iOS

간단하게 iOS에 대해서 알아보자

8교시 정창용

iOS
개념

- iOS란, Apple이 생산하는 제품 중 모바일 기기에 탑재되는 운영 체제, iPhone, iPod touch가 사용하고 있는 모바일 운영 체제이다.

그럼 iOS를 개발할 때 어떤 언어를 써야 하나요?

C, C++, Objective-C 등이 있지만 저희가 오늘 공부할 언어는...

Swift

입니다.

Swift의 기본 로고

그럼 Swift란?

손쉽게 학습할 수 있는 강력한 프로그래밍 언어.

- Swift는 macOS, iOS, watchOS, tvOS를 위한 강력하고 직관적인 프로그래밍 언어입니다.

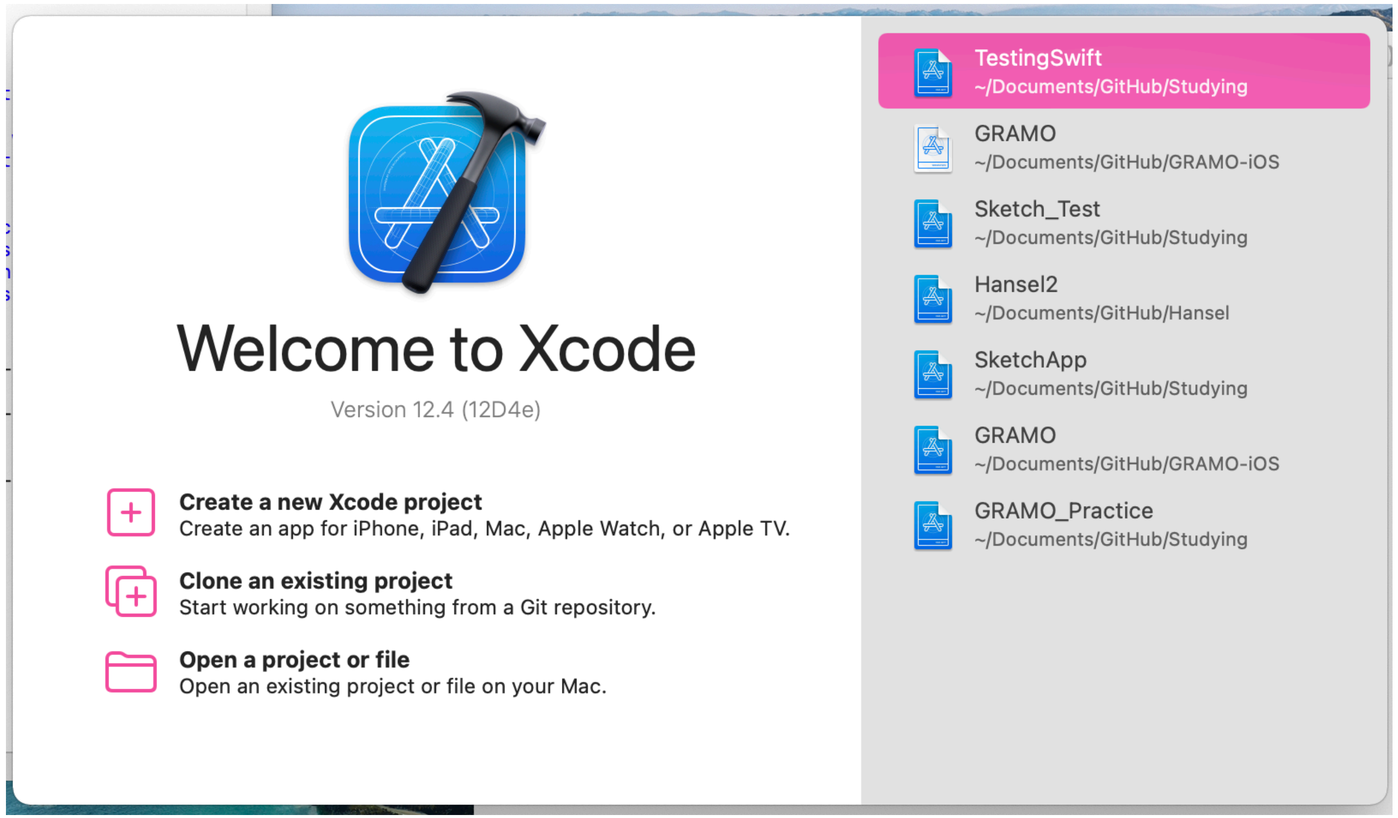
- Swift 코드는 안전하게 설계되었으며 빛의 속도로 빠르게 실행되는 소프트웨어도 제작할 수 있습니다.

그럼 지금부터 Swift를 직접 사용해보도록 하겠습니다

맥북에서 Xcode를 찾아 실행시켜 줍니다.
Xcode는 Apple의 macOS, iOS, watchOS 및 tvOS 개발 전용 IDE.

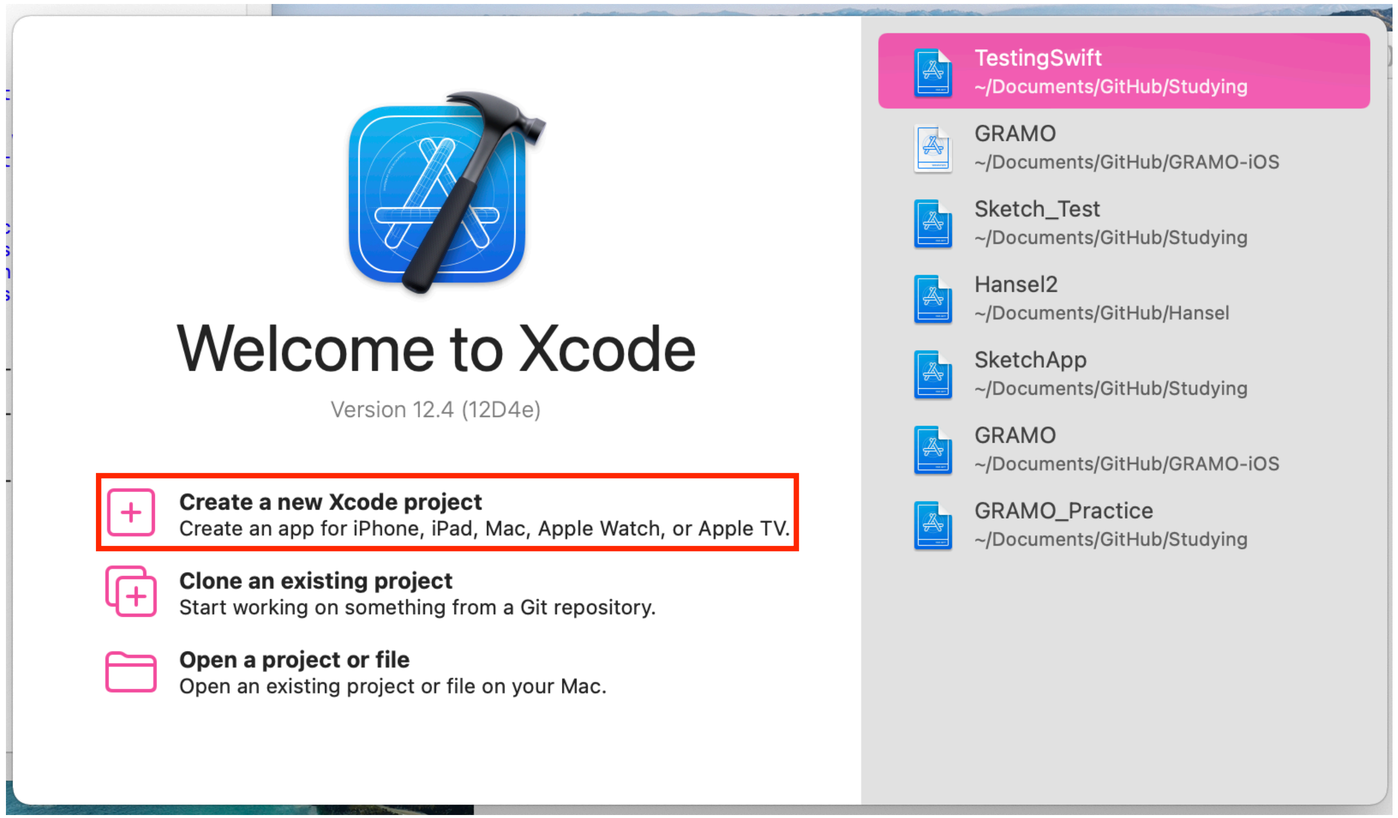
Xcode에서 프로젝트 만들기

Xcode를 처음 실행했을 때 뜨는 화면입니다.

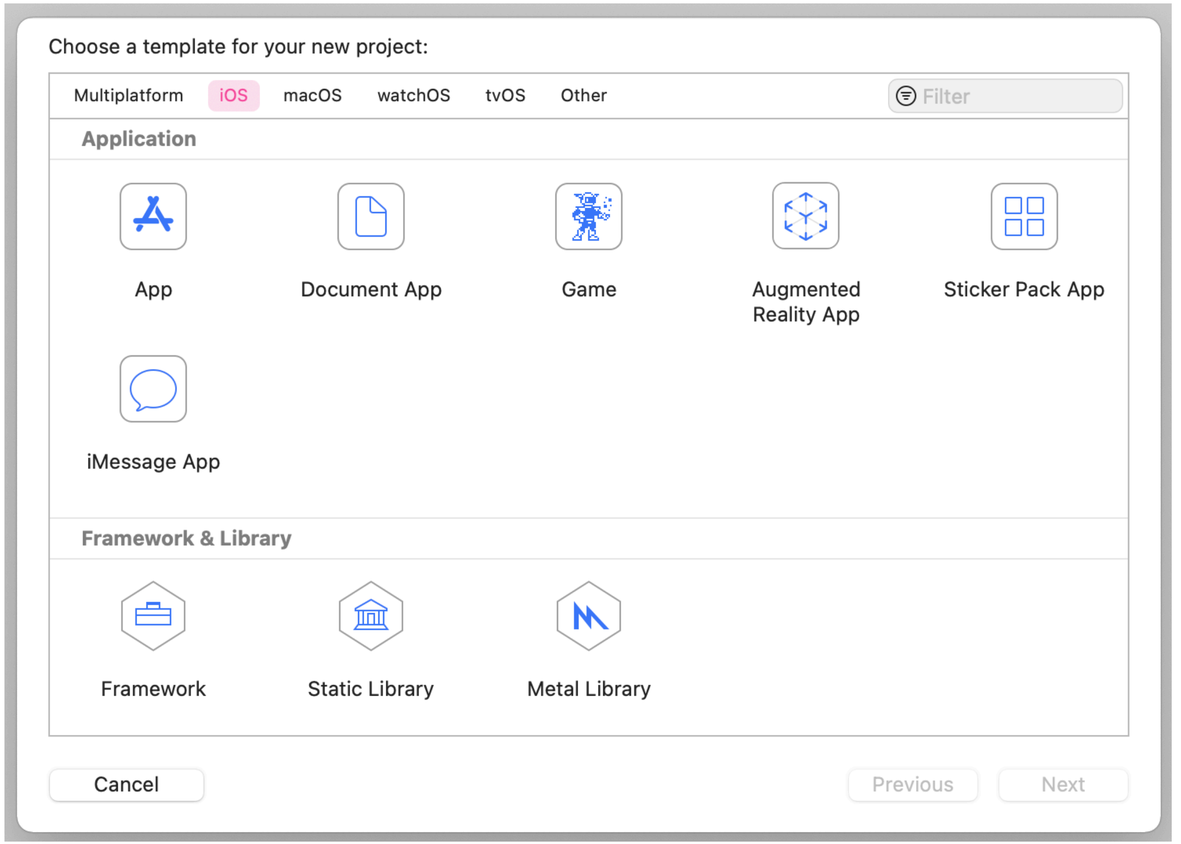


Xcode에서 프로젝트 만들기

빨강색으로 표시된 버튼을 눌러 들어가줍니다.

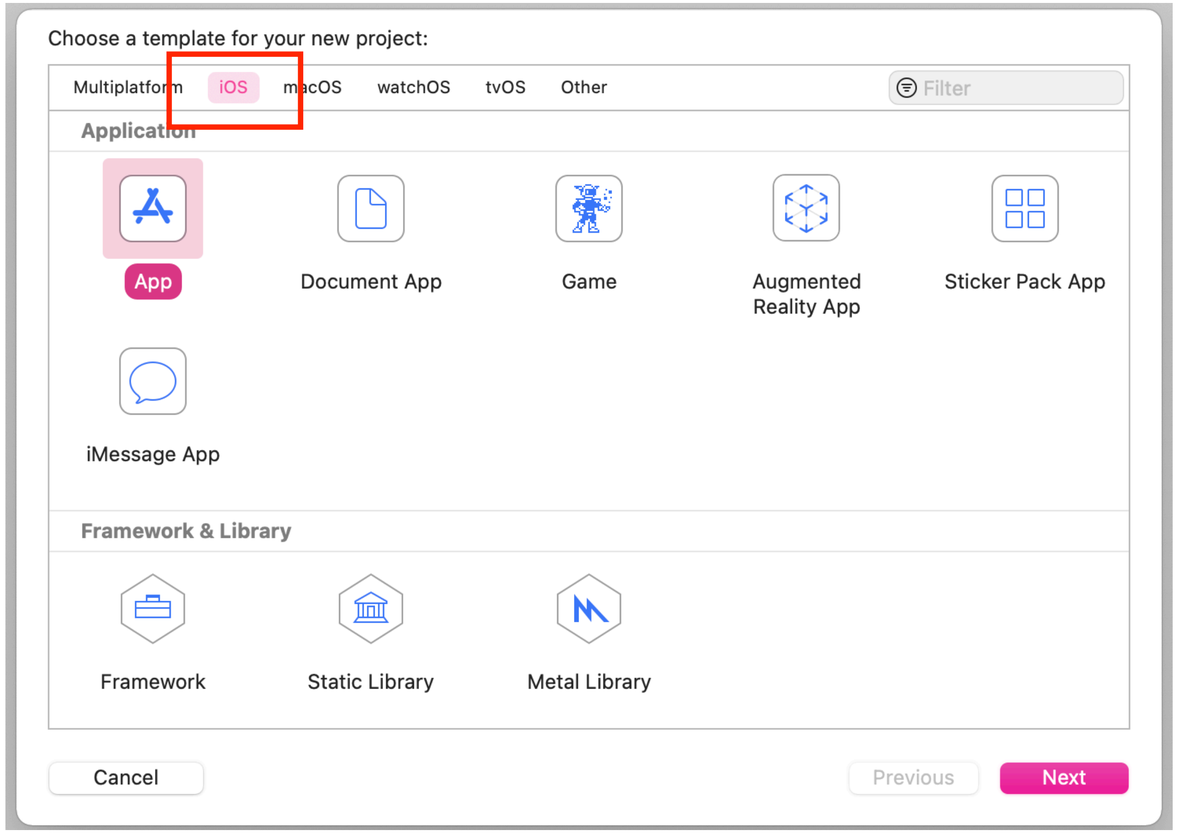


Xcode에서 프로젝트 만들기



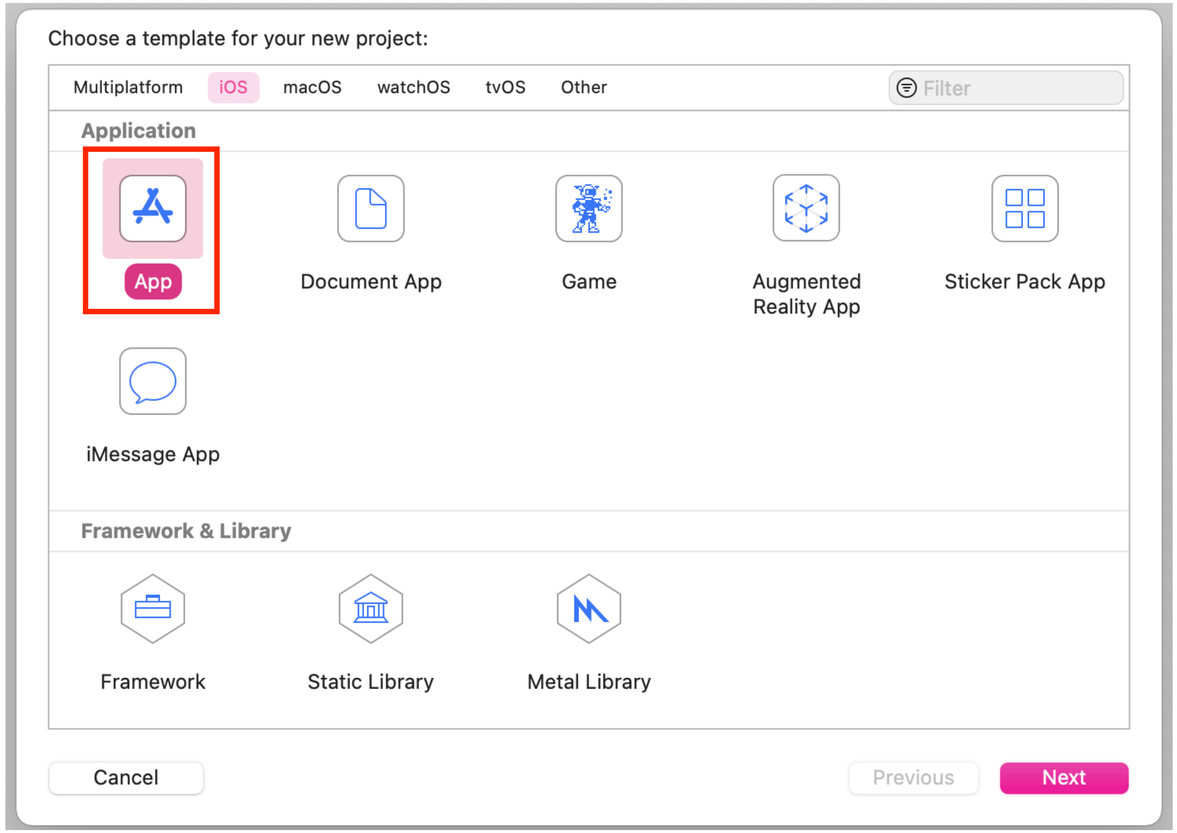
Xcode에서 프로젝트 만들기

빨간색으로 표시된 부분을 순서대로 눌러줍니다.



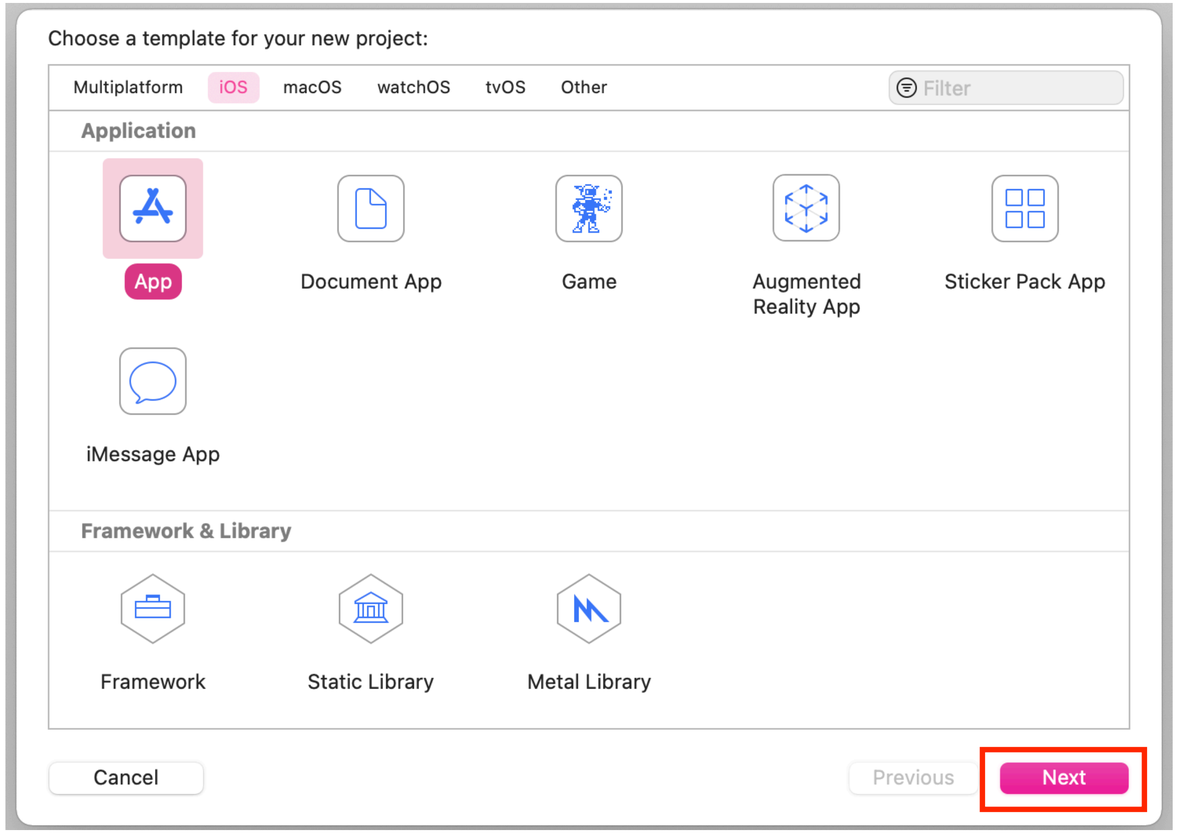
Xcode에서 프로젝트 만들기

빨간색으로 표시된 부분을 순서대로 눌러줍니다.



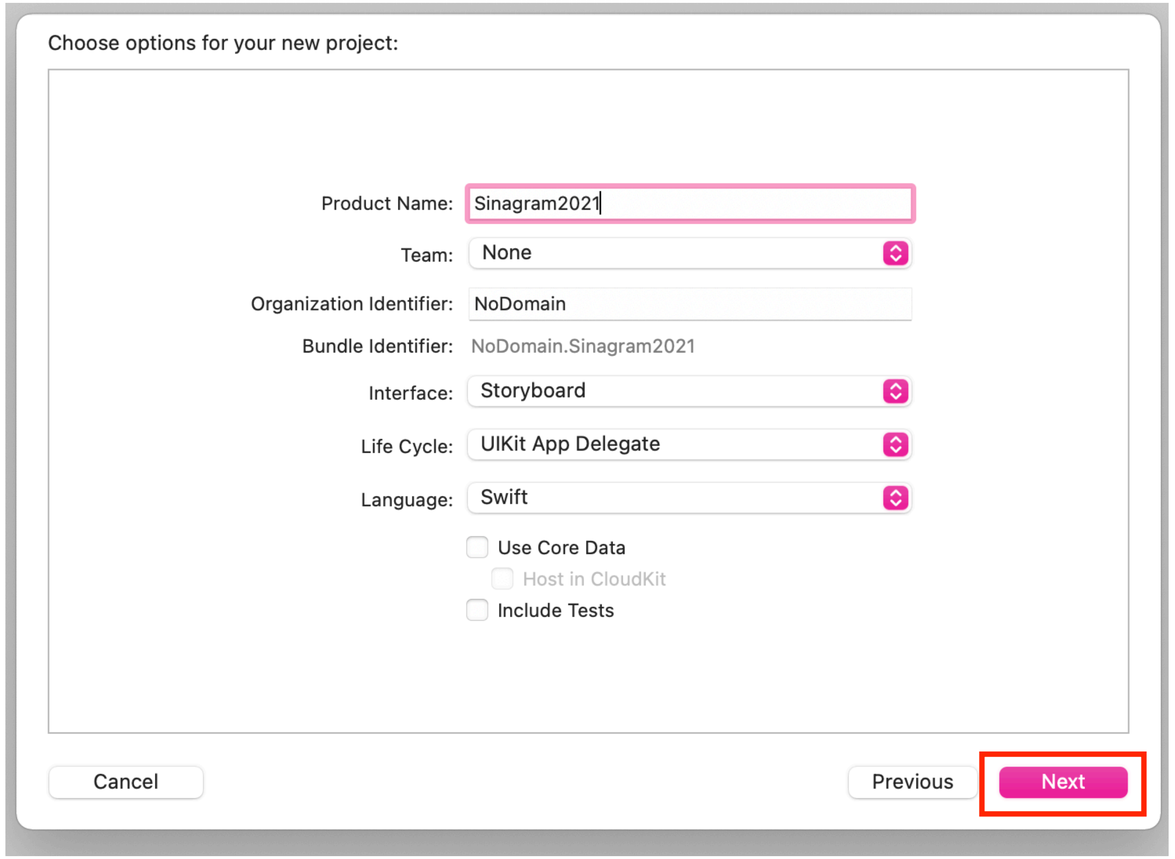
Xcode에서 프로젝트 만들기

빨간색으로 표시된 부분을 순서대로 눌러줍니다.



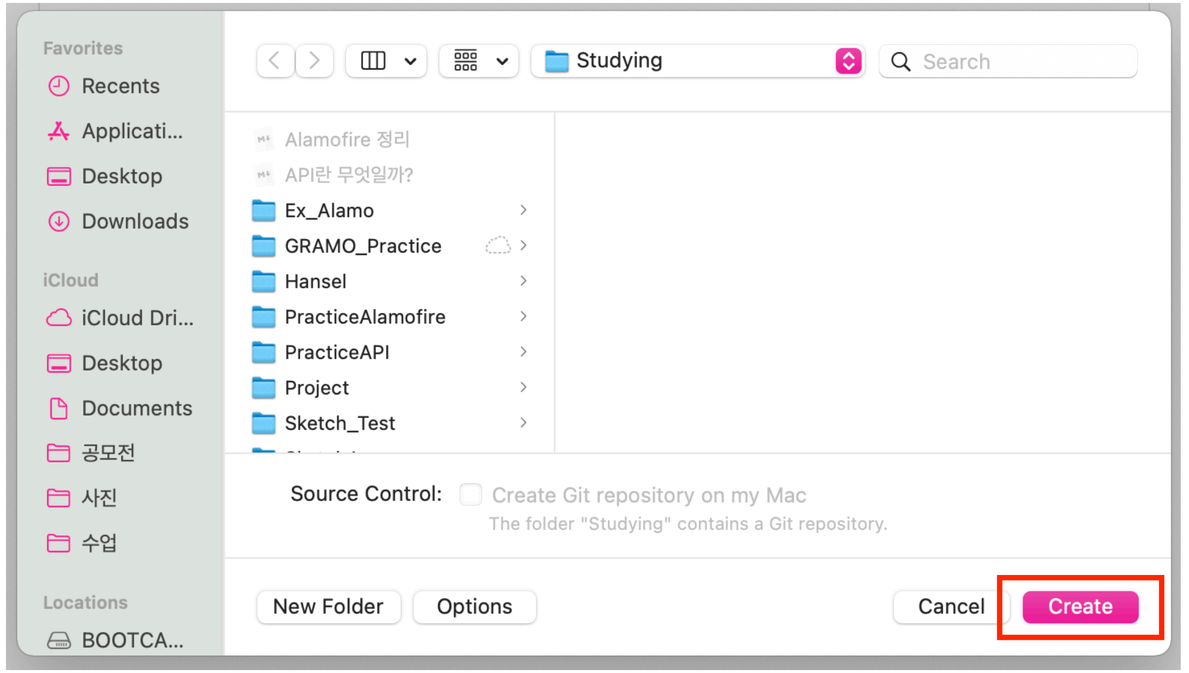
Xcode에서 프로젝트 만들기

Product Name 칸에 프로젝트 이름을 작성한 뒤, 빨간색으로 표시된 버튼을 눌러줍니다.



Xcode에서 프로젝트 만들기

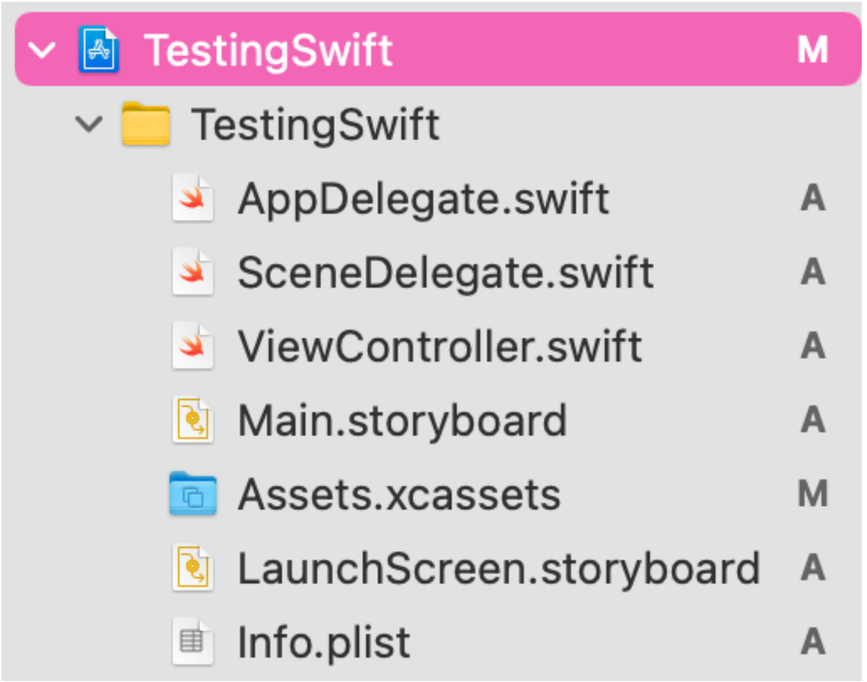
마지막 부분입니다.
프로젝트를 저장할 공간을 정한 뒤,
마찬가지로 빨간색으로 표시된
Create 버튼을 눌러
프로젝트를 만들어줍니다.



신입생 분들은 Desktop(바탕화면)에 만들어 주시면 감사하겠습니다.

Xcode 프로젝트

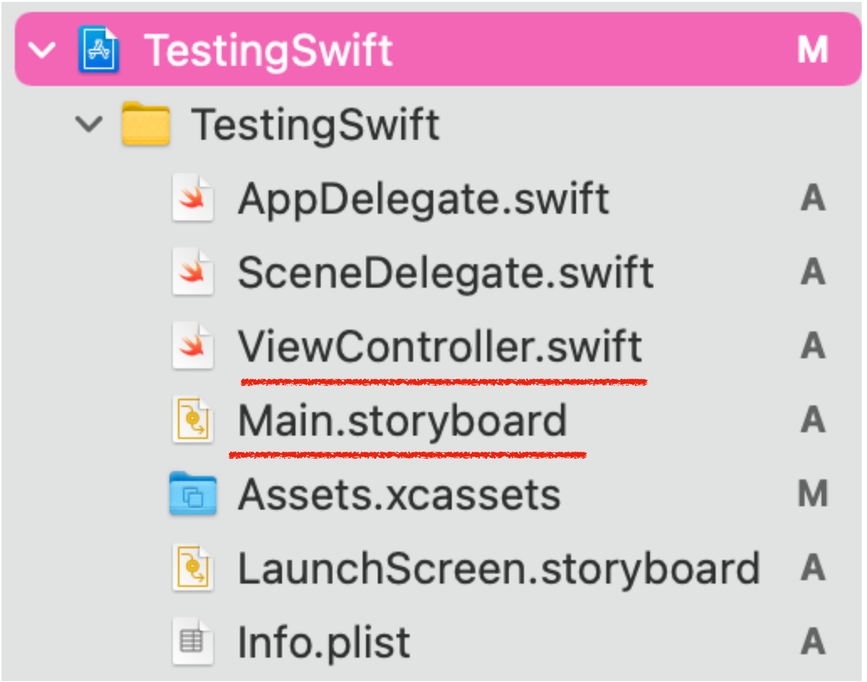
기본 파일구조



위의 TestingSwift이라는 파일이 없으신 것은 이 파일의 이름이 곧 프로젝트 이름이기 때문에 당연히 이름이 다를 수 밖에 없습니다.

Xcode 프로젝트

기본 파일구조

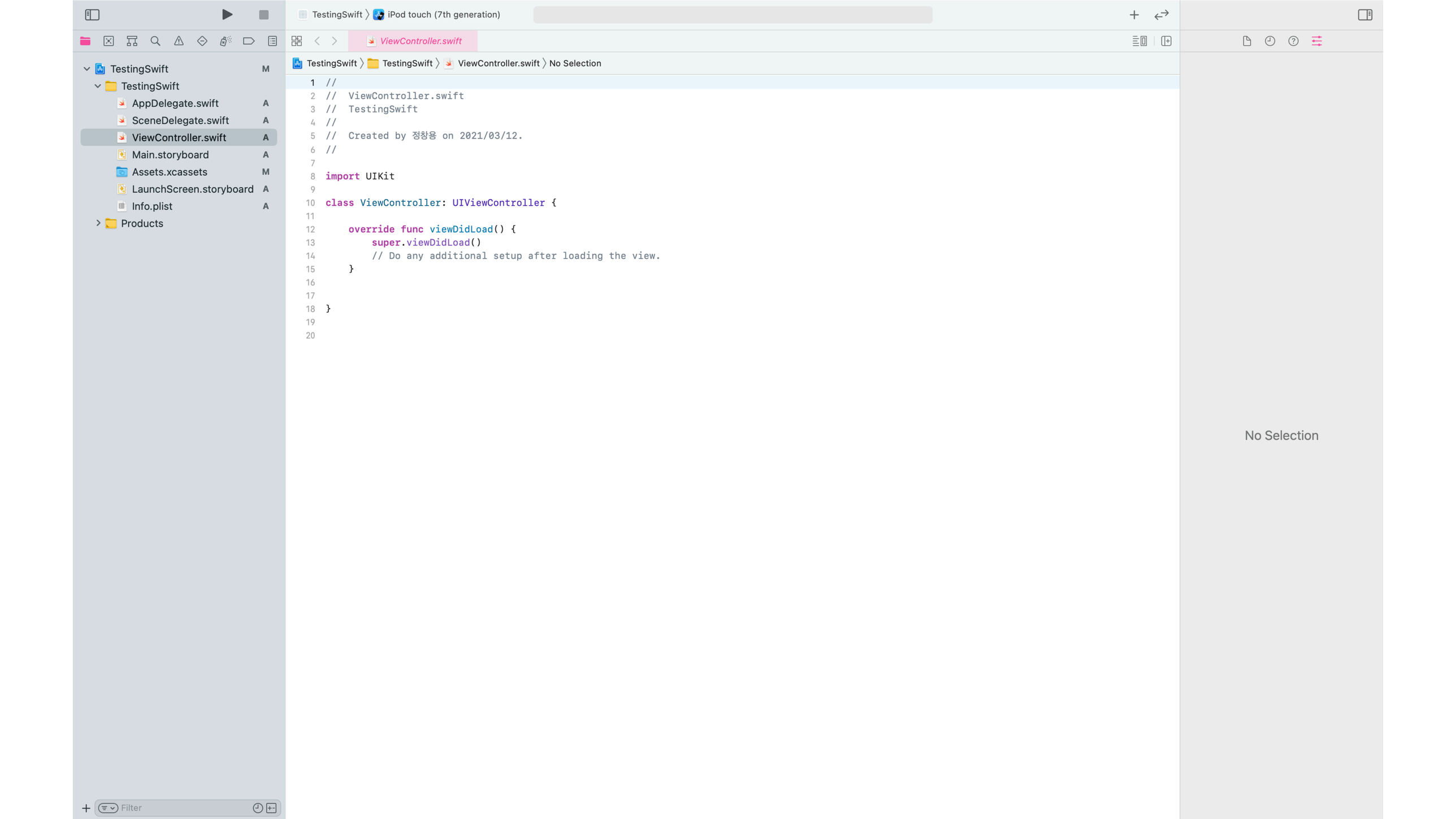


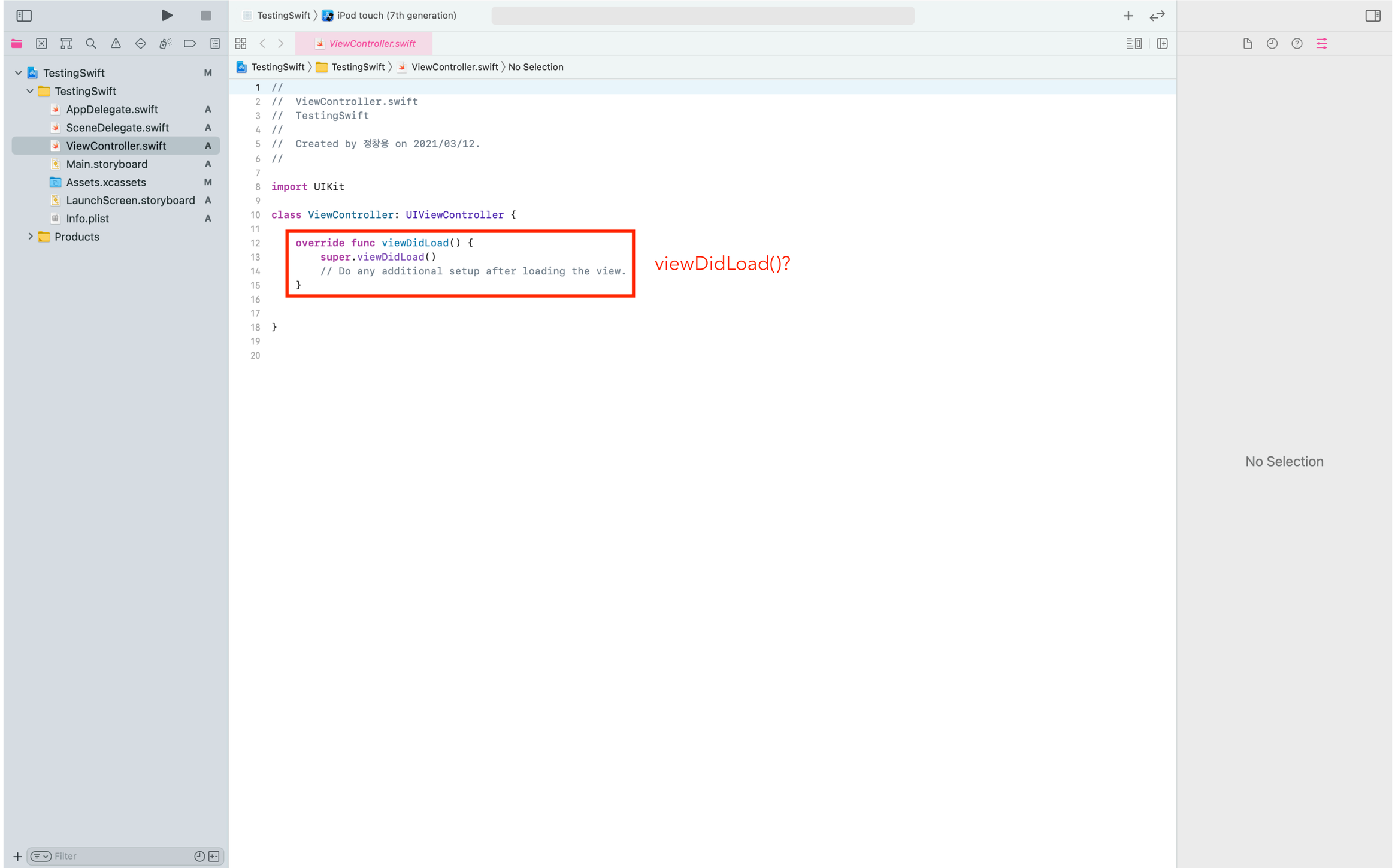
ViewController.swift?

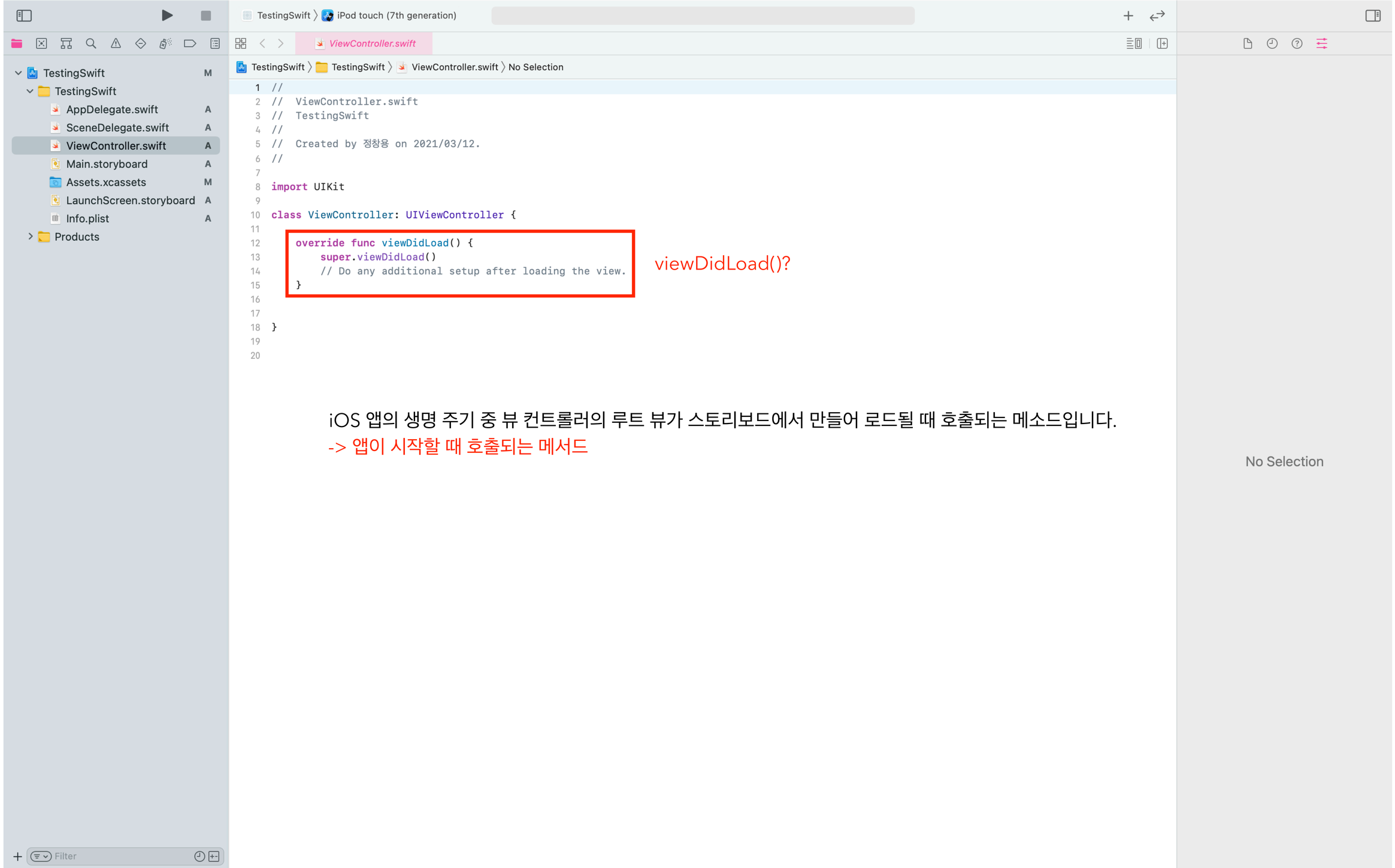
ViewController.swift는 어떤 역할을 할까?

- UIKit을 사용하는 앱의 인터페이스를 관리하기 위한 도구입니다.
- 뷰 컨트롤러는 하나의 루트 뷰 만을 관리하고, 해당 루트 뷰가 여러 개의 서브 뷰들을 가지는 방식으로 한 화면을 여러 개의 뷰로 구성할 수 있습니다.
- 뷰들과 함께 사용자와의 대화에 응답한다.
- 뷰들의 사이즈 재조정과 전반적인 인터페이스의 레이아웃을 관리한다.
- 다른 객체(뷰 컨트롤러 등)들과 함께 앱을 구성한다.
- -> 앱의 인터페이스를 관리하기 위한 프로젝트 핵심 파일

뷰(View)란? - iOS 앱 사용자 인터페이스의 기본 구성 요소이다.


TestingSwift
iPod touch (7th generation)
ViewController.swift
TestingSwift
TestingSwift
TestingSwift
AppDelegate.swift A
SceneDelegate.swift A
ViewController.swift A
Main.storyboard A
Assets.xcassets M
LaunchScreen.storyboard A
Info.plist A
Products
TestingSwift
TestingSwift
ViewController.swift
No Selection
1 //
2 // ViewController.swift
3 // TestingSwift
4 //
5 // Created by 정창용 on 2021/03/12.
6 //
7
8 import UIKit
9
10 class ViewController: UIViewController {
11
12 override func viewDidLoad() {
13 super.viewDidLoad()
14 // Do any additional setup after loading the view.
15 }
16
17
18 }
19
20
Filter
No Selection


TestingSwift
iPod touch (7th generation)
ViewController.swift
TestingSwift
TestingSwift
TestingSwift
AppDelegate.swift
SceneDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
TestingSwift
TestingSwift
ViewController.swift
No Selection
1 //
2 // ViewController.swift
3 // TestingSwift
4 //
5 // Created by 정창용 on 2021/03/12.
6 //
7
8 import UIKit
9
10 class ViewController: UIViewController {
11
12 override func viewDidLoad() {
13 super.viewDidLoad()
14 // Do any additional setup after loading the view.
15 }
16
17
18 }
19
20
viewDidLoad()?
No Selection
Filter


TestingSwift
iPod touch (7th generation)
ViewController.swift
TestingSwift
TestingSwift
TestingSwift
AppDelegate.swift
SceneDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
TestingSwift
TestingSwift
ViewController.swift
No Selection
//
// ViewController.swift
// TestingSwift
//
// Created by 정창용 on 2021/03/12.
//
import UIKit
class ViewController: UIViewController {
override func viewDidLoad() {
super.viewDidLoad()
// Do any additional setup after loading the view.
}
}
viewDidLoad()?
iOS 앱의 생명 주기 중 뷰 컨트롤러의 루트 뷰가 스토리보드에서 만들어 로드될 때 호출되는 메소드입니다.
-> 앱이 시작할 때 호출되는 메서드
No Selection
Filter

Main.storyboard?

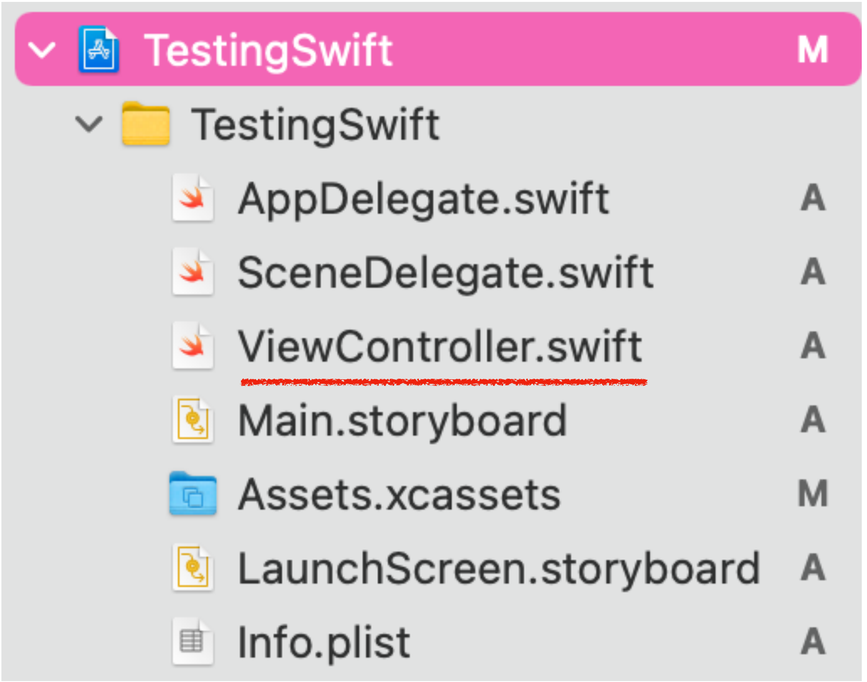
Main.storyboard는 어떤 역할을 할까?

- 스토리보드는 iOS 앱의 사용자 인터페이스를 시각적으로 표현하여 컨텐츠 화면과 화면 간의 연결을 보여주는 도구입니다.
- 스토리보드는 일련의 장면들(scenes)로 구성되며, 각 scene은 뷰 컨트롤러와 해당 뷰를 나타냅니다.
- -> 앱의 사용자 인터페이스를 시각적으로 표현하는 프로젝트 핵심 파일

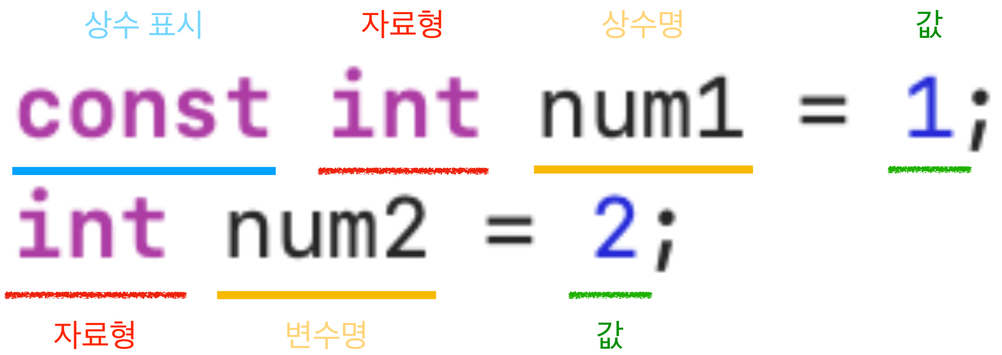
문법

지금부터 문법에 대하여 알아보는 시간을 가지도록 하겠습니다

Xcode 왼쪽 창에서 방금 알아본
ViewController.swift 파일에
들어가보도록 하겠습니다.



상수, 변수



- 신입생 분들이 배우고 있는
- C의 상수, 변수 선언 문법입니다.

상수, 변수

```
let num1: Int = 1
var num2: Int = 2
```

- Swift의 기본 상수, 변수 선언 문법입니다.

상수, 변수

```
let num1: Int = 1
var num2: Int = 2
```

- 앞의 let, var에 따라서 상수인지 변수인지가 결정됩니다.
- 상수 선언 키워드 let
- 변수 선언 키워드 var

상수, 변수

```
let num1: Int = 1
var num2: Int = 2
```

- 다음은 상수, 변수의 이름입니다.

상수, 변수

```
let num1: Int = 1
var num2: Int = 2
```

- 그리고 콜론을 써주고

상수, 변수

```
let num1: Int = 1
var num2: Int = 2
```

- 타입을 써줍니다.
- 값의 타입이 명확하다면 타입은 생략이 가능합니다.

상수, 변수

```
let num1 = 1
var num2 = 2
```

- 이렇게요!

상수, 변수

```
let num1: Int = 1
var num2: Int = 2
```

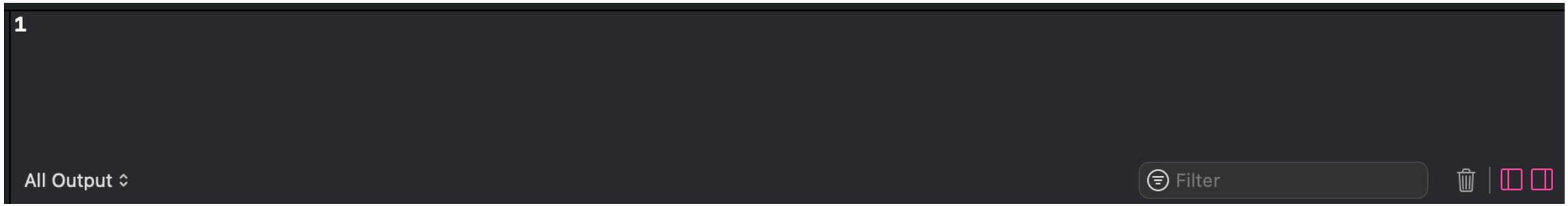
- 마지막으로 값을 넣어줍니다.
- 이렇게 상수, 변수 선언은 끝입니다.

출력문

```
print(num1, num2)
```

- Swift의 출력문은 C와 다르게 printf()가 아닌
- print()입니다.

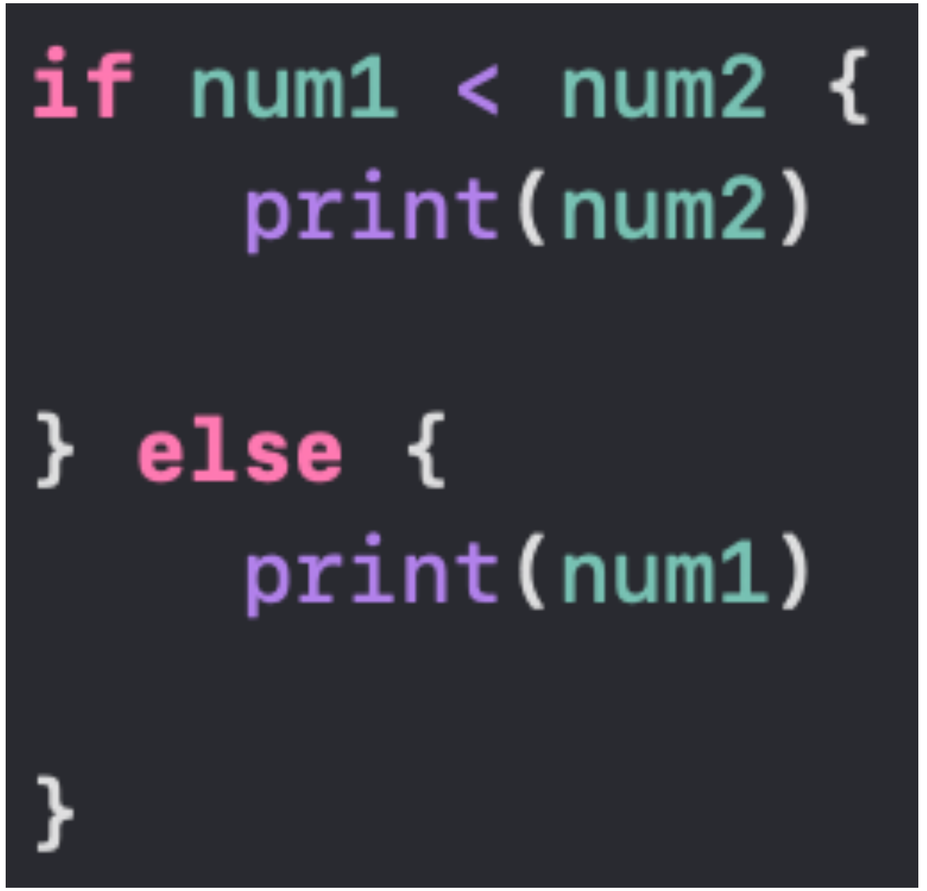
출력문



- 실행을 시켜보았을 때,
- 출력이 잘 되는 것을 볼 수 있습니다.

if

- 기본적인 if 문의 문법입니다.
- C와 다른 점이라면 if 문 옆 괄호를 생략할 수 있습니다.



함수

```swift
func plusNum(num: Int) -> Int {
    return num + 1

}
```

- 함수의 기본 형식입니다.

함수

```
func plusNum(num: Int) -> Int {
    return num + 1


}
```

- 함수를 선언할 때에는
- Function의 약자인
- Func를 붙여줍니다.

함수

```
func plusNum(num: Int) -> Int {
    return num + 1


}
```

- 함수명을 설정해줍니다.
- 함수명은 자신이 원하시는 대로 써주셔도 됩니다.

함수

```
func plusNum(num: Int) -> Int {
    return num + 1

}
```

- 매개변수를 선언해줍니다.
- num은 매개변수의 이름,
- Int는 매개변수의 타입입니다.
- 매개변수가 필요하지 않을 때에는 선언해주지 않으셔도 됩니다.

함수

```
func plusNum(num: Int) -> Int {
    return num + 1

}
```

- 다음으로는 반환값을 설정해줍니다.
- -> 를 써주신 다음 반환 타입을 써주시면 됩니다.
- 반환값이 없을 때에는 쓰지 않을 수도 있습니다.

함수

```
func plusNum(num: Int) -> Int {
    return num + 1

}
```

- return 함수를 이용하여
- 반환해줍니다.
- 반환 값이 없는 함수는 return 함수를 쓰지 않습니다.

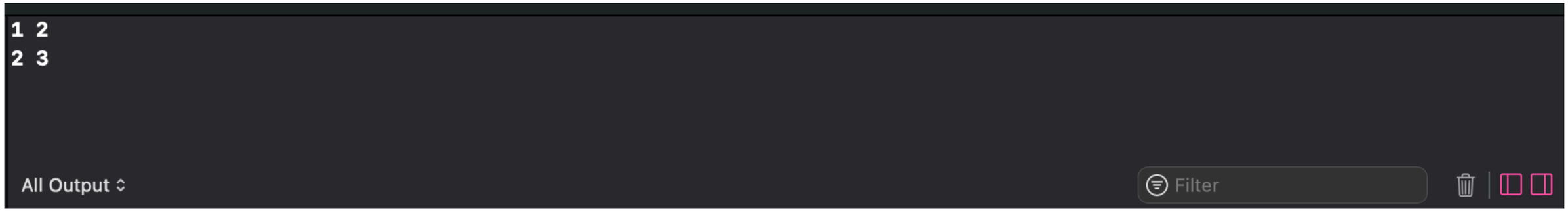
함수

```
print(num1, num2)

print(plusNum(num: num1), plusNum(num: num2))
```

- 이제 viewDidLoad() 함수에 코드를 써서
- 함수가 잘 실행되는지 확인합니다.

함수



- 이렇게 출력이 되었다면 성공입니다!

옵셔널(Optional)?

옵셔널은 무슨 역할을 할까?

- Swift가 갖는 옵셔널이라는 개념은 변수의 값이 nil 일 수 있다는 것을 표현하는 것입니다. 반대로 옵셔널이 아니라면(non-optional) 해당 값은 nil이 될 수 없음을 의미합니다.

- Swift에서는 옵셔널은 말 그대로
- 옵션(선택적)이며 기본값은 non-Optional입니다.

옵셔널의 키워드

옵셔널(Optional)?

옵셔널은 무슨 역할을 할까?

옵셔널 변수의 선언은 ? 키워드를 사용합니다.

```swift
var name: String?

```

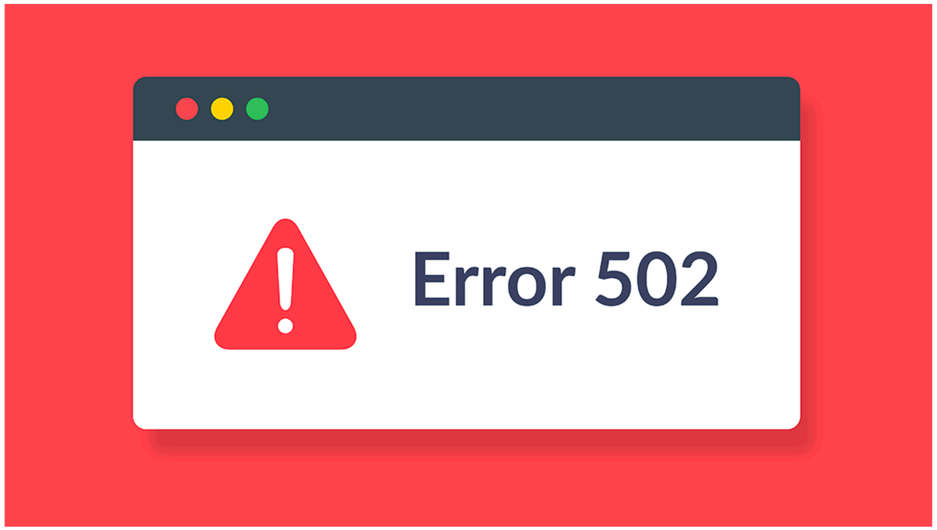
옵셔널의 디폴트 값은 nil로 name은 nil을 갖게 됩니다.

옵셔널(Optional)?

옵셔널은 무슨 역할을 할까?

- 만약 옵셔널 키워드를 사용하지 않았다면 값을 입력하라는 에러가 발생하고, 그 이후에라도 nil을 넣으려하면 컴파일에러가 발생합니다.

- nil에 대한 컴파일 에러를 통해 개발자는 nil에 대해 명확한 예외처리가 강제되며, 런타임에 nil로 인한 문제를 컴파일 단계에서 예방할 수 있습니다.



Swift가 잠재적 오류에 대해 안전하다고 표현하는 글도 많았습니다.

옵셔널의 활약으로

SWIFT

```swift
var name: String // 컴파일에러
var name = nil // 컴파일에러

```

감사합니다

지금까지 버텨주신 모든 분들께