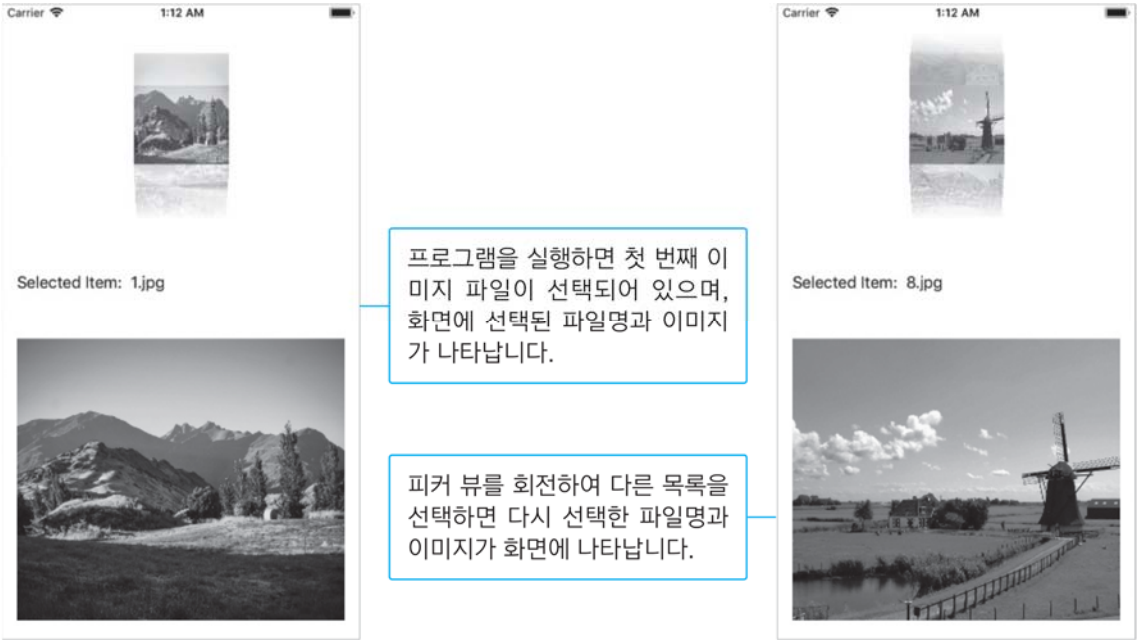
05 장.피커 뷰를 사용해 원하는 항목 선택하기

피커 뷰는 아이폰에서 날짜와 시간을 선택하거나 여러 항목 중 하나를 선택하는 화면에서 볼 슈 있다.
앞으로 살펴본 데이트 피커처럼 아이폰을 사용한 적이 있다면 여러분도 피커 뷰에서 원하는 항목을 선택해 보았을 것이다.

피커뷰를 사용해 선택한 이미지의 파일명을 피커 뷰의 선택 목록에 표시하고 선택된 목록을 가져오는 방법을 알아보자.
그리고 선택한 파일명을 레이블에 출력한 후 이미지 뷰를 사용하여 해당 이미지 파일을 화면에 출력해 보자

더불어 피커 뷰의 선택 목록에 텍스트가 아닌 이미지를 표시하는 방법도 알아보자. 아래 완성된 모습을 미리 보고 앞으로 어떤 기능을 구현할 지 머릿속에 그려 보자.



05-1 피커 뷰란?

피커 뷰(Picker View)는 아이폰에서 원하는 항목을 선택할 수 있 게 해주는 객체로, 피커(Picker)라고도 합니다. 데이트 피커가 날짜와 시간을 선택하기 위한 객체라면 피커 뷰는 문자열을 선택하기 위한 객체입니다.

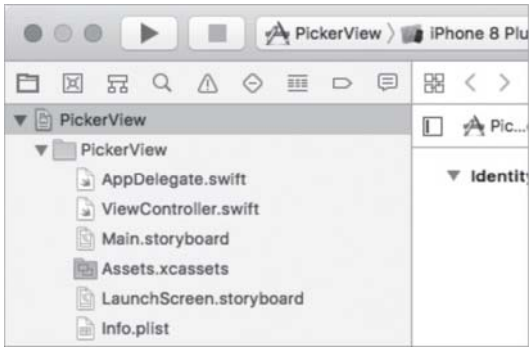
고속버스모바일 앱

05-2 피커 뷰 앱을 위한기본 환경 구성하기

이 앱에서 구현할 기능은 피커 뷰의 롤렛 중 하나를 선택하면 해당 이미지를 이미지뷰에 나타내 주는 기능이다.

이를 위해 스토리보드를 아이폰 모양으로 바꾸고 이미지를 추가해 보자.

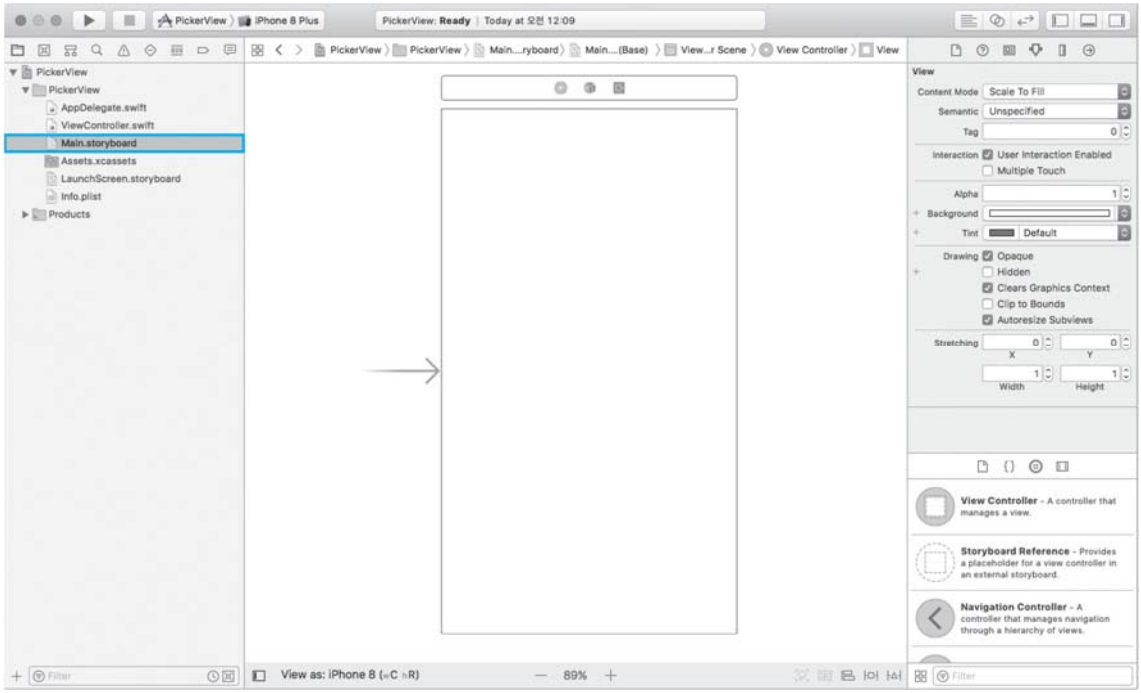
1. Xcode 를 실행한 후 'PickerView'라는 이름으로 프로젝트를 만든다,



2. 뷰 컨트롤러 크기 조절하기

Xcode 화면이 처음 열렸을 때 화면 왼쪽의 내비케이터 영역에서 [Main.storyboard]를 선택하면 스토리보드가 화면에 나타난다.

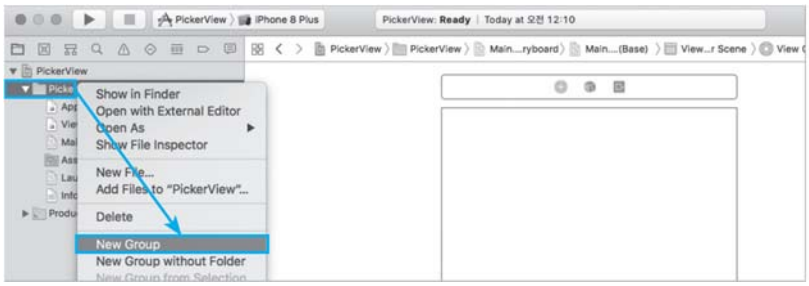
아이폰 모양의 뷰 컨트롤러 크기를 사용자의 상황에 맞게 조절한다.



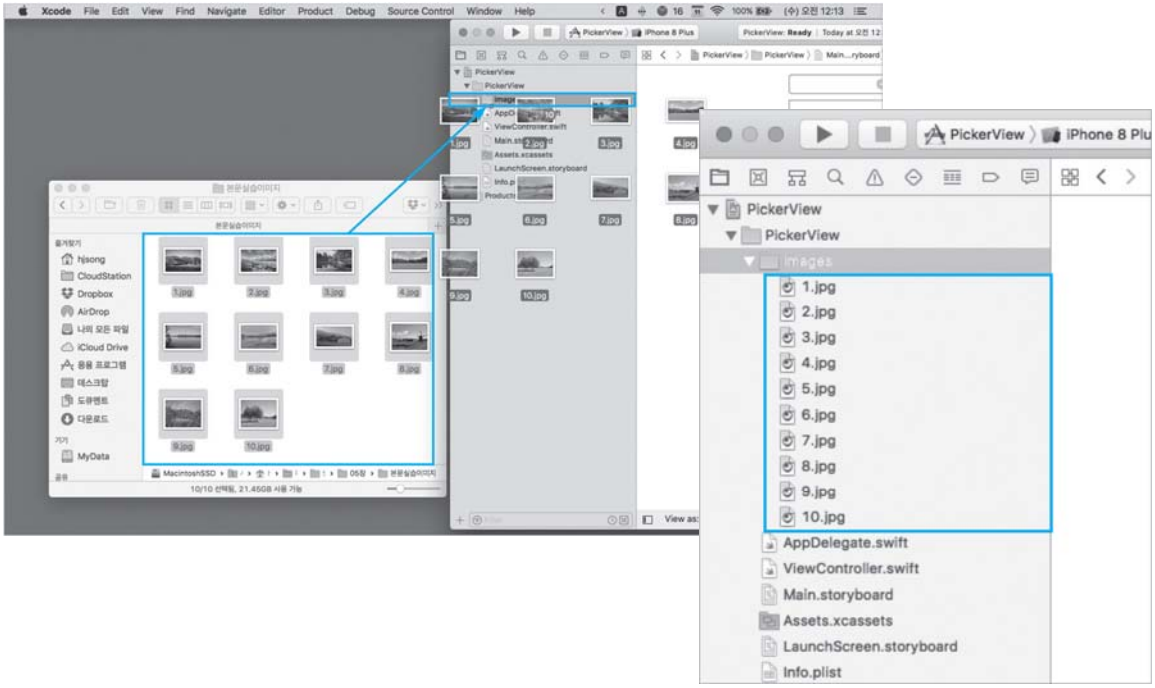
3. 이미지 추가하기

앱에서 사용할 이미지를 프로젝트에 추가해 보겠다.

여러 이미지를 추가해야 할 경우 파일을 관리하기 위해 그룹을 만드는 것이 좋다.
내비게이터 영역에서 [PickerView]폴더를 선택한 후 마우스 오른쪽 버튼을 클릭하여 [New Group]을 선택한다.
그리고 그룹 이름을 'images'라고 입력한다.



4. 파인더에서 원하는 이미지를 선택한 후 앞에서 만든 [images]그룹으로 끌어와 추가한다.
사용가능한 이미지 포멧은 JPEG, JPEG2000, TIFF, PICT, GIF, PNG, ICNS, BMP, ICO 이다.
추가한 이미지를 프로젝트 폴더로 복사하기 위하여 [Destinition : Copy items needed]항목에 체크가 되어 있는지 확인 한 후 [Finish]버튼을 클릭한다.

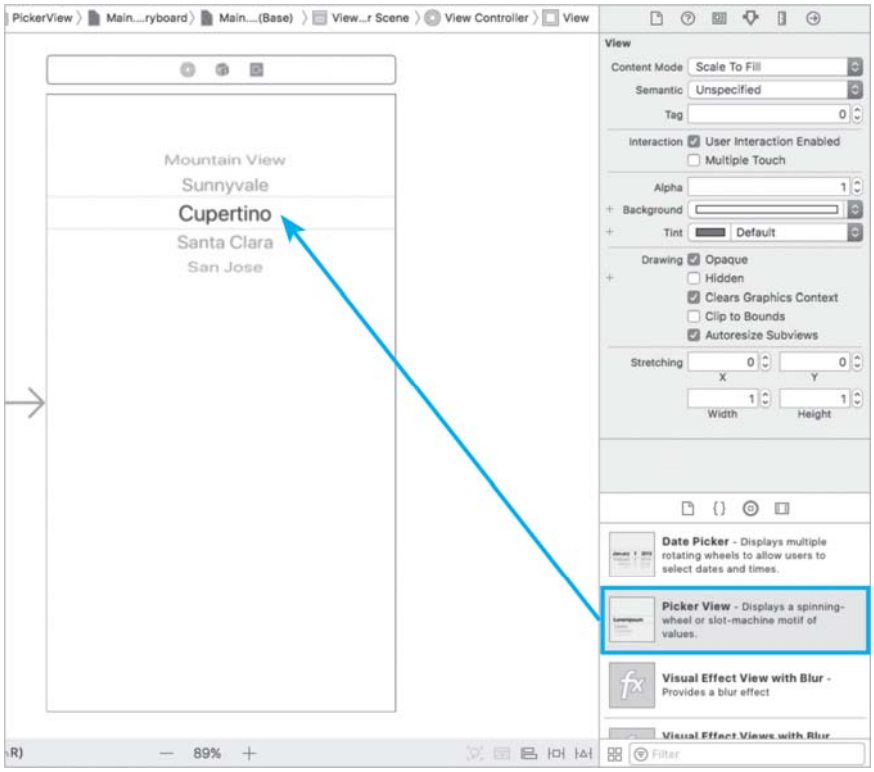


05-3 스토리보드로 피커 뷰 앱 화면 꾸미기

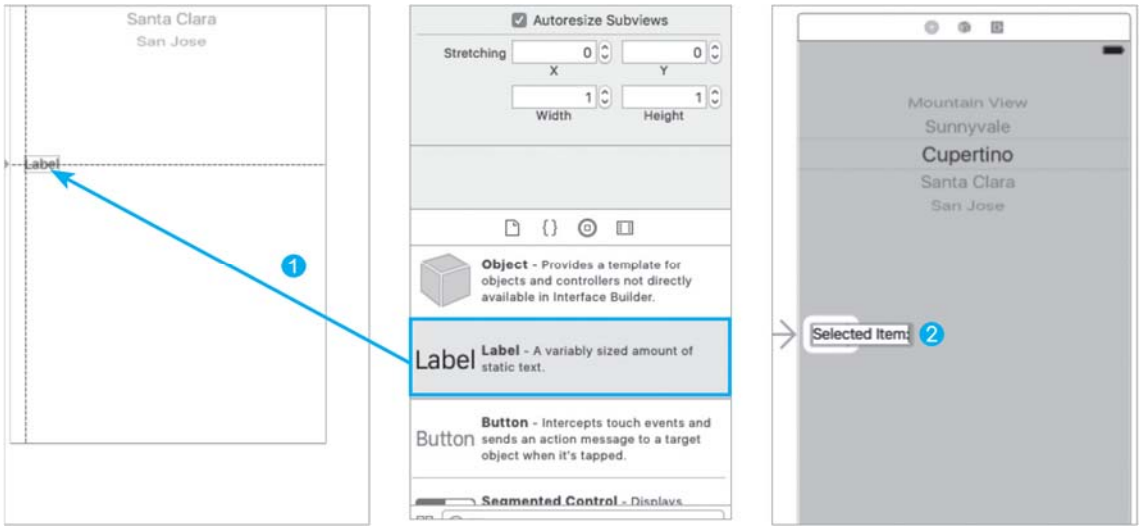
이미지 파일 목록을 보여 주고 선택하기 위한 피커 뷰(PickerView)객체와 선택된 목록에 해당하는 이미지 파일명을 표시할 레이블(Label) 객체를 사용한다.

아래 그림은 완성된 스토리보드 화면이다

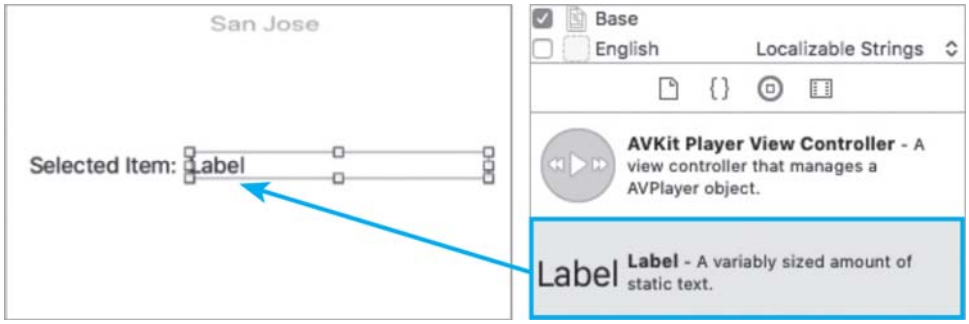
1. 화면 오른쪽 아랫부분의 오브젝트 라이브러리에서 [피커 뷰(PickerView)]를 선택한 후 스토리보드로 끌어와 화면에 배치한다.



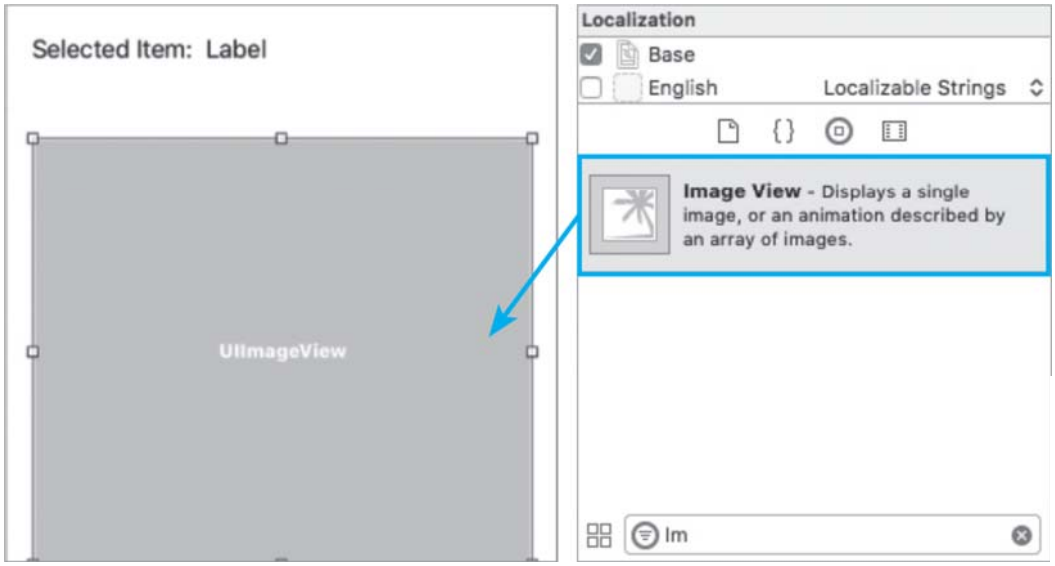
2. 오브젝트 라이브러리에서 레이블 객체를 찾아 스토리보드로 끌어와 왼쪽 중앙에 배치한다. 그리고 추가한 레이블을 더블 클릭하여 'Selected Item:' 이라고 입력한다.



3. 레이블 객체를 하나 더 스토리보드로 끌어와



4. 마지막으로 오른쪽 아랫 부분의 오브젝트 라이브러리에서 임지 뷰 객체를 찾아 스토리보드로 끌어 온 후 아래쪽에 배치하고 크기를 키운다.



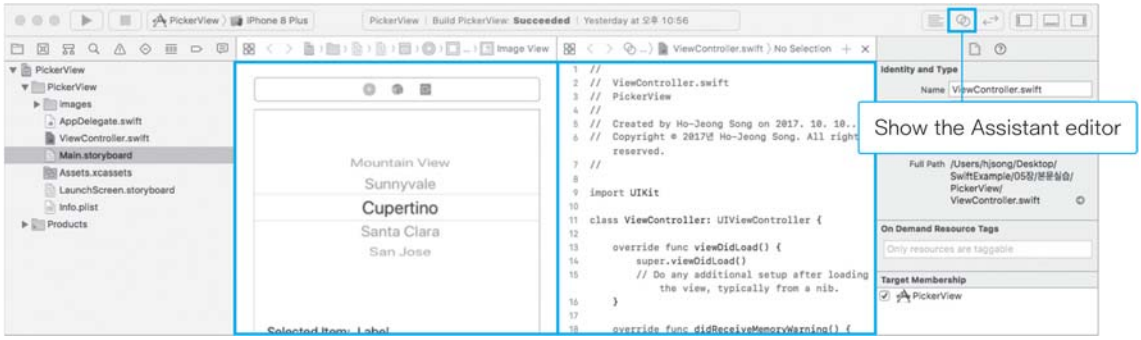
05-4 아웃렛 변수 추가하고 델리게이트 설정하기

이제 프로그램에서 사용할 아웃렛 변수를 추가해 보겠다.

피커 뷰를 선택했을 때 어떤 목록을 선택했는지 알 수 있게 해주는 피커 뷰 아웃렛 변수와 선택한 목록을 레이블에 출력하는 레이블 아웃렛 변수 그리고 선택한 목록에 해당하는 이미지를 이미지 뷰에 보여 줄 이미지 뷰 아웃렛 변수를 추가해 보자.

1. 보조 편집기 영역 열기

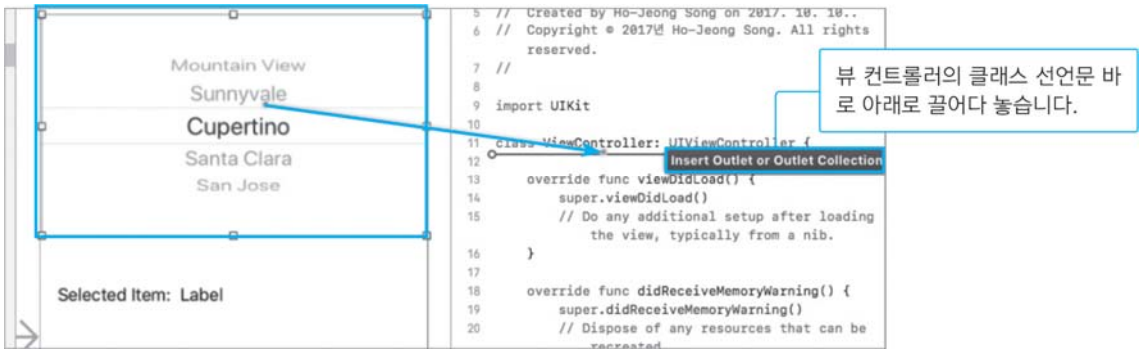
아웃렛 변수와 액션 함수를 추가하려면 우선 오른쪽 윗부분의 [Show the Assistant editor]버튼을 클릭하여 보조 편집기 영역(Assistant editor)을 열어야 한다. 가운데 화면의 스토리 보드 부분이 둘로 나눠지면서 왼쪽에는 스토리 보드, 오른쪽에는 소스를 편집하는 영역이 나타난다.



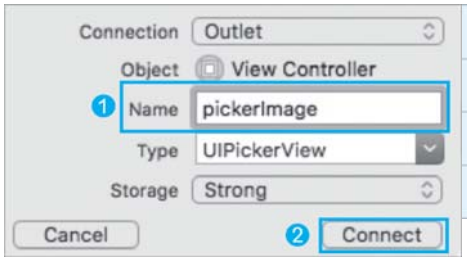
2. 피커 뷰에 대한 아웃렛 변수 추가하기

우선 위쪽 피커뷰에 대한 아웃렛 변수를 추가해 보자. 위쪽의 피커 뷰를 마우스 오른쪽 버튼으로 선택한 후 오른쪽 보조 편집기 영역으로 드래그하면 다음과 같이 연결선이 나타난다.

이 연결선을 뷰 컨트롤러의 클래스 선언문 바로 아래에 배치한 후 마우스 버튼에서 손을 뗀다.

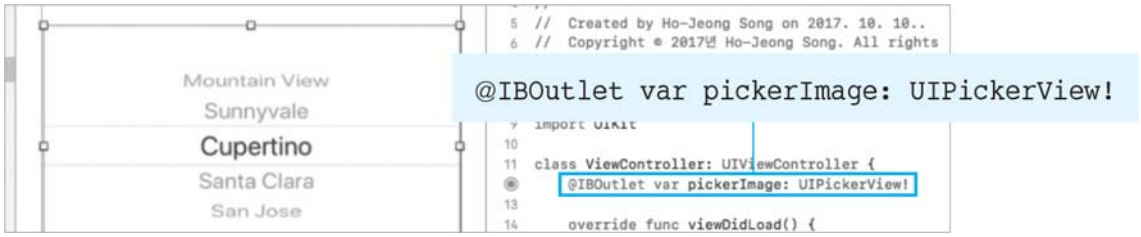


연결 설정 창이 나타나면 다음과 같이 설정 한 후 [Connect]버튼을 클릭하여 피커 뷰와 아웃렛 변수를 연결한다.



위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	pickerImage
유형(Type)	UIPickerView

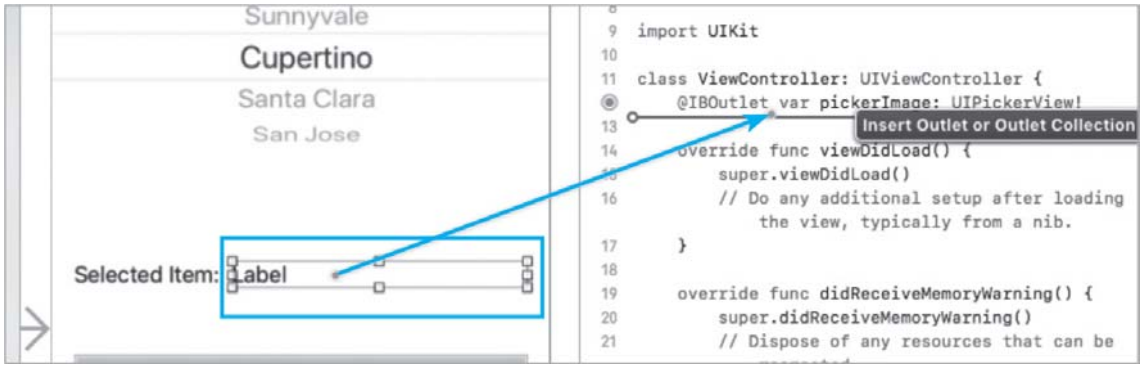
3. 피커 뷰에 대한 아웃렛 변수가 다음과 같이 추가된다.



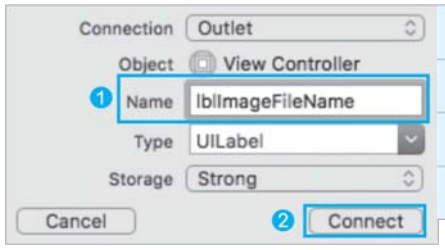
4. 레이블에 대한 아웃렛 변수 추가하기

두 번째로 레이블에 대한 아웃렛 변수를 추가해 보자.

앞에서와 같은 방법으로 마우스 오른쪽 버튼을 [레이블(Label)]을 선택한 후 드래그해서 보조 편집기 영역의 조금전에 추가한 피커 뷰 아웃렛 변수 아래로 끌어다 놓는다.

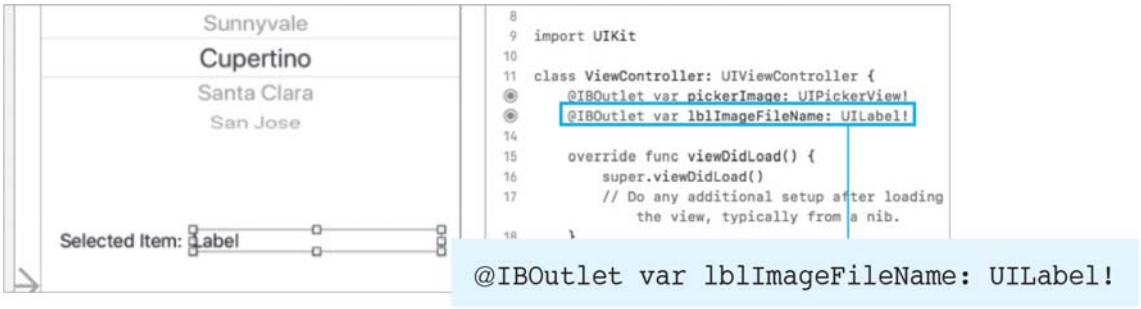


연결 설정 창이 나타나면 다음과 같이 설정한 후 [Connect]버튼을 클릭하여 레이블과 아웃렛 변수를 연결한다.



위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	lblImageFileName
유형(Type)	UILabel

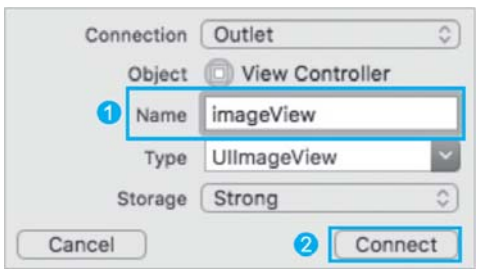
5. 레이블에 대한 아웃렛 변수가 다음과 같이 추가되었다.



6. 레이블 뷰에 대한 아웃렛 변수 추가하기

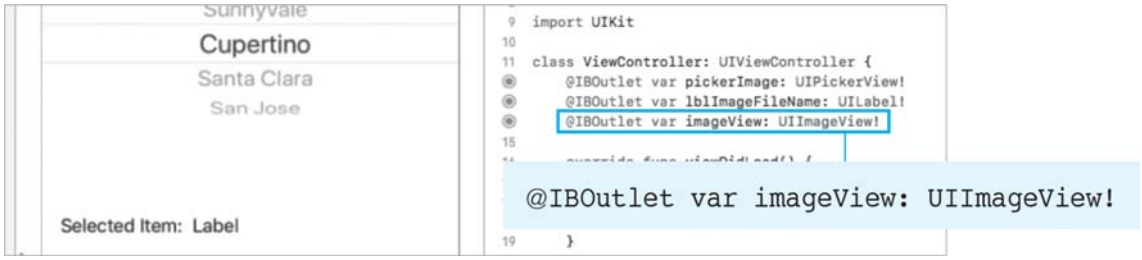
마지막으로 이미지뷰에 대한 아웃렛 변수를 추가해 보자. 앞에서 했던 것과 같은 방법으로 마우스 오른쪽 버튼으로 [이미지 뷰(image View)]를 선택한 후 드래그해서 보조 편집기영역의 조금 전에 추가한 레이블 아웃렛 변수 아래로 끌어다 놓는다.

연결 설정 창이 나타나면 다음과 같이 설정한 후 [Connect]버튼을 클릭하여 이미지 뷰와 아웃렛 변수를 연결한다,



위치	뷰 컨트롤러의 클래스 선언문 바로 아래
연결(Connection)	Outlet
이름(Name)	imageView
유형(Type)	UIImageView

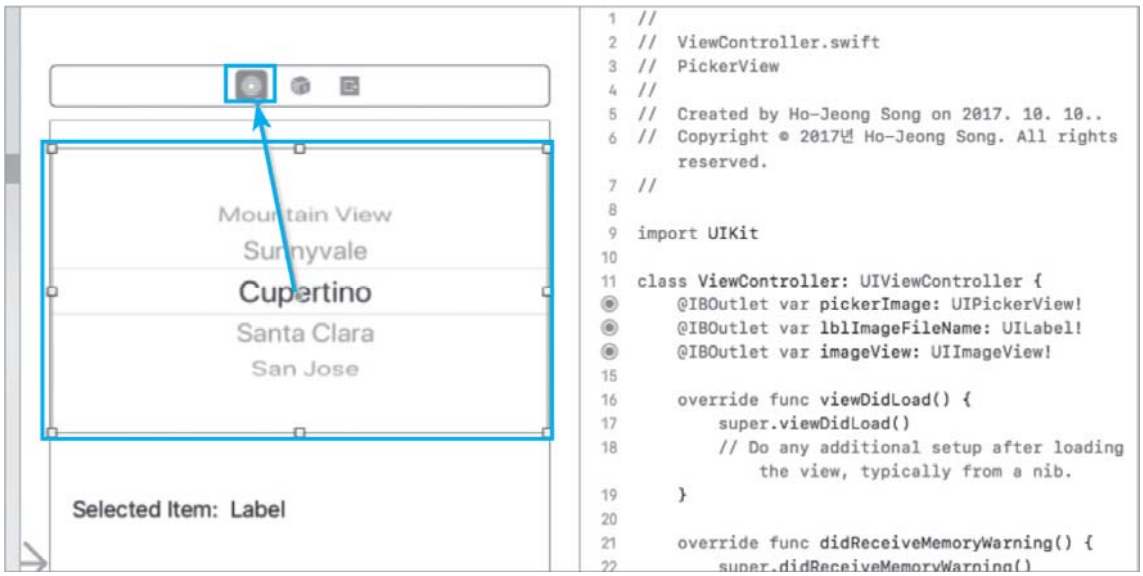
7. 이미지 뷰에 대한 아웃렛 변수가 다음과 같이 추가 되었다.



피커 뷰의 델리게이트 설정하기

피커 뷰가 상호 작용하려면 피커 뷰에 대한 델리게이트 메서드를 사용해야 한다,
델리게이트는 대리자라고도 하며 누군가 해야할 일을 대신 해주는 역할을 한다.
예를 들어 특정 객체와 상호작용할 때 메시지를 넘기면 그 메시지에 대한 책임은 델리게이트에게 위인한다, 그리고 델리게이트 메서드는 해당 역할을 수행하며 처리결과나 메시지등을 받는다. 즉, 사용자가 객체를 터치했을 때 해야 할 일을 델리게이트 메서드에 구현하고 해당 객체가 터치되었을 때 델리게이트가 호출되어 위임받는 일을 하게 되는 것이다,
그렇기 때문에 현재 사용하고 있는 뷰 컨트롤러에서 피커 뷰에 대한 델리게이트 메서드를 사용한다고 설정해야한다,

1. 피커 뷰의 델리게이트 사용을 설정하기 위하여 마우스 오른쪽 버튼으로 피커 뷰를 선택한 후 위쪽의 뷰 컨트롤러 아이콘 위로 끌어다 놓는다.



2. 다음과 같은 선택 화면이 나오면 [delegate]를 선택한다.

이렇게 피커 뷰 객체와 레이블, 이미지 뷰객체에 아웃렛 변수를 추가했다.

아웃렛 변수를 사용해 각 객체에 접근할 수 있게 되었다. 또한 피커 뷰의 델리게이트 메서드를 사용할 수 있도록 설정을 완료하였다.

05-5 피커 뷰 동작 코드 작성하기

피커 뷰를 동작시키기 위한 코드를 작성해 보자. 피커 뷰를 동작시키기 위해서는 피커 뷰 델리게이트 클래스를 상속 받아야 하며 피커 뷰의 델리게이트 메서드를 추가해야 한다.

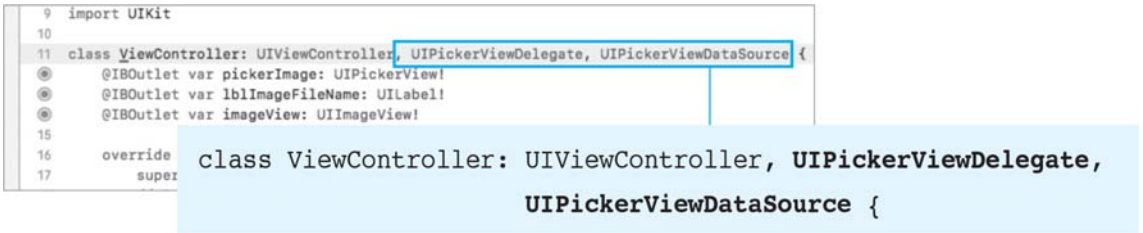
1. 스탠더드 에디터로 화면 모드 수정하기

오른쪽 윗부분에 있는 [Show the Standard editor]버튼을 클릭한 후 왼쪽의 내비게이터 영역에서 [ViewController.swift]를 선택한다.

2. UIPickerViewDelegate 클래스 상속받기

피커 뷰의 델리게이트를 사용하려면 UIPickerViewDelegate 클래스와 UIPickerViewDataSource 클래스를 상속받아야 한다.

ViewController 클래스 선언문 오른쪽에 다음과 같이 ' ,UIPickerViewDelegate, UIPickerViewDataSource'를 입력하면 클래스를 상속 받을 수 있다.



[스위프트 문법] 클래스 상속받기

클래스는 부모 클래스와 자식 클래스로 구분할 수 있다. 자식 클래스는 상속을 받는 클래스, 부모 클래스는 상속되는 클래스를 말한다. 클래스를 상속받는다는 위미는 상속받고자 하는 클래스의 변수 및 함수를 모두 사용할 수 있다는 뜻이다.

상속을 받기 위해서는 클래스를 선언하면서 클래스 이름의 오른쪽에 ':'과 함께 상속받을 클래스의 이름을 입력하면 된다.

우리가 지금까지는 'ViewController.swift'에서 ViewController 클래스를 선언할 때 자동으로 선언되어 알지 못하고 있었지만 항상 UIViewController 클래스를 상속받고 있었다.

```
class ViewController: UIViewController {
...
}
```

이제 여기서 추가로 다른 클래스를 상속받으려면 ':'와 함께 상속받을 클래스의 이름을 추가하면 된다. 앞의 ViewController 클래스에서 추가로 피커 뷰 델리게이트 클래스와 피커 뷰 데이터 소스 클래스를 상속받으려면 다음과 같이 ',UIPickerViewDelegate, UIPickerViewDataSource'를 추가하면 된다.

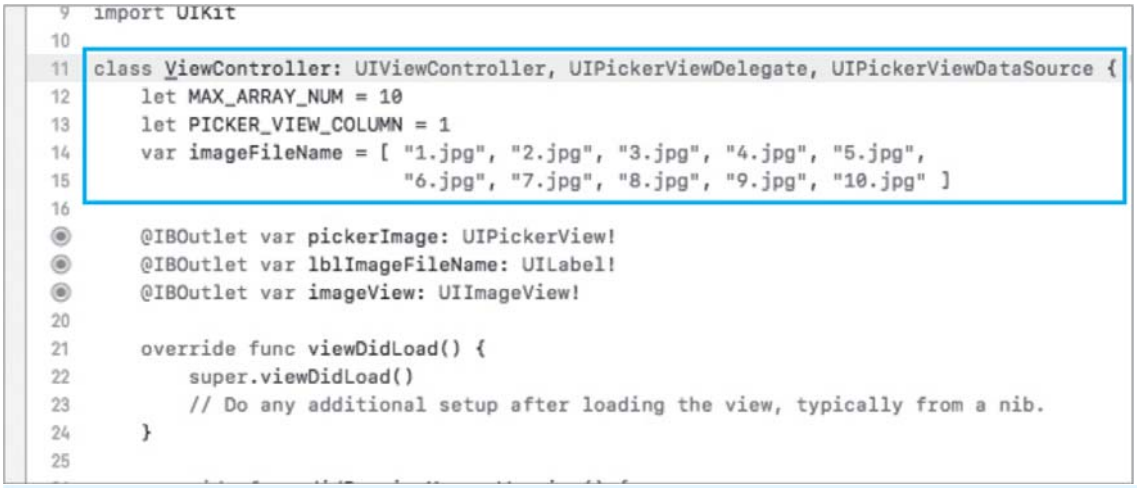
```
class ViewController: UIViewController, UIPickerViewDelegate {
...
}
```

3. 변수 및 상수 추가하기

피커 뷰가 동작하는 데 필요한 변수 및 상수를 추가 해보자.

다음 소스를 ViewController 클래스 선언부와 아웃렛 변수 선언부 사이에 추가한다.

단, 1)번에는 본인이 사용할 이미지의 개수를 2)번에 들어간 이미지 이름은 앞으로 자신이 추가한 이미지의 이름과 파일명에 맞춰 기입한다.




```
class ViewController: UIViewController, UIPickerViewDelegate,
                      UIPickerViewDataSource {
    let MAX_ARRAY_NUM = 10 ❶
    let PICKER_VIEW_COLUMN = 1 ❷
    var imageFileName = [ "1.jpg", "2.jpg", "3.jpg", "4.jpg", "5.jpg",
        "6.jpg", "7.jpg", "8.jpg", "9.jpg", "10.jpg" ] ❸
...
```

1) MAX_ARRAY_NUM : 이미지의 파일명을 저장할 배열의 최대 크기를 지정한다.
2) PICKER_VIEW_COLUMN : 피커 뷰의 열의 개수를 지정한다.
3) imageFileName : 이미지의 파일명을 저장할 배열이다,

4. 피커 뷰에게 무엇을 어떻게 보여주고 어떻게 동작할지를 설정해 보자.
우선 피커 뷰의 동작에 필요한 델리게이트 메서드를 뷰 컨트롤러(ViewController)클래스의 맨 아래에 추가한다.

```
26      override func didReceiveMemoryWarning() {
27          super.didReceiveMemoryWarning()
28          // Dispose of any resources that can be recreated.
29      }
30
31      // returns the number of 'columns' to display.
32      func numberOfComponents(in pickerView: UIPickerView) -> Int {
33          return PICKER_VIEW_COLUMN
34      }
35
36      // returns the # of rows in each component..
37      func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
38          return imageFileName.count
39      }
40
41      func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) ->
            String? {
42          return imageFileName[row]
43      }
44  }
45
```

33 행은 피커뷰에게 컴포넌트의 수를 정수 값으로 넘겨주는 델리게이트 메서드이다.
피커 뷰의 컴포넌트는 피커 뷰에 표시되는 열의 개수를 의미한다.
여기서는 PICKER_VIEW_COLUMN 의 값인 1 을 넘겨준다.
38 행은 numberOfRowINComponent 인수를 가지는 델리게이트 메서드이다.
피커 뷰에게 캄포넌트의 열의 개수를 정수 값으로 넘겨준다.
이 값은 피커 뷰의 해당 열에서 선택할 수 있는 행의 개수(데이터의 개수)를 의미한다.
여기서는 배열의 imageFileName 의 값인 10 을 imageFileName.count 를 사용하여 넘긴다.
42 행은 titleForRow 인수를 가지는 델리게이트 메서드이다,
피커 뷰에게 컴포넌트의 각 열의 타이틀 문자열(String)값으로 넘겨준다.
imageFileName 에 저장되어 있는 파일명을 넘겨 준다.
여기서는 imageFileName 에 저장되어 있는 파일명을 넘겨 준다.

5. 결과 확인
[실행] 버튼을 클릭하여 앱을 실행시켜 보자. 앞에서와 마찬가지로 시뮬레이터의 기기는 [iPhone8]로 선택하고 [실행]버튼을 클릭한다. 피커 뷰의 열의 개수와 행의 개수를 설정하고 각 행의 타이틀을 전달해 주면 피커 뷰가 동작하는 것을 확인할 수 있다.

[스위프트 문법] 배열이란?

배열은 하나의 변수 이름으로 여러 개의 데이터를 저장할 수 있는 공간을 의미합니다. 일반적으로 하나의 변수는 하나의 데이터만 저장할 수 있습니다.

```
var Name = “Ho-Jeong”
var score = 100;
```

하지만 10명 또는 100명의 이름과 점수를 저장하려면 어떻게 해야 할까요? 이름 변수와 점수 변수를 각각 10개씩 또는 100개씩 만들어야 할까요? 이렇게 많은 양의 데이터를 저장하기 위해 배열이라는 자료형을 사용합니다. 즉, 배열이란 하나의 이름으로 여러 개의 데이터를 저장할 수 있는 공간을 만들어서 사용하는 것을 말합니다.

위 그림과 같이 ‘name’이라는 이름으로 N개의 저장 공간을 만들어 사용할 수 있습니다. 여기서 주의해야 할 점은 배열의 번호는 1부터 시작하지 않고 0부터 시작한다는 점입니다.
그럼 선언 방법을 알아보겠습니다.

2) 빈 배열을 선언하고 값을 추가하는 방법

```
var name = [String]()
var score = [Int]()

name.append(“슈퍼맨”)
name.append(“배트맨”)
…

score.append(100);
score.append(80)
…
```

1)처럼 배열을 만들어도 나중에 2)와 같이 append 메서드로 추가할 수 있습니다.
배열을 선언했으면 이제 배열 값을 접근하는 방법을 알아보겠습니다. 접근한다는 의미는 배열 값을 읽어오거나 변경한다는 의미입니다.

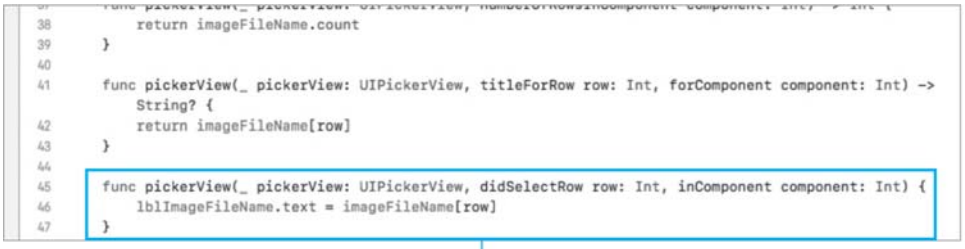
```
let someoneName = name[0]    // name 배열에서 첫 번째 값인 “슈퍼맨”을 읽어옵니다.
neme[1] = “홍길동”            // 배열의 두 번째 값인 name[1]에 “홍길동”을 저장합니다.
                             // 이전에 있던 “배트맨” 값은 사라지고 “홍길동”으로 변합니다.

let someScore = score[2]     // score 배열에서 세 번째 값인 95를 읽어옵니다.
scroe[3] = 60                // 배열의 네 번째 값인 score[3]에 60을 저장합니다.
```

05-6 선택한 이미지 이름과 해당 이미지 출력하기

앞에서 만든 앱을 실행 시켜 보면 피커 뷰가 나타나고 선택 목록에 파일명이 보인다.
그리고 마우스로 피커 뷰를 선택하여 위, 아래로 움직여 보면 피커 뷰의 롤렛이 돌아가는 것을 확인 할 수 있다.
하지만 피커 뷰의 롤렛을 아무리 움직여도 어떤 변화가 생기지는 않는다.
피커 뷰에 아직 어떤 동작을 하라는 코딩을 하지 않았기 때문이다.
피커 뷰의 이미지를 돌려 특정 이미지의 이름을 선택했을 때 해당 이미지의 이름이 레이블에 출력되고 선택한 이미지가 이미지 뷰에 나타나도록 코딩해 보자.

1. 코드 수장 및 추가하기

사용자 피커 뷰의 롤렛을 돌려 원하는 열을 선택했을 때 할 일을 델리게이트에게 지시하는 메서드를 맨 아래에 추가한다. 피커 뷰의 델리게이트 메서드 중 didSelectRow 인수가 포함된 메서드는 사용자가 피커 뷰의 롤렛을 선택했을 때 호출된다.
사용자가 선택한 피커 뷰의 row 를 사용하여 원하는 일을 코딩할 수 있다.



```
38          return imageFileName.count
39      }
40
41      func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) ->
            String? {
42          return imageFileName[row]
43      }
44
45      func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
46          lblImageFileName.text = imageFileName[row]
47      }
```

```
func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
        inComponent component: Int) {
        lblImageFileName.text = imageFileName[row] ①
}
```

1) 사용자가 피커 뷰의 롤렛에서 선택한 row 값을 사용하여 imageFileName 배열에서 row 값에 해당하는 문자열을 가지고 온다. 그라 | 고 거져온 문자열을 레이블의 아웃렛 변수인 lblImageFileName.txt 에 저장한다.

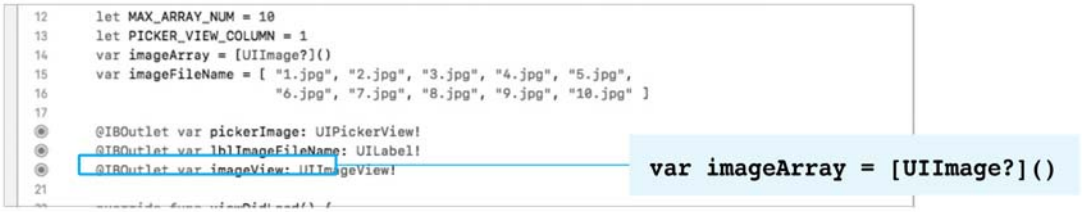
2. 결과 보기

[실행]버튼을 클릭하여 결과응 확인 한다, 피커 뷰의 델리게이트 메서드를 사용하여 롤렛이 선언되었을 때 레이블에 선택된 파일명을 출력하도록 코딩하였기 때문에 파일명이 레이블에 출력되는 것을 확인할 수 있다.

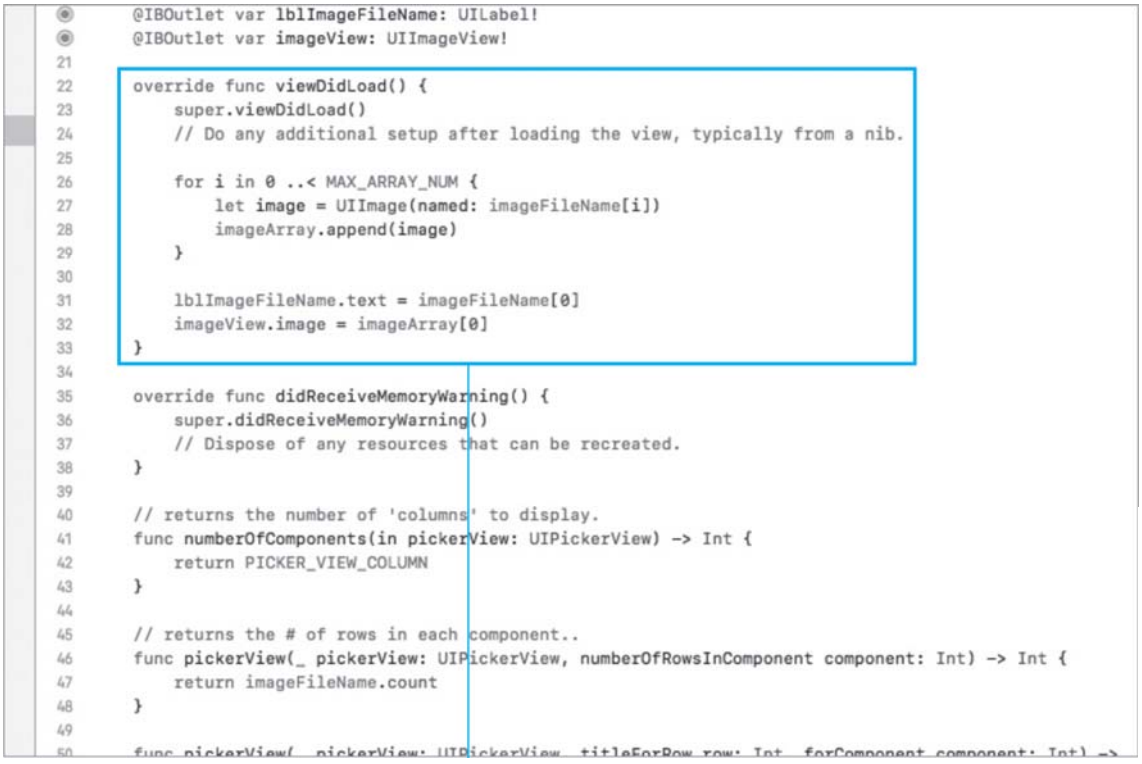
** 선택한 이미지를 이미지 뷰에 출력하기

이제는 피커 뷰의 롤렛을 선택할 경우 선택된 파일명에 해당하는 이미지를 이미지 뷰에 출력해 보자

1. 앞에서 변수를 추가한 위치 중에서 PICKER_VIEW_COLUMN 과 imageFileNAme 사이에 UIImage 타입의 배열 imageArray 를 선언한다.(선언하는 위치는 어느 위치든 상관없다. 상수, 변수, 아웃렛 변수는 가장 윗 부분에 선언하고 함수와 아랫부분에 선언하면 된다.)



2. 뷰가 로드되었을 때 MAX_ARRAY_NUM 의 갯수만큼 imageFileName 에 있는 이미지를 가져와 UIImage 타입의 상수 image 에 할당하고, 할당된 image 를 배열 imageArray 에 추가한다. 또한 레이블과 이미지 뷰에 배열의 처음에 해당하는 imageFileName[0]과 imageArray[0]을 각각 출력한다.



26 행의 i 라는 변수를 0 부터 MAX_ARRAY_NUM 보다 작을 때까지 루프를 실행한다.

0 ~ 9 까지루프를 실행한다.

27 행의 image 라는 변수를 선언한다. imageFileName[i]에 있는 파일명을 사용하여 UIImage 타입의 이미지를 생성한다. 생성한 이미지를 image 라는 변수에 할당한다.

28 행은 imageArray 배열에 방금 만든 image 를 추가한다.
31 행은 lblImageFileName 레이블에 ImageFileName 배열의 첫번째 파일명을 출력한다.
32 행은 이미지 뷰에 첫번째 이미지가 나타난다.

[스위프트 문법] for 루프

for 루프는 특정 코드를 특정한 조건을 만족하는 동안 반복해서 실행하기 위해 제공하는 기능입니다. 오른쪽 코드처럼 사용할 수 있으며, 변수 값이 Range 안에 있을 동안 변수 값을 1씩 증가시키면서 {, } 안의 코드를 반복해서 실행합니다.

```
for 변수 in Range {
      ...
}
```

Range를 설정하는 방법은 다음과 같습니다. 변수 i를 0부터 9까지 반복하면서 print(i)를 출력합니다. 즉, 0부터 9까지 출력합니다.

```
for i in 0...9 {
    print(i)
}
```

변수 i를 0부터 10보다 작을 동안 반복하면서 print(i)를 출력합니다. 즉, 0부터 9까지 출력합니다. 결과는 같지만 조건이 다른 것을 확인할 수 있습니다.

```
for i in 0..<10 {
    print(i)
}
```

3. 마지막으로 피커 뷰의롤렛이 선택되었을 때 동작하는 didSelectRow 인수가 포함된 델리게이트 메서드의 맨 아랫부분에 선택한 이미지를 이미지뷰에 나타내 주는 코드를 추가한다.

```
54      func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
55          lblImageFileName.text = imageFileName[row]
56          imageView.image = imageArray[row]
57      }
```

51 행은 사용자가 피커 뷰의 롤렛에서 선택한 row 값을 사용하여 imageArray 배열에서 row 값에 해당하는 이미지를 가지고 온다. 그리고 가져온 이미지를 이미지 뷰의 아웃렛 변수인 imageView.image 에 저장한다.

4. 결과 보기
[실행] 버튼을 클릭하여 결과를 확인한다. 롤렛을 선택하면 선택된 파일명은 레이블에 이미지는 이미지 뷰에 나타나는 것을 확인 할 수 있다.

05-7 피커 뷰 룰렛에 파일명 대신 이미지 출력하기

1. 코드 수정 및 추가하기

앞에서 선언한 titleForRow 인수를 가지는 델리게이트 메서드를 주석 처리한다.

titleForRow 인수를 가지는 델리게이트메서드는 각 row 에 대한 타이틀을 정의하는 메서드이다.

```
45      // returns the # of rows in each component..
46      func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
47          return imageFileName.count
48      }
49
50  //    func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) ->
        String? {
51  //        return imageFileName[row]
52  //    }
53
54      func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
55          lblImageFileName.text = imageFileName[row]
56          imageView.image = imageArray[row]
57      }
```

2. viewForRow 인수가 포함되어 있는 델리게이트 메서드를 새롭게 추가한다.

viewForRow 인수가 포함되어 있는 메서드는 각 row 의 view 를 정의하는 메서드이다.

새롭게 추가되는 메서드는 각 row 의 view 를 설정하고 UIView 타입을 리턴한다.

여기서는 imageView 를 리턴한다.

```
        String? {
51  //        return imageFileName[row]
52  //    }
53
54      func pickerView(_ pickerView: UIPickerView, viewForRow row: Int, forComponent component: Int,
            reusing view: UIView?) -> UIView {
55          let imageView = UIImageView(image:imageArray[row])
56          imageView.frame = CGRect(x: 0, y: 0, width: 100, height: 150)
57
58          return imageView
59      }
60
61      func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
62          lblImageFileName.text = imageFileName[row]
63          imageView.image = imageArray[row]
64      }
```

```
func pickerView(_ pickerView: UIPickerView, viewForRow row: Int, forComponent
                component: Int, reusing view: UIView?) -> UIView {
    let imageView = UIImageView(image:imageArray[row])
    imageView.frame = CGRect(x: 0, y: 0, width: 100, height: 150)

    return imageView
}
```

1) 피커 뷰에게 컴포넌트의 각 열의 뷰를 UIView 타입의 값으로 넘겨준다.

여기서는 이미지 뷰에 저장되어 있는 이미지를 넘겨준다.

2) 선택된 row 에 해당하는 이미지를 iomageArray 에서 가져온다.

3) 이미지 뷰의 프레임 크기를 설정한다.

4) 이미지 뷰를 리턴 한다.

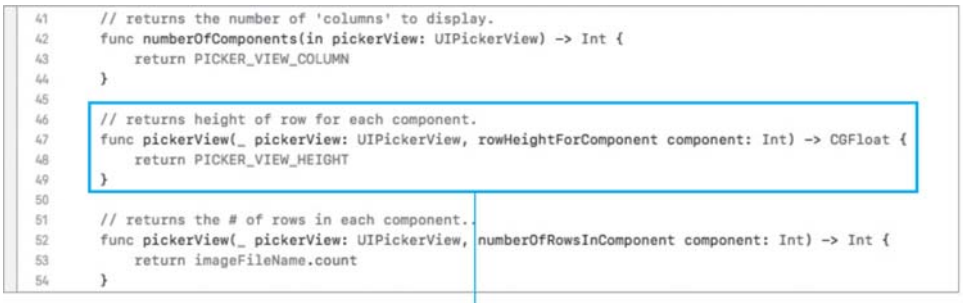
** 롤렛의 높이 변경하기

피커 뷰도 보기는 좋지만 이미지의 세로 높이가 좁아 이미지가 잘 보이지 않는다,
마지막으로 피커 뷰 롤렛의 세로 높이를 변경해 보자.

1. 피커 뷰의 높이를 지정할 상수를 맨 위의 상수 선언부 중 PICKER_VIEW_COLUMN 과 imageArray 사이에 정의합니다.


```
let PICKER_VIEW_HEIGHT:CGFloat = 80
```

2. 피커 뷰의 높이를 전달할 피커 뷰 델리게이트 메서드를 numberOfComponents 메서드 아래에 추가한다.




```
// returns height of row for each component.
func pickerView(_ pickerView: UIPickerView, rowHeightForComponent
                component: Int) -> CGFloat {
    return PICKER_VIEW_HEIGHT
}
```

1) 피커 뷰에게 컴포넌트의 높이를 정수 값으로 넘겨주는 델리게이트 메서드이다.
여기서는 PICKER_VIEW_HEIGHT 의 값인 80 을 넘겨준다.