func textFieldShouldClear(_ textField: UITextField) -> Bool  
    // called when clear button pressed. return NO to ignore (no notifications)  
    @available(iOS 2.0, *)  
    optional public func textFieldShouldReturn(_ textField: UITextField) -> Bool  
    // called when 'return' key pressed. return NO to ignore.  
}
```

하위 클래스에서 반드시 구현 안해도 된다.

문자가 입력할 때마다 callback 함수 호출하도록 등록



입력 오류 체크를 위한 Delegation

■ UITextFieldDelegate 생성

- 아래 그림과 같이 별도의 extension에서 UITextFieldDelegate를 상속받는다.
- extension 내부({ })에서 textField를 타이핑하면 관련 딜리게이트 함수에 대한 팝업창이 뜬다.
- 그곳에서 원하는 딜리게이트 함수를 선택한다.

```
44 extension ConversionViewController: UITextFieldDelegate{
45     textField
46     [M] textFieldDidEndEditing(_ textField: UITextField)
47     [M] textFieldDidBeginEditing(_ textField: UITextField)
         [M] textFieldDidChangeSelection(_ textField: UITextField)
         [M] textFieldShouldClear(_ textField: UITextField) -> Bool
         [M] textFieldShouldReturn(_ textField: UITextField) -> Bool
         [M] textFieldShouldEndEditing(_ textField: UITextField) -> Bool
         [M] textFieldShouldBeginEditing(_ textField: UITextField) -> Bool
         [M] textFieldDidEndEditing(_ textField: UITextField, reason: UITextField.DidEndEditingReason)
         Tells the delegate that editing stopped for the specified text field.
```

- 선택후의 모습

```
44 extension ConversionViewController: UITextFieldDelegate{
45     func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {
46         code
47     }
48 }
```



입력 오류 체크를 위한 Delegation

UITextFieldDelegate 생성

```
extension ConversionViewController: UITextFieldDelegate{

    func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

        let existing = textField.text?.range(of: ".")
        let comming = string.range(of: ".")
        if exist != nil, comming != nil {
            return false
        }
        return true
    }
}
```

Delegator 지정

```
override func viewDidLoad() {
    super.viewDidLoad()
    let tapGesture = UITapGestureRecognizer(target: self, action:
                                                #selector(dismissKeyboard))
    view.addGestureRecognizer(tapGesture)

    fahrenheitTextField.delegate = self
}
```

입력 오류 체크를 위한 **Delegation**

■ 테스트

- .을 1번 이상 입력하면 입력이 되지 않는다.

