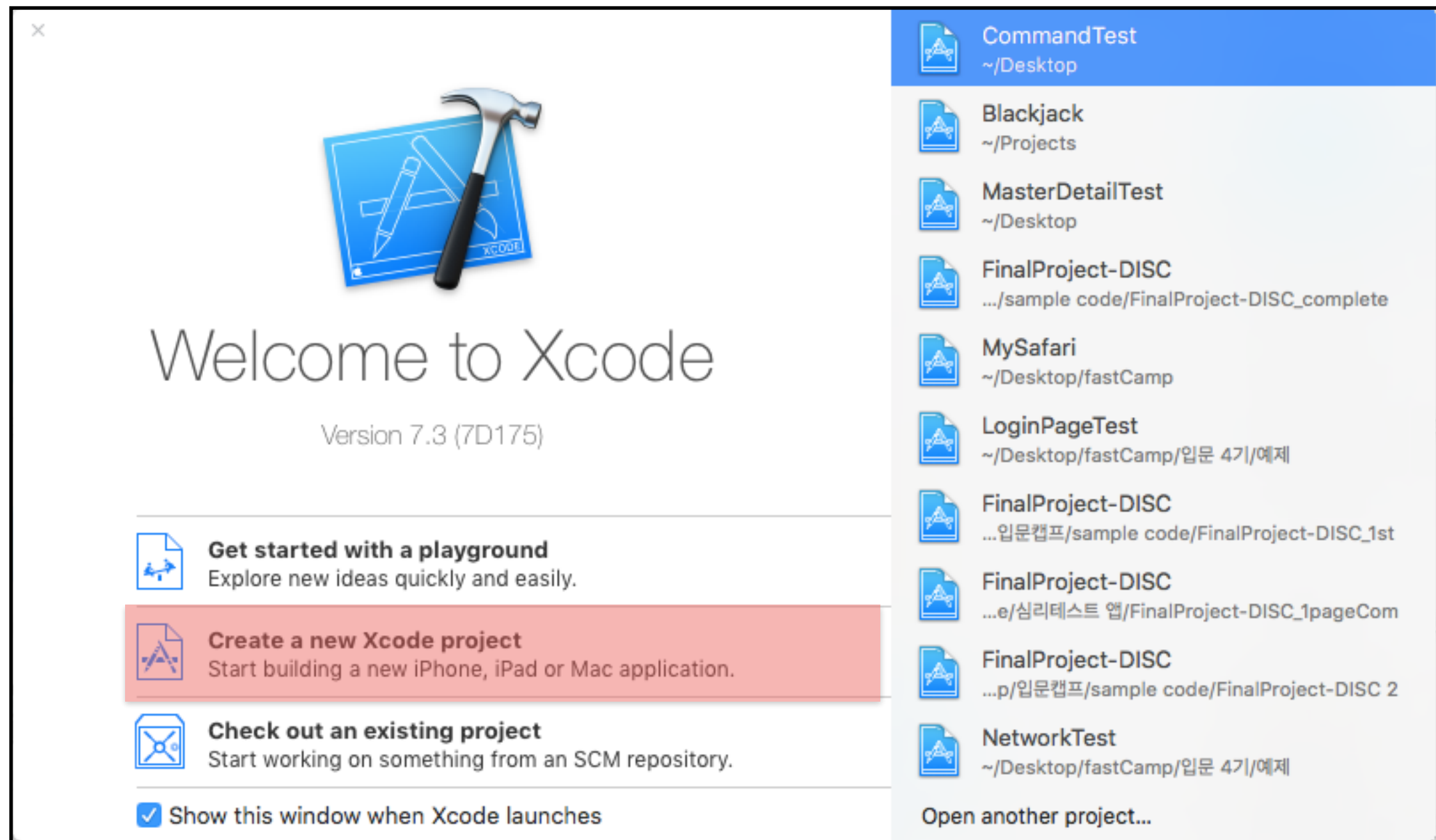
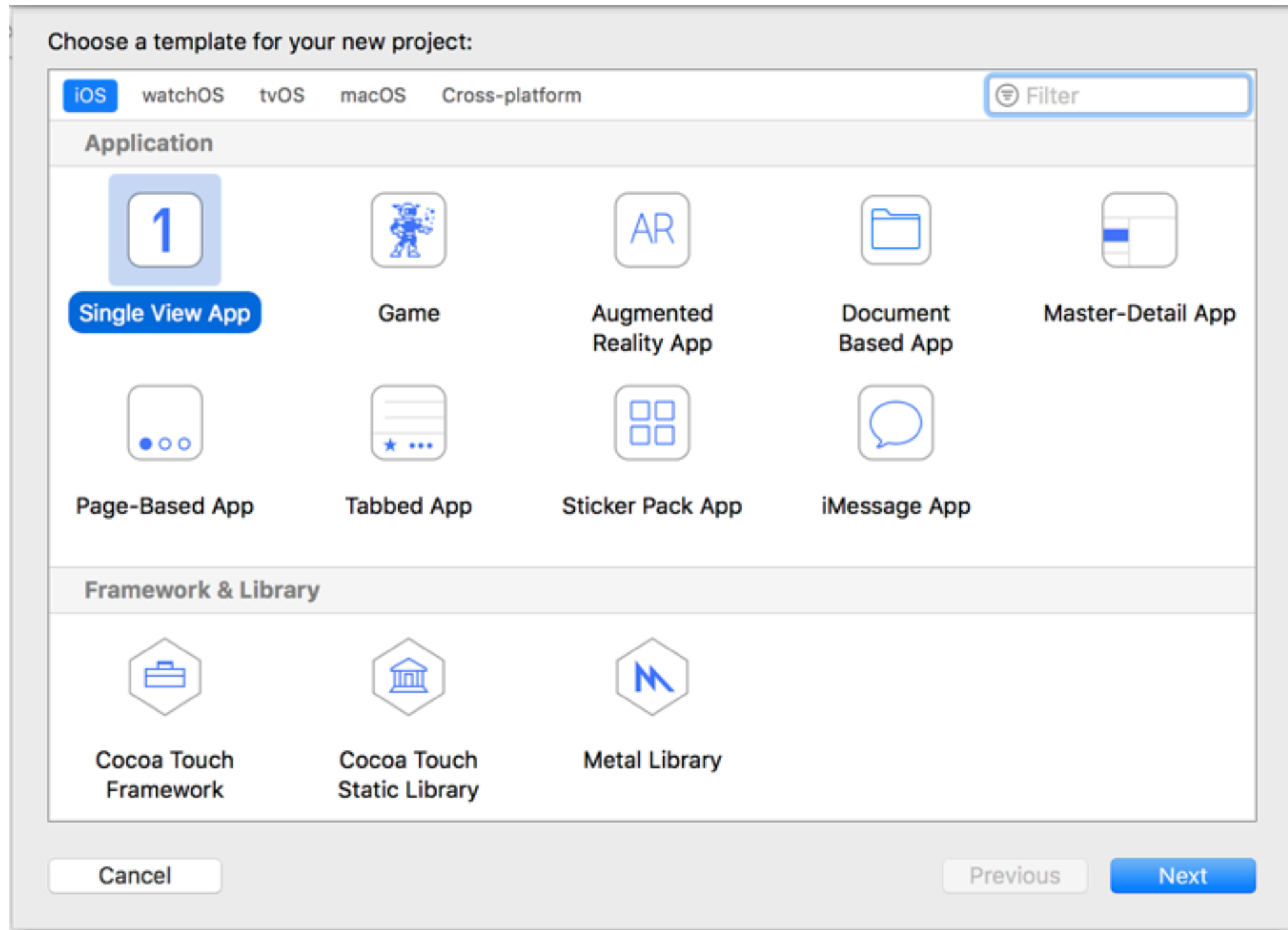

Xcode 사용법

Xcode - 시작



템플릿 선택



프로젝트 생성

Choose options for your new project:

Product Name:

Team:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

Toolbar

Navigator

Editor

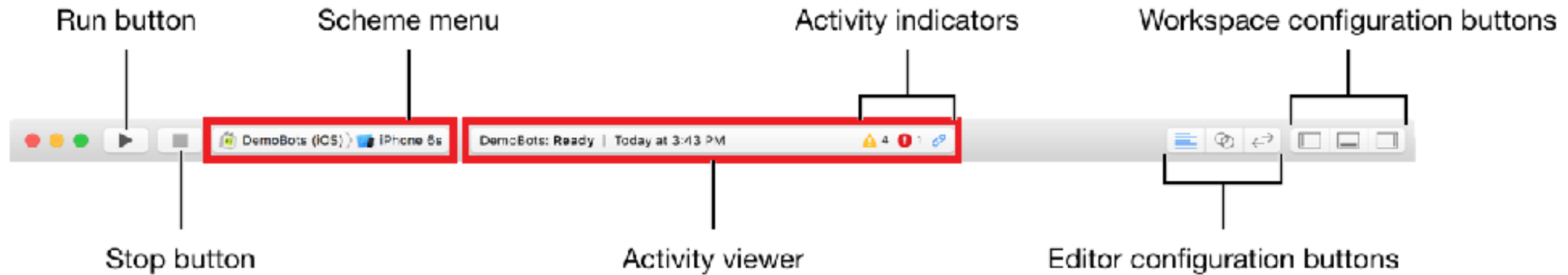
Debug Area

Utilities

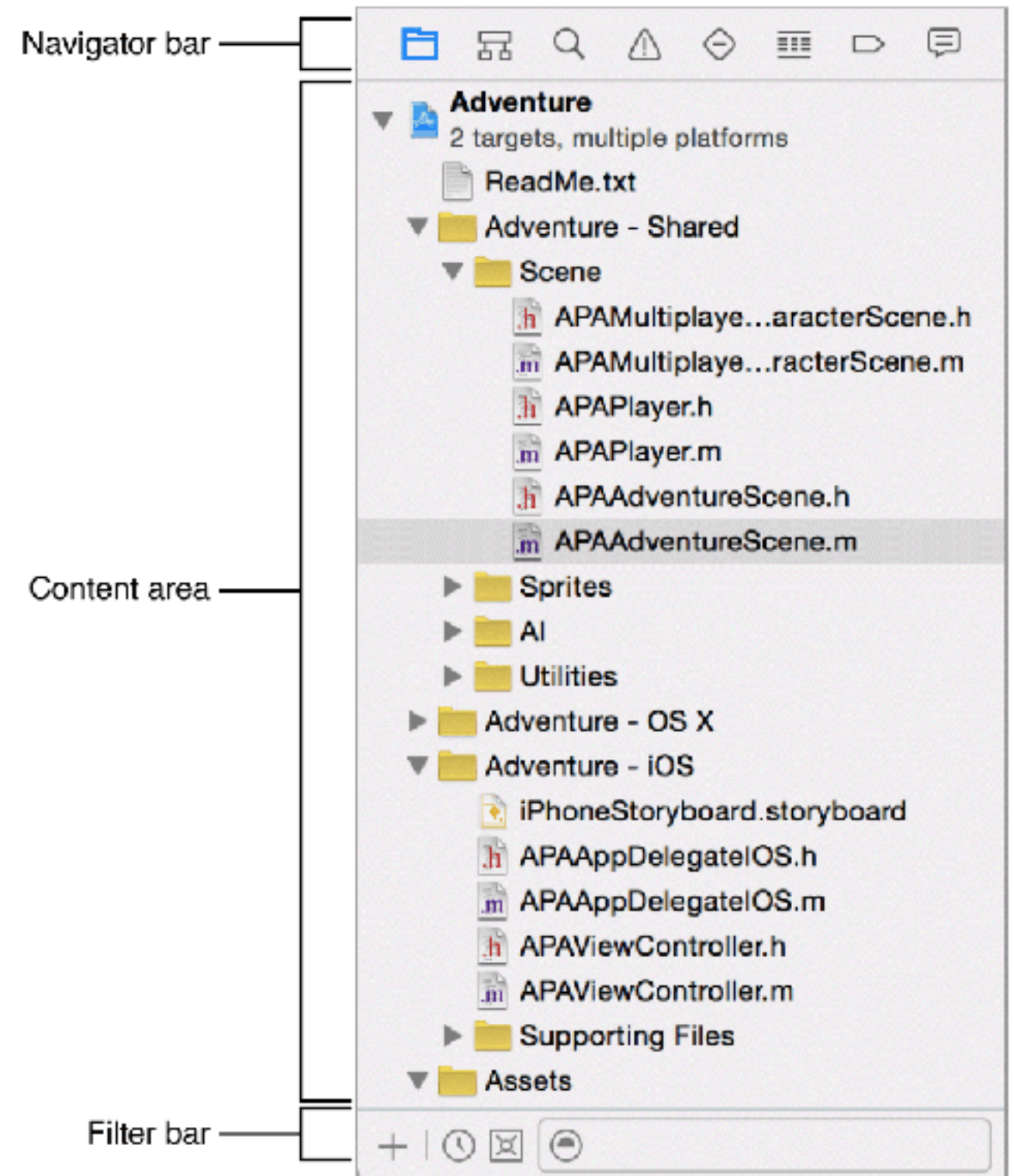
Xcode Main Window

- Navigator : 프로젝트 관리를 위한 도구 모음
- Editor : Project Navigator에서 선택한 파일의 내용을 수정하는 화면
- Debug Area: 프로그램 실행 중 Debugging를 위한 콘솔창
- Utilities : Project Navigator에서 선택된 파일의 상세 정보 및 UI속성 수정등의 작업을 위한 공간

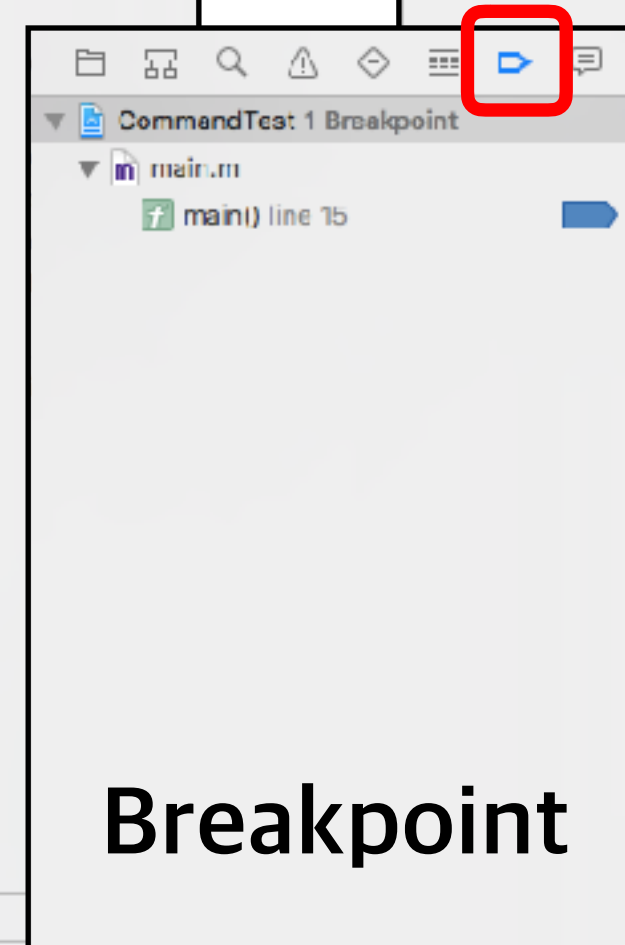
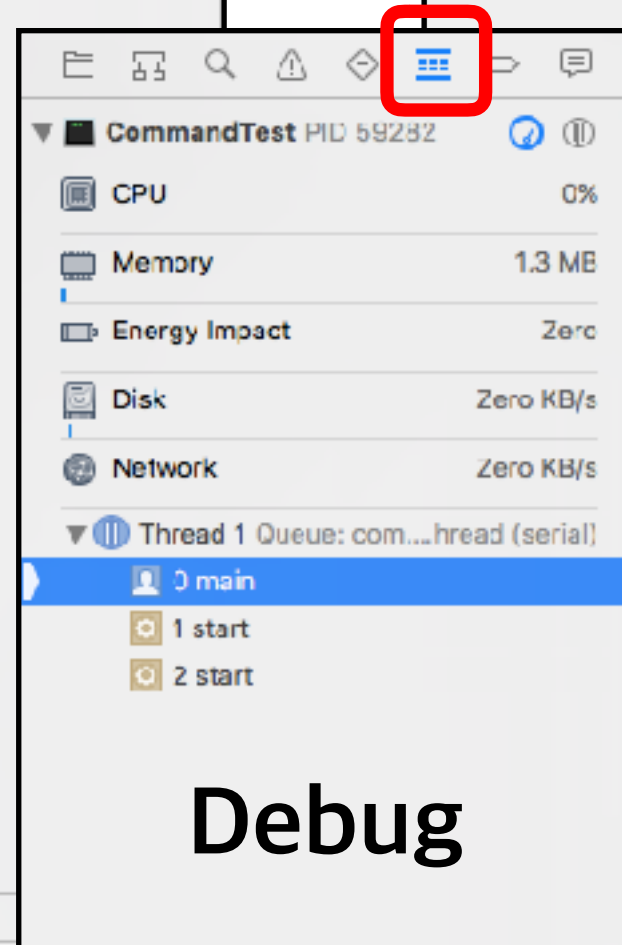
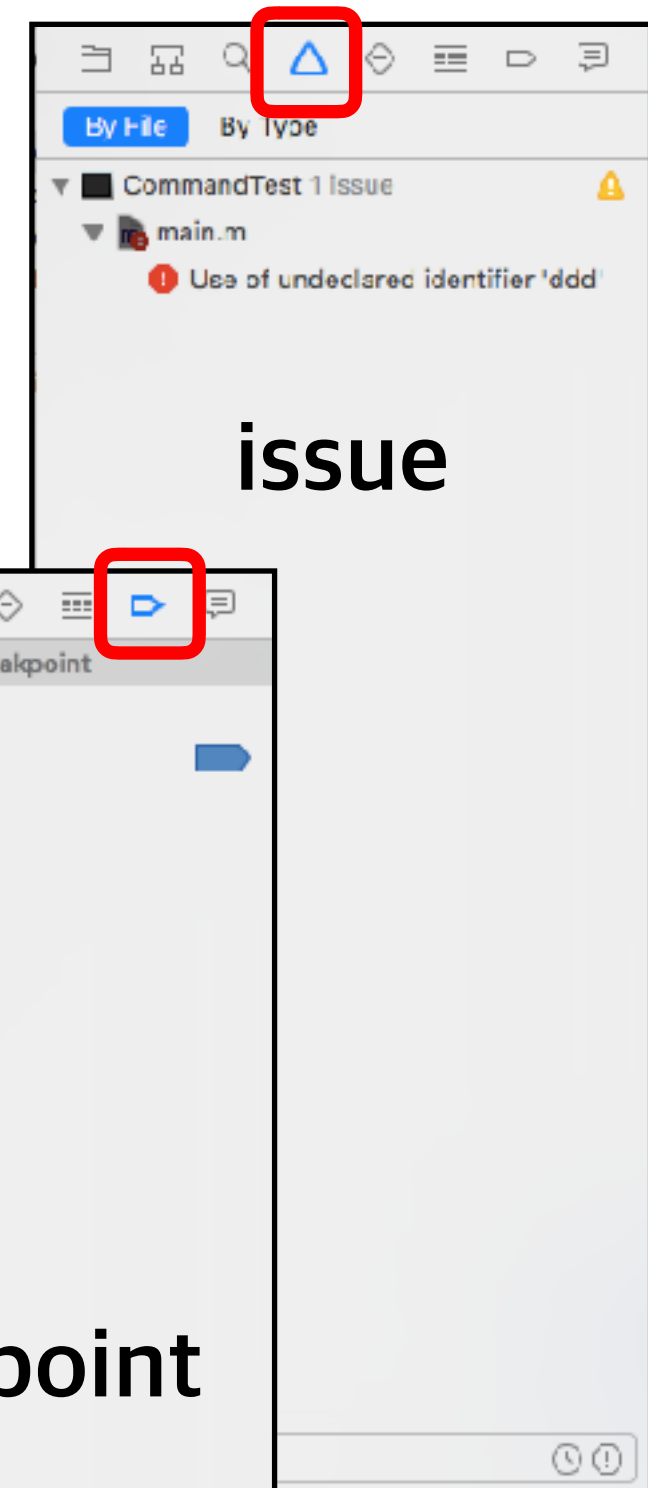
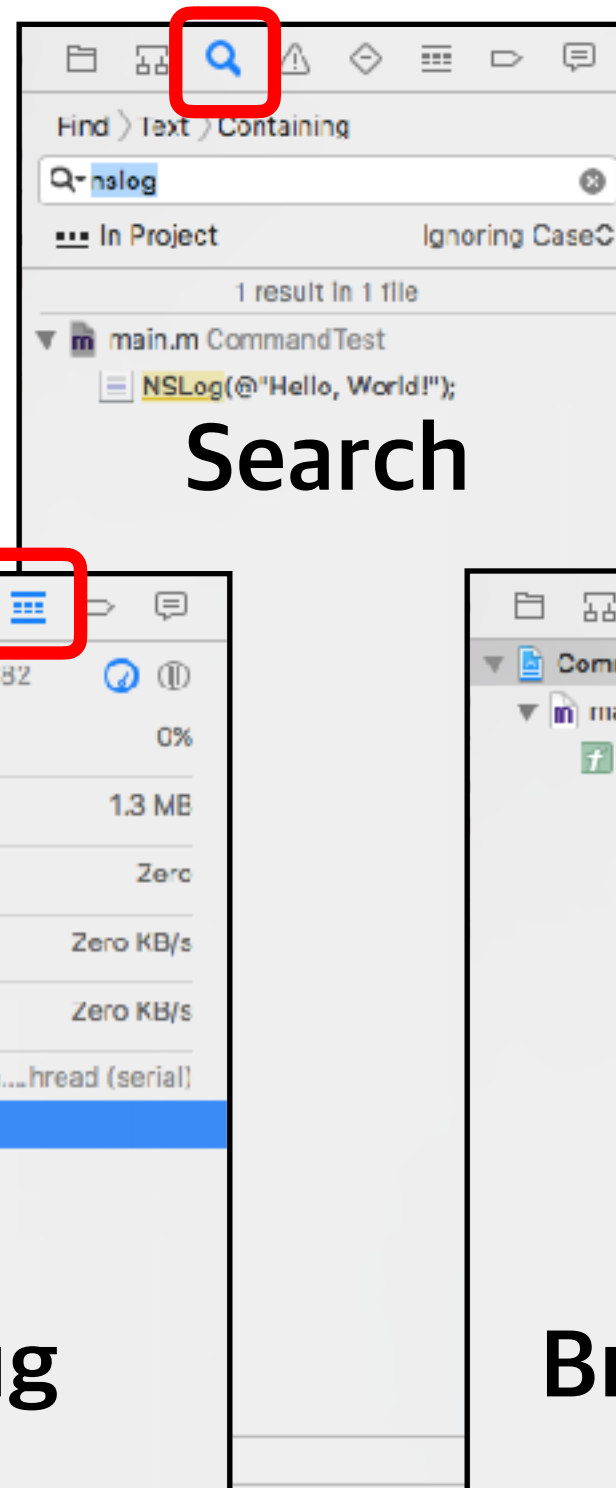
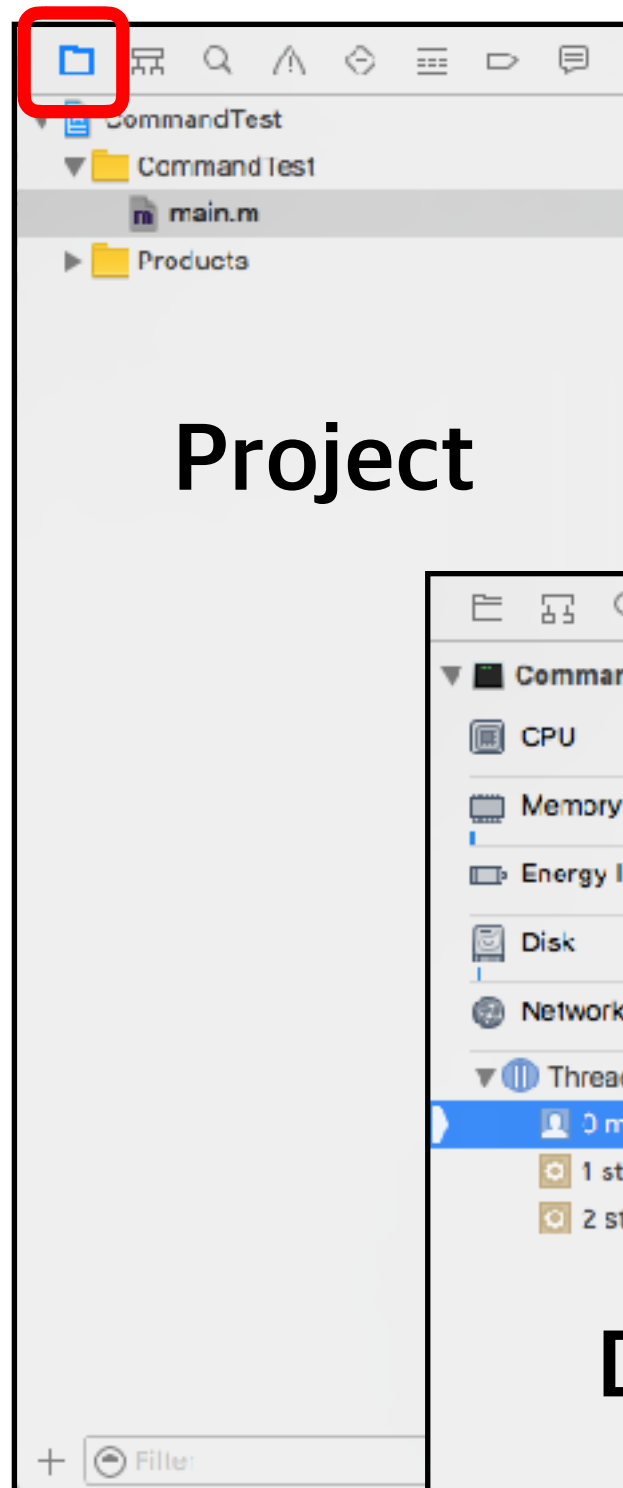
Toolbar



Navigator Area

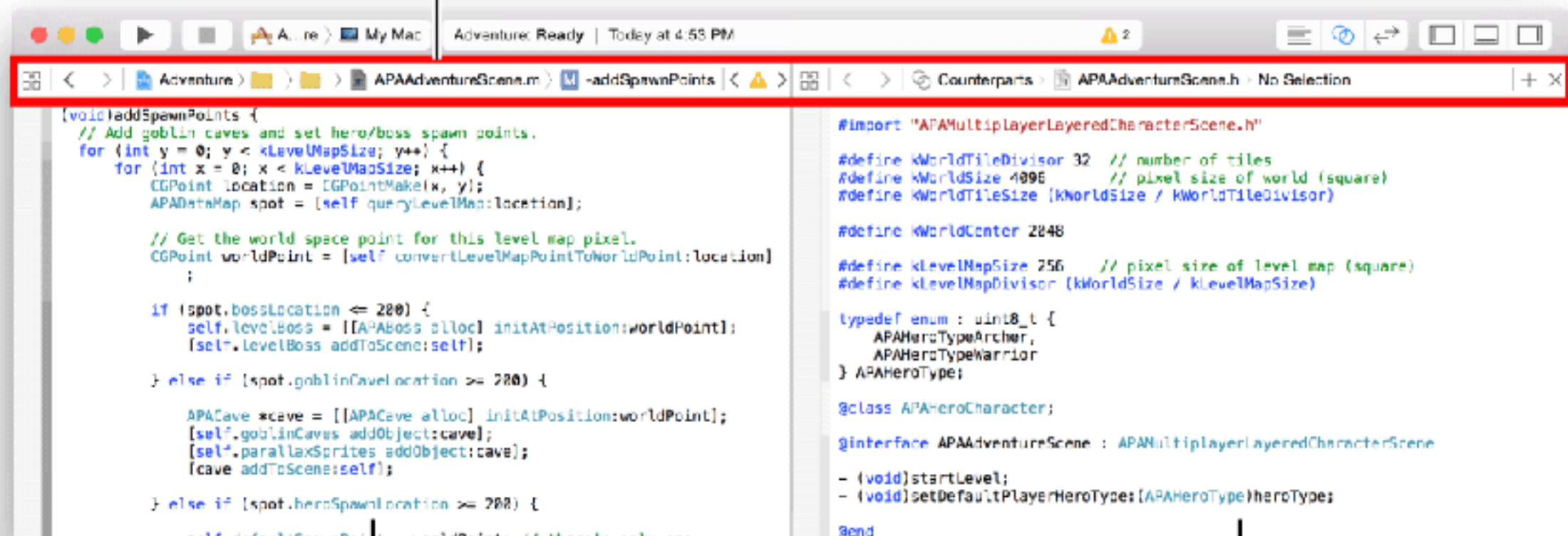


Navigator bar Menu



Editor




Jump bars



Standard editor pane

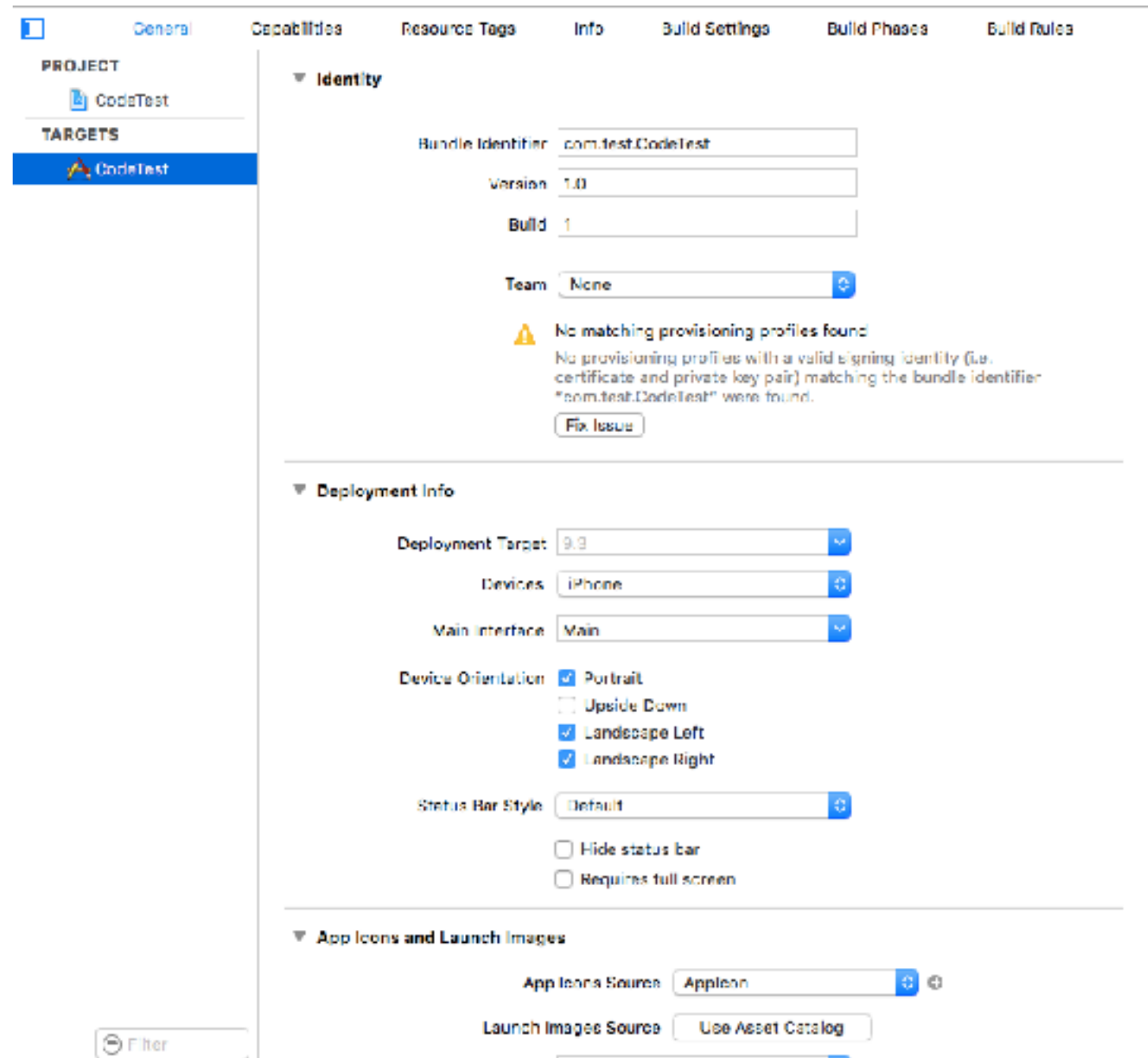
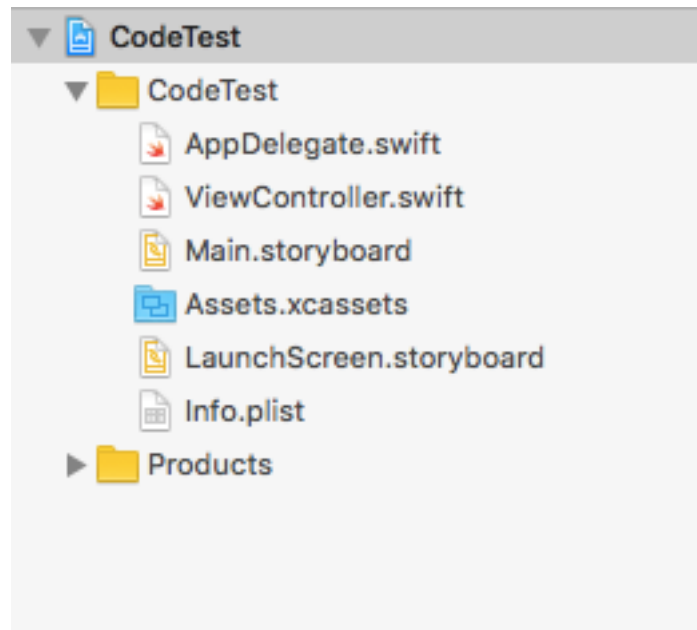
Assistant editor pane

Editor 상태편집

- Standard(): fills a single editor pane with the contents of the selected file.
- Assistant(): presents a separate editor pane with content logically related to that in the standard editor pane. Use the split controls in the
- Version (): shows the differences between the selected file in one pane and another version of that same file in a second pane.

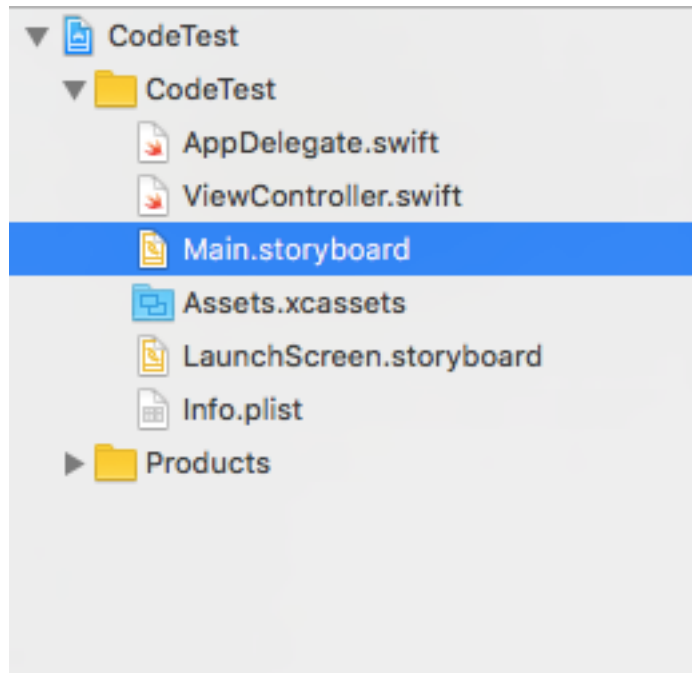
Project Editor

- 프로젝트 설정 변경



Source Editor

- 선택된 파일의 코드를 수정할 수 있다.



```
1 //
2 // ViewController.swift
3 // CodeTest
4 //
5 // Created by youngmin joo on 2016. 5. 3..
6 // Copyright © 2016년 WingsCompany. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         // Do any additional setup after loading the view, typically from a nib.
16     }
17
18     override func didReceiveMemoryWarning() {
19         super.didReceiveMemoryWarning()
20         // Dispose of any resources that can be recreated.
21     }
22
23 }
24
25
26
```

Break point

- debug를 위한 방법
- 여기서 실행을 멈춰라!

```
28 //프로그램
29 int main(int argc, const char * argv[]) {
30     @autoreleasepool {
31         // insert code here...
32
33         printf("여기에서 브레이크 포인트가 실행된다.");
34
35         printf("이코드는 아직 실행되지 않습니다.");
36
37         printf("다음 스텝을 눌러야 실행됩니다.");
38
39     }
40     return 0;
41 }
42
43
44
45
46
```

Thread 1: breakpoint 1.1

CommandTest > Thread 1 > 0 main

▸ A argv = (const char **) 0x7fff5fbff808
A argc = (int) 1

여기에서 브레이크 포인트가 실행된다. (lldb)

Break point

The screenshot shows a C program in a debugger. The code is as follows:

```
28 //프로그램
29 int main(int argc, const char * argv[]) {
30     @autoreleasepool {
31         // insert code here...
32
33         printf("여기에서 브레이크 포인트가 실행된다.");
34
35         printf("이코드는 아직 실행되지 않습니다.");
36
37         printf("이코드는 실행되었습니다.");
38
39     }
40
41     return 0;
42 }
43
44
45
46
```

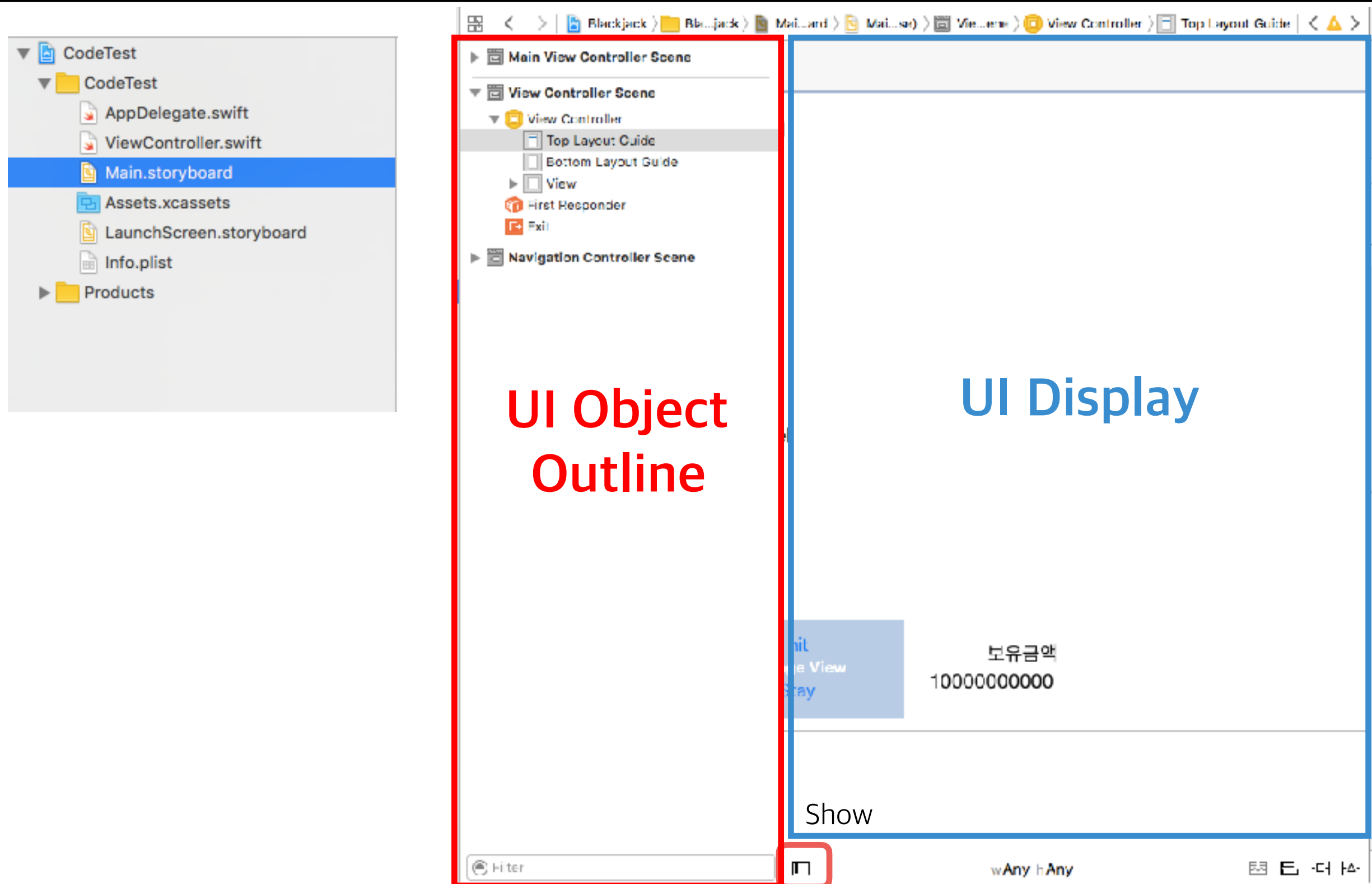
A breakpoint is set at line 35, indicated by a green bar and the text "Thread 1: breakpoint 1.1".

Red text labels with arrows point to the debugger's toolbar:

- break point enable** points to the blue play button icon.
- continue** points to the blue square icon with a right-pointing triangle.
- next Step** points to the blue icon with a right-pointing triangle and a horizontal line below it.

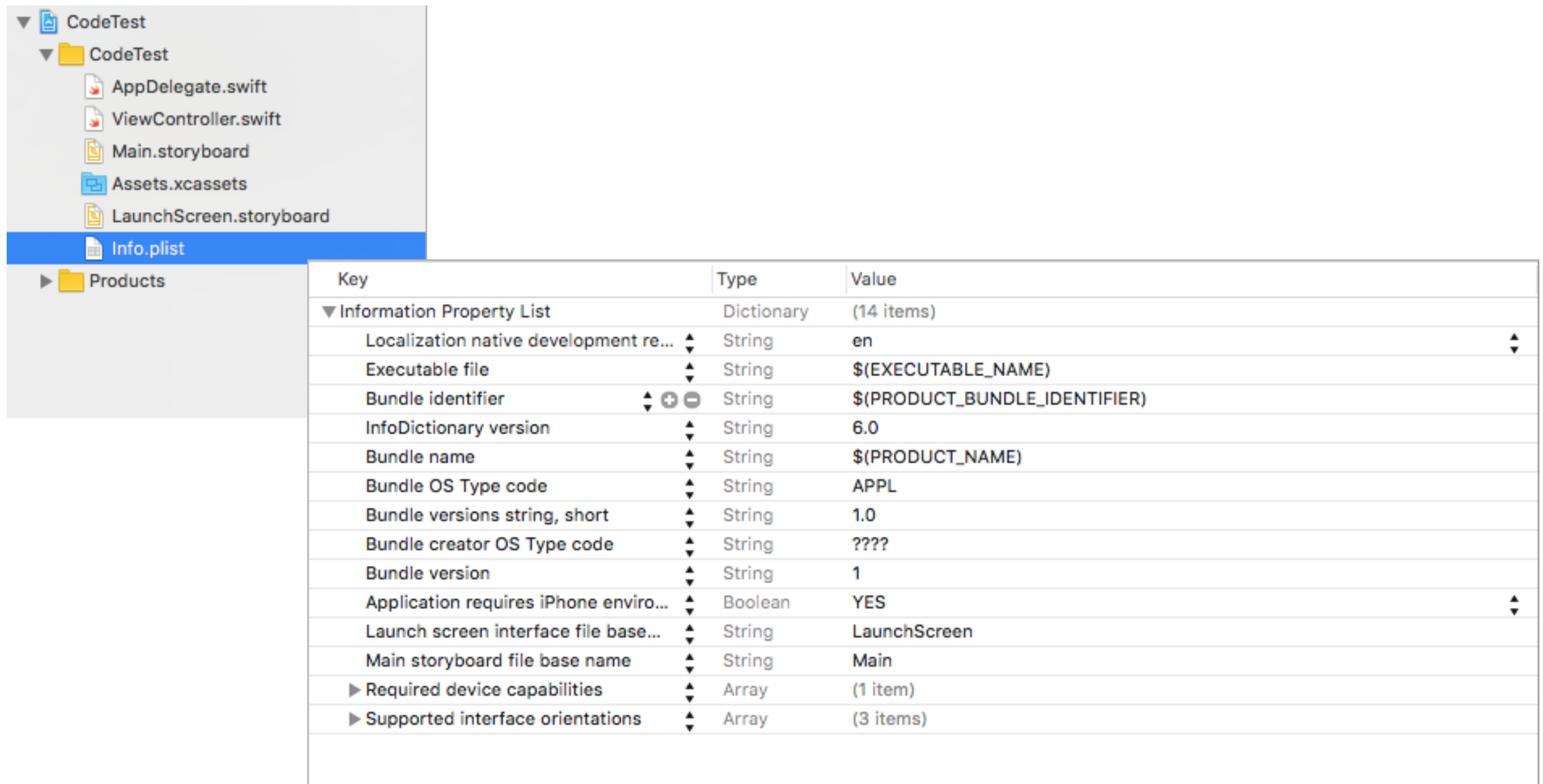
The debugger's status bar shows "CommandTest > Thread 1 > 0 main". The console output shows the message "여기에서 브레이크 포인트가 실행된다. (lldb)".

Interface Builder



Property list Editor

- property list(plist)파일 편집



Key	Type	Value
▼ Information Property List	Dictionary	(14 items)
Localization native development re...	String	en
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone enviro...	Boolean	YES
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(3 items)

Debug Area

Variables View

The screenshot shows the Xcode interface with the Debug Area open. The Variables View on the left displays two variables: `self` of type `(MainViewController *)` with address `0x7fbb43f2f8a0`, and `PI` of type `(const double)` with value `3.1400000000000001`. The Console on the right shows a log of card numbers being dealt in a Blackjack game, with a Korean message indicating a full house (버킹 창입니다.).

Variables View:

- `self` = (MainViewController *) 0x7fbb43f2f8a0
- `PI` = (const double) 3.1400000000000001

Console Output:

```
2016-04-21 13:35:57.179 Blackjack[59913:460018] card num : 11
2016-04-21 13:35:57.179 Blackjack[59913:460018] card num : 4
2016-04-21 13:35:57.180 Blackjack[59913:460018] card num : 10
2016-04-21 13:35:57.180 Blackjack[59913:460018] card num : 10
2016-04-21 13:35:57.181 Blackjack[59913:460018] card num : 5
2016-04-21 13:35:57.182 Blackjack[59913:460018] 여기는 디버깅 창입니다.
(lldb)
```

메모리 주소, 값 확인

Debug Area

Console

▶ self = (MainViewController *) 0x7fbb43f2f8a0

PI = (const double) 3.1400000000000001

log 출력,

명령어 직접 실행

Auto ↕ |

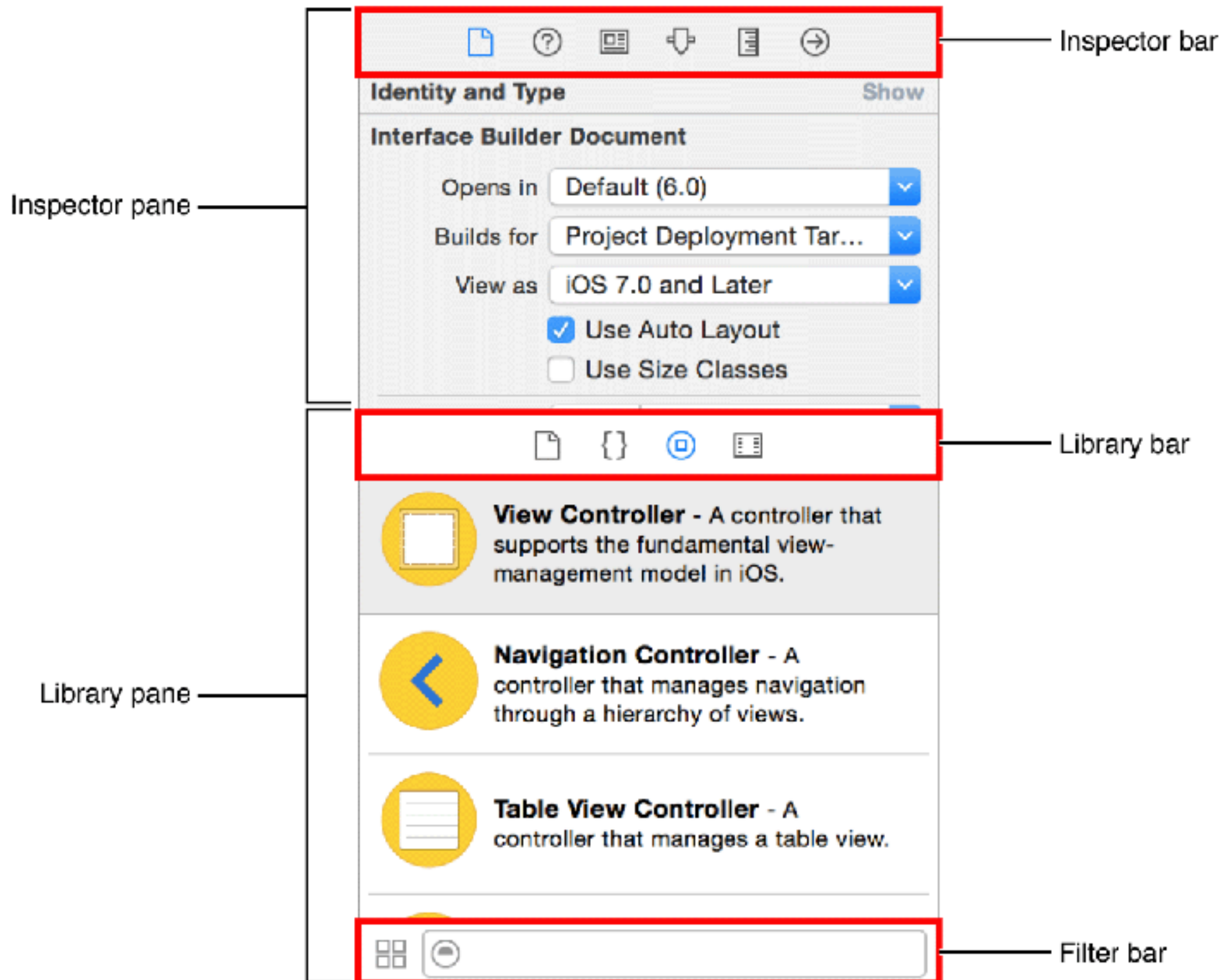
Filter

Blackjack > Thread 1 > 0 -[MainViewController viewDidLoad]

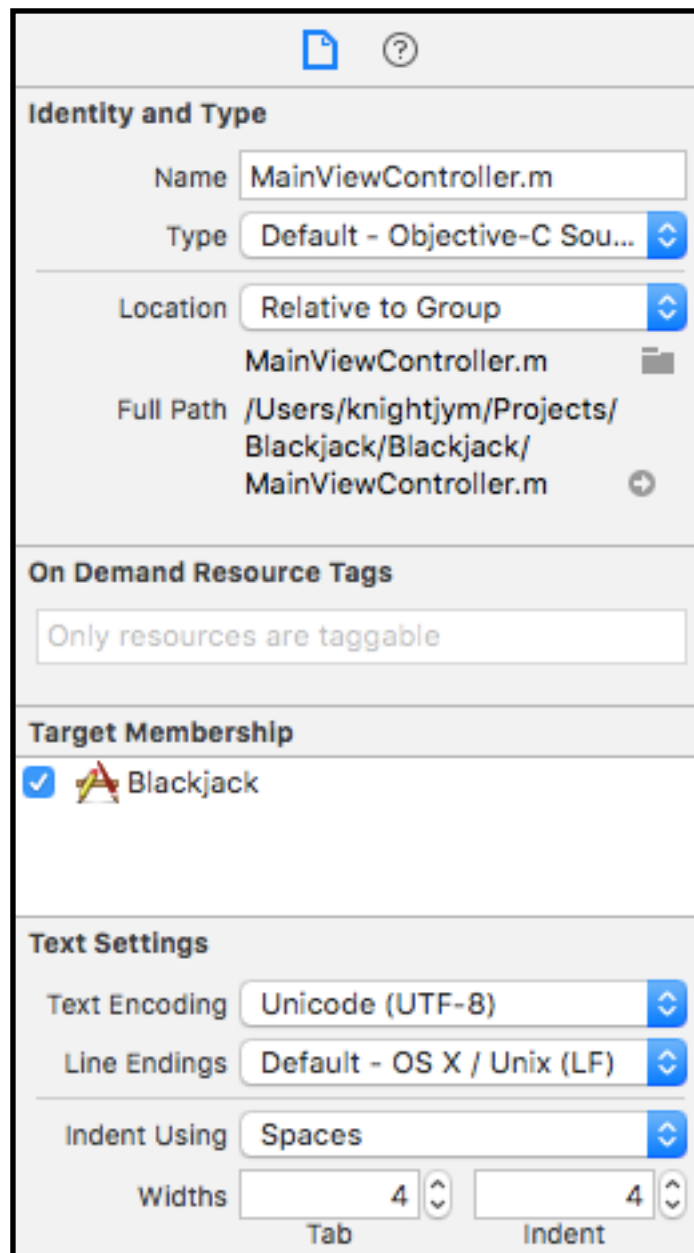
```
2016-04-21 13:35:57.179 Blackjack[59913:460018] card
num : 11
2016-04-21 13:35:57.179 Blackjack[59913:460018] card
num : 4
2016-04-21 13:35:57.180 Blackjack[59913:460018] card
num : 10
2016-04-21 13:35:57.180 Blackjack[59913:460018] card
num : 10
2016-04-21 13:35:57.181 Blackjack[59913:460018] card
num : 5
2016-04-21 13:35:57.182 Blackjack[59913:460018] 여기는 디
버깅 창입니다.
(lldb)
```

All Output ↕

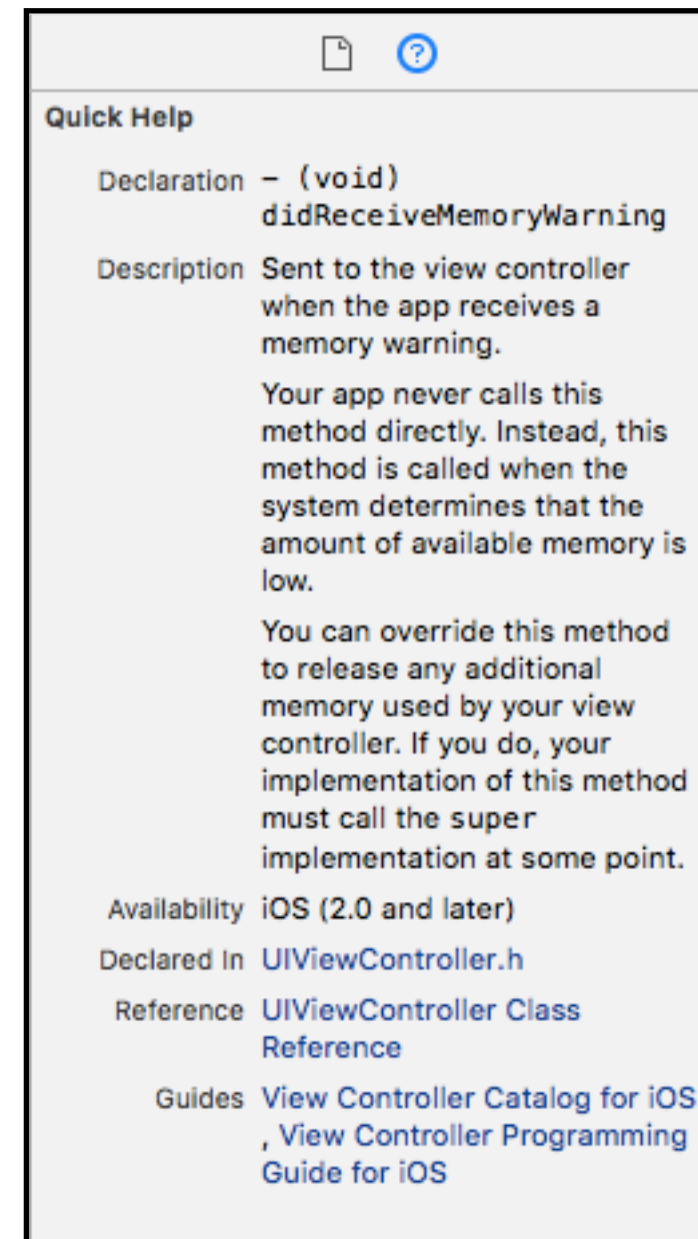
Utilities



Utilities - Inspector1

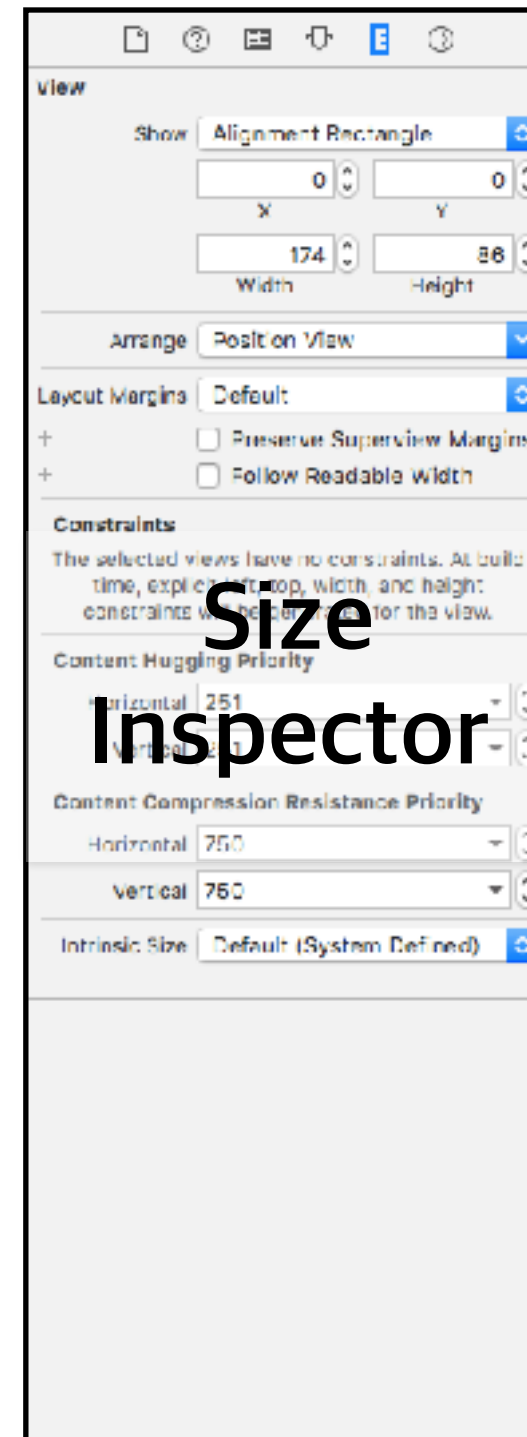
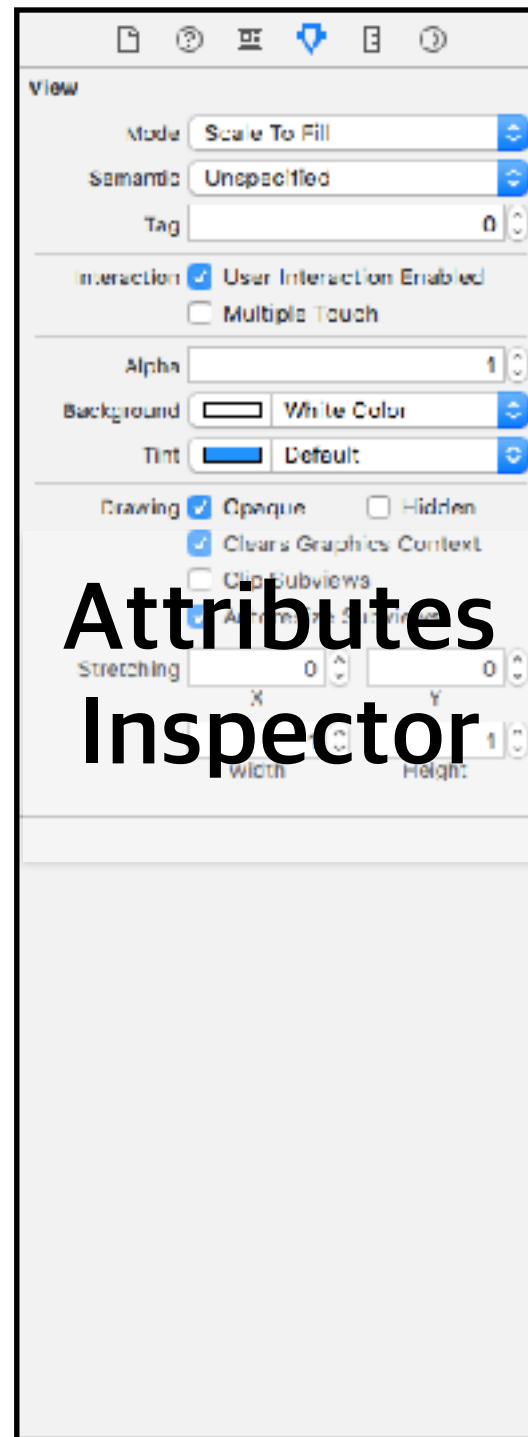
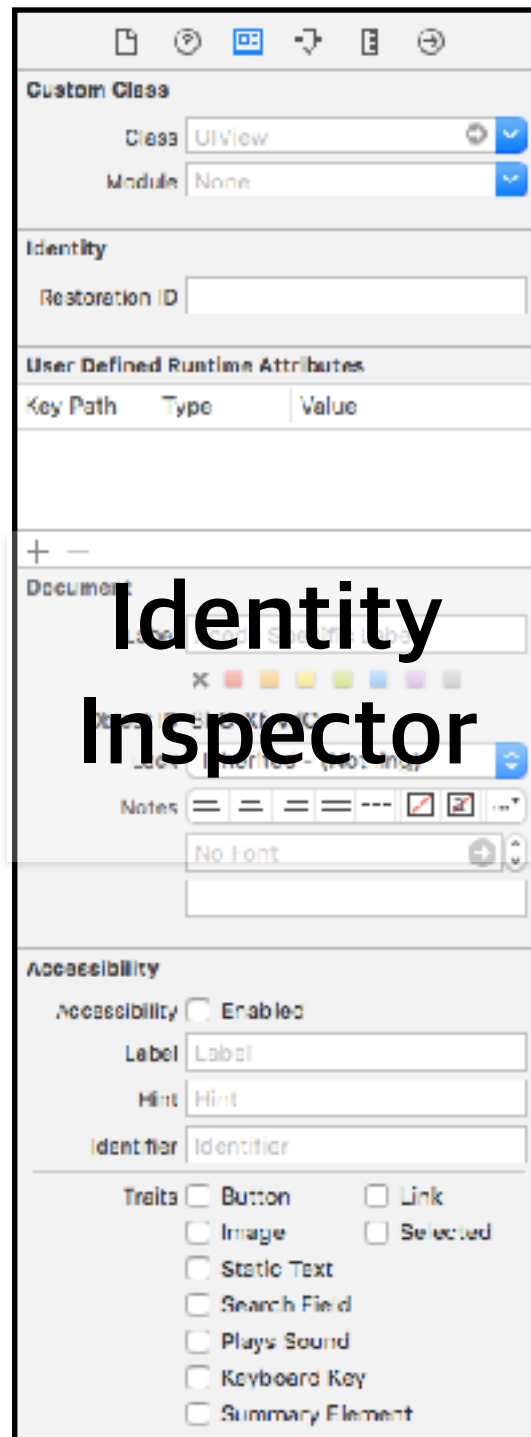


File Inspector



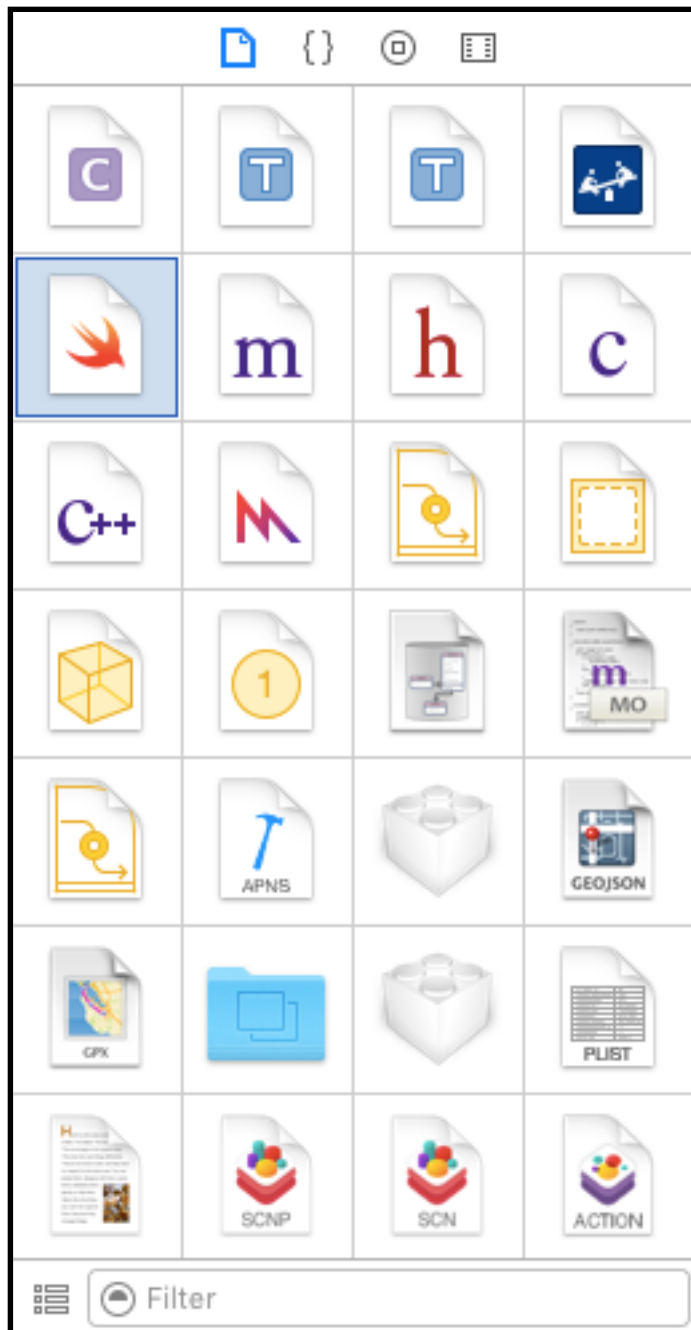
Quick Help Inspector

Utilities - Inspector2 ver UI

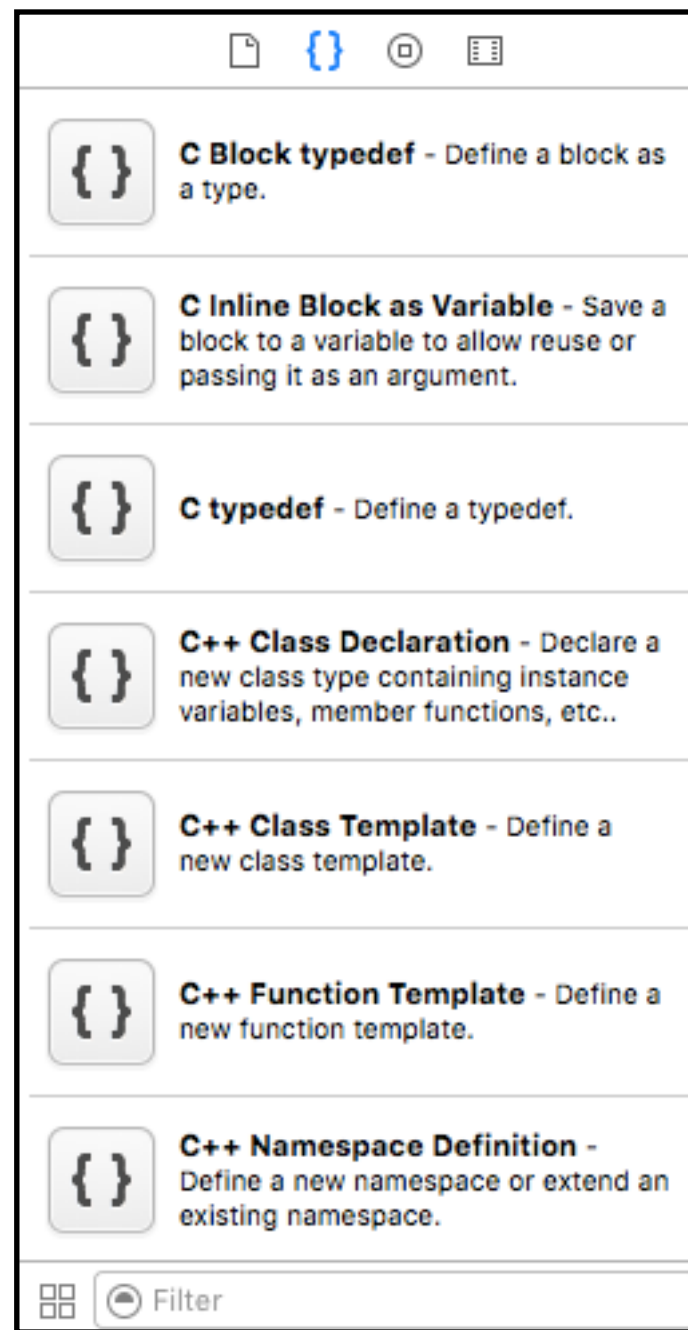


Utilities - library

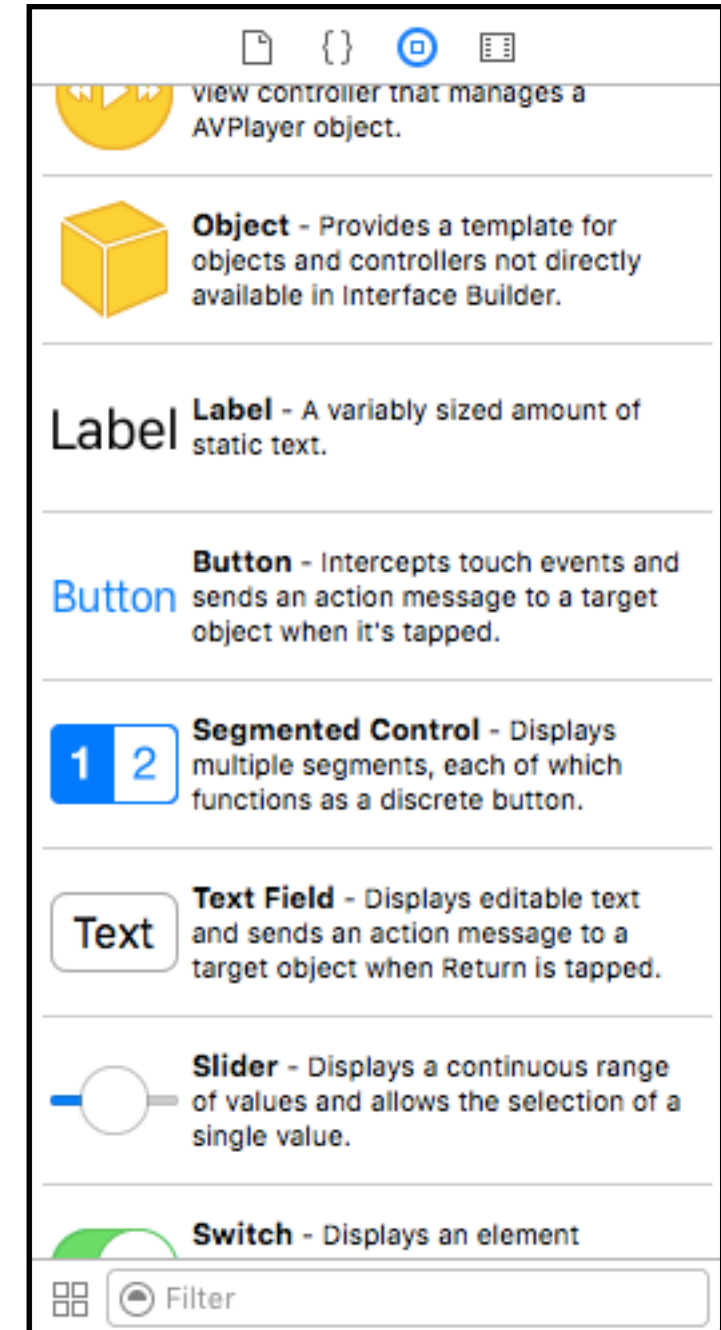
File Template



Code Snippet

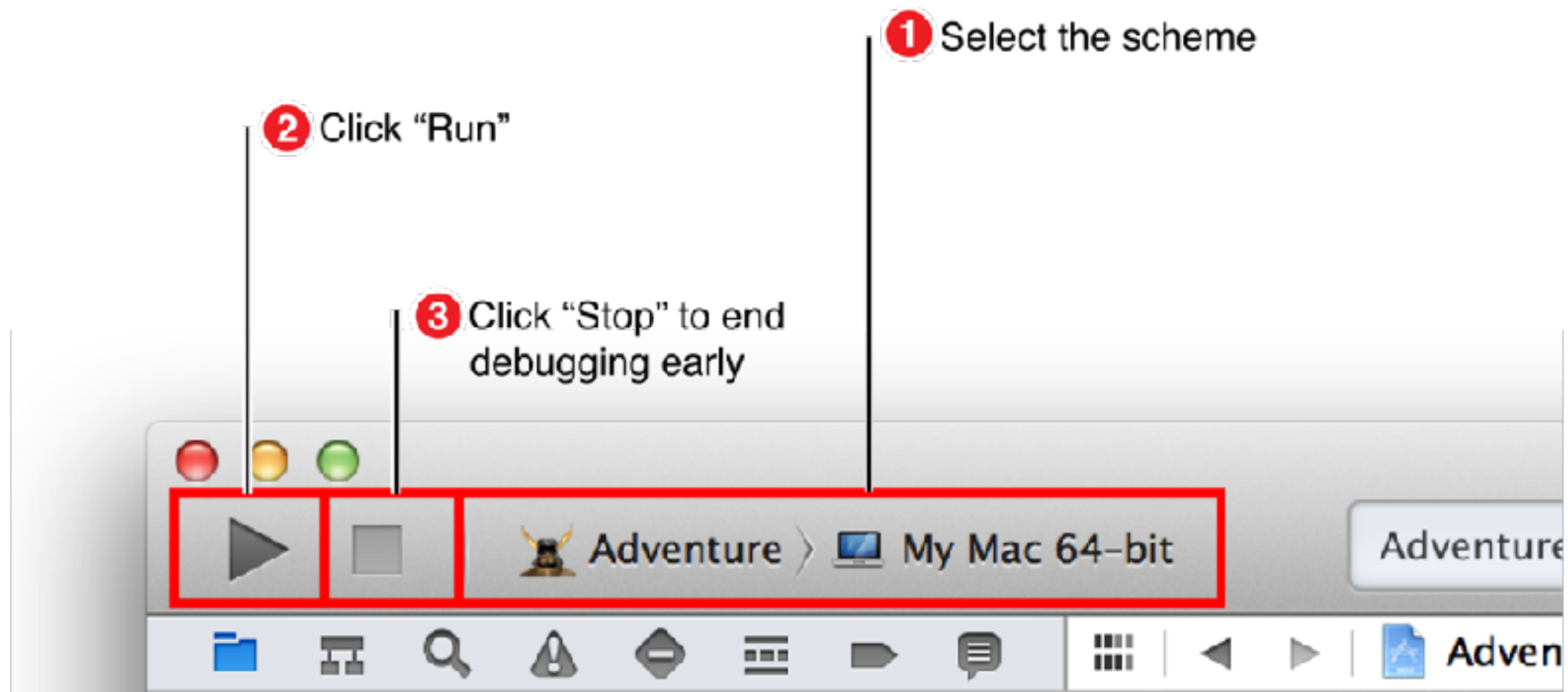


Object



빌드 & 런

1. Select an active scheme and destination.
2. Click Run to build and run your code with the active scheme.
3. Use the Stop button to stop an in-progress build or end the current debugging session.



객체지향기초

기초 중의 기초

- 객체 지향형 프로그래밍이란?

객체지향형 프로그래밍 방법론

객체 지향 프로그래밍은 컴퓨터 프로그램을 명령어의 목록으로 보는 시각에서 벗어나 여러 개의 독립된 단위, 즉 "객체"들의 모임으로 파악하고자 하는 것이다. 각각의 객체는 메시지를 주고받고, 데이터를 처리할 수 있다.

객체지향형 프로그래밍 방법론

즉!! 객체의 단위로 쪼개서 개발하는 방법론!

RPG 게임을 만들어 볼까요??

- 디아블로를 예로 들어서 생각해 봅시다.
- 게임을 만든다고 했을때 어떻게 만들어야 할까요?

명세서

Map

- 스테이지 단계
- 캐릭터
- 몬스터 들
- 지형
- 아이템

캐릭터

- 직업
- 이름
- 스킬
- 레벨
- 스탯
- 공격액션
- 등등...

아이템

- 공격력
- 마법력
- 모습
- 특수능력
- 소유자
- 등등...

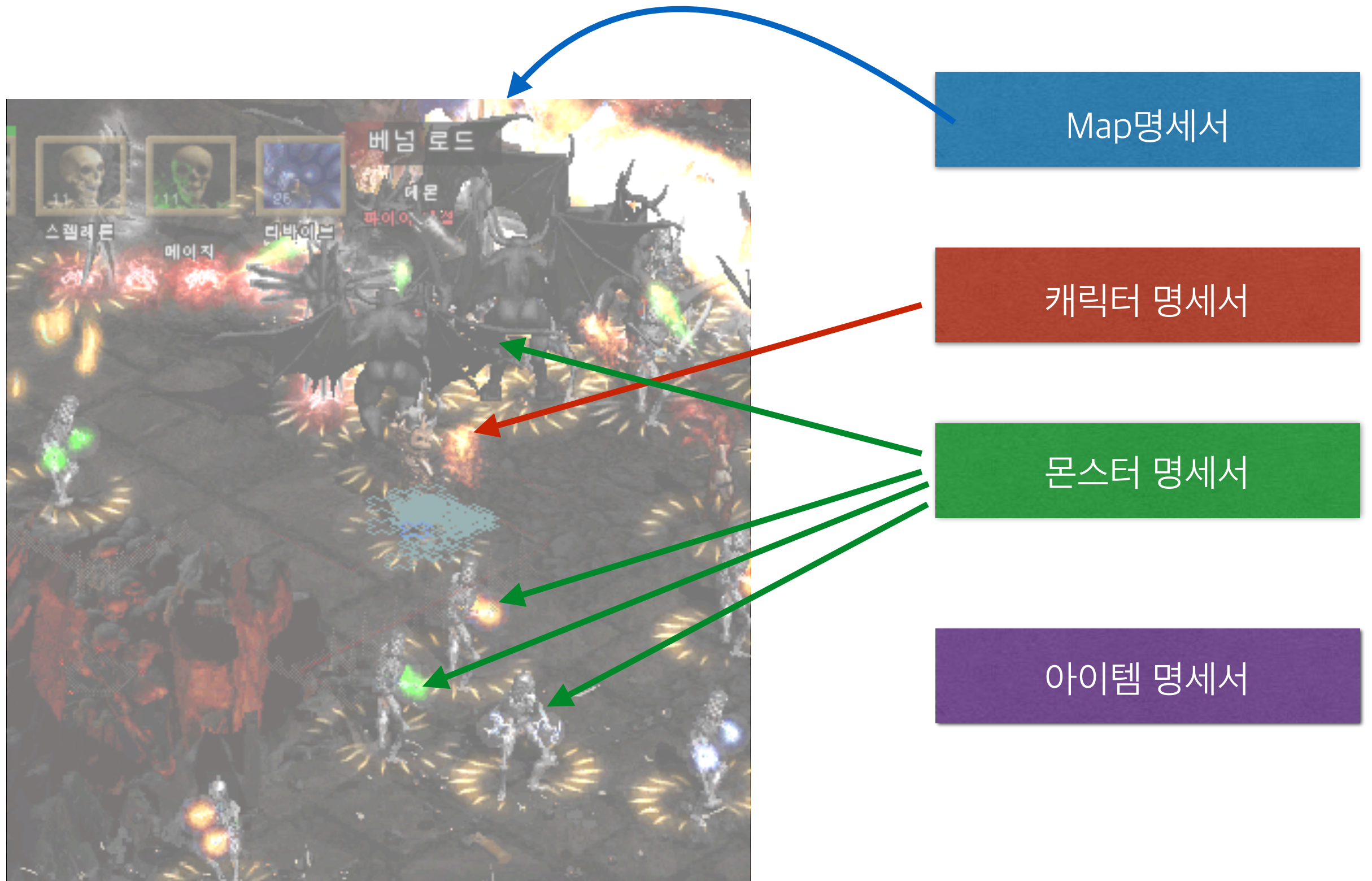
몬스터

- 이름
- 레벨
- 공격액션
- 모습(이미지)
- 등등...

몇줄의 코드를 만들어야 할까요?



객체 지향



기본 구성 요소

- **클래스(Class)** - 같은 종류(또는 문제 해결을 위한)의 집단에 속하는 속성(attribute)과 행위(behavior)를 정의한 것으로 객체지향 프로그램의 기본적인 사용자 정의 데이터형(user define data type)이라고 할 수 있다. 클래스는 프로그래머가 아니지만 해결해야 할 문제가 속하는 영역에 종사하는 사람이라면 사용할 수 있고, 다른 클래스 또는 외부 요소와 독립적으로 디자인하여야 한다.
- **객체(Object)** - 클래스의 인스턴스(실제로 메모리상에 할당된 것)이다. 객체는 자신 고유의 속성(attribute)을 가지며 클래스에서 정의한 행위(behavior)를 수행할 수 있다. 객체의 행위는 클래스에 정의된 행위에 대한 정의를 공유함으로써 메모리를 경제적으로 사용한다.
- **메서드(Method), 메시지(Message)** - 클래스로부터 생성된 객체를 사용하는 방법으로서 객체에 명령을 내리는 메시지라 할 수 있다. 메서드는 한 객체의 서브루틴(subroutine) 형태로 객체의 속성을 조작하는 데 사용된다. 또 객체 간의 통신은 메시지를 통해 이루어진다.

객체지향형 프로그래밍 특징

- 추상화
- 캡슐화
- 은닉화
- 상속성
- 다형성

따라해보아요!

- 클래스 만들기과 인스턴스 만들기 실습

기초 문법

Swift Class Architecture

```
class ClassName : superClass
{
    var vName1 = "1"
    var vName2 = 4

    func fName1() - > Any
    {

    }

    func fName2(_ ani:Bool)
    {

    }
}
```

<ClassName.swift>

클래스 만들기

```
class Person  
{
```

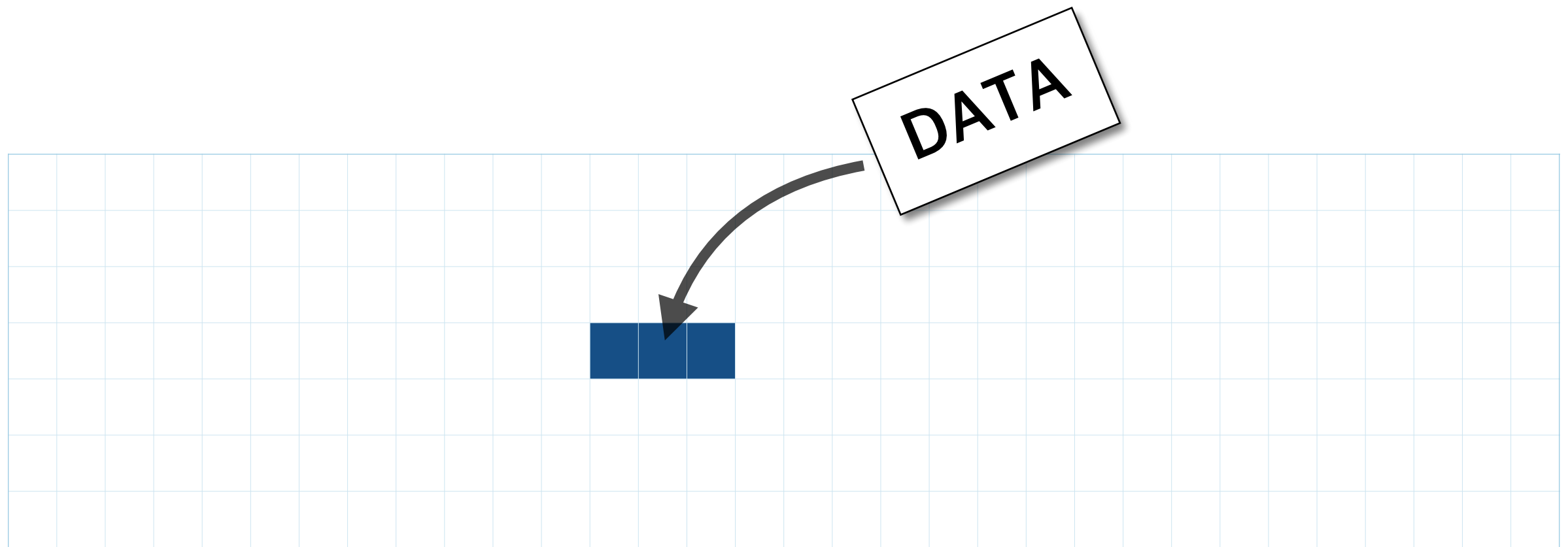
```
//클래스 내용 작성 ( 변수 & 함수)
```

```
}
```

변수 & 함수

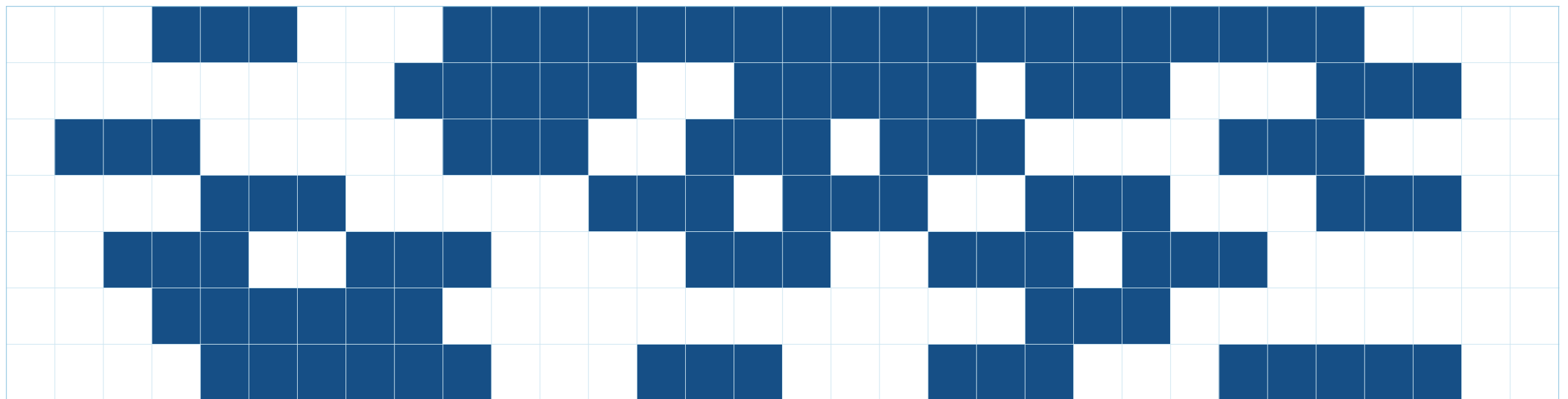
- 변수 : 프로그램에서 데이터의 저장공간을 담당
- 함수 : 프로그램이 실행되는 행동을 담당

변수



<메모리>

각 메모리 안에는 어떤 데이터가 들어있을까요?
조금 전 넣은 데이터는 어디 일까요?



<메모리>

변수 문법

키워드 + 변수 명(Name) + 변수 타입(Type)

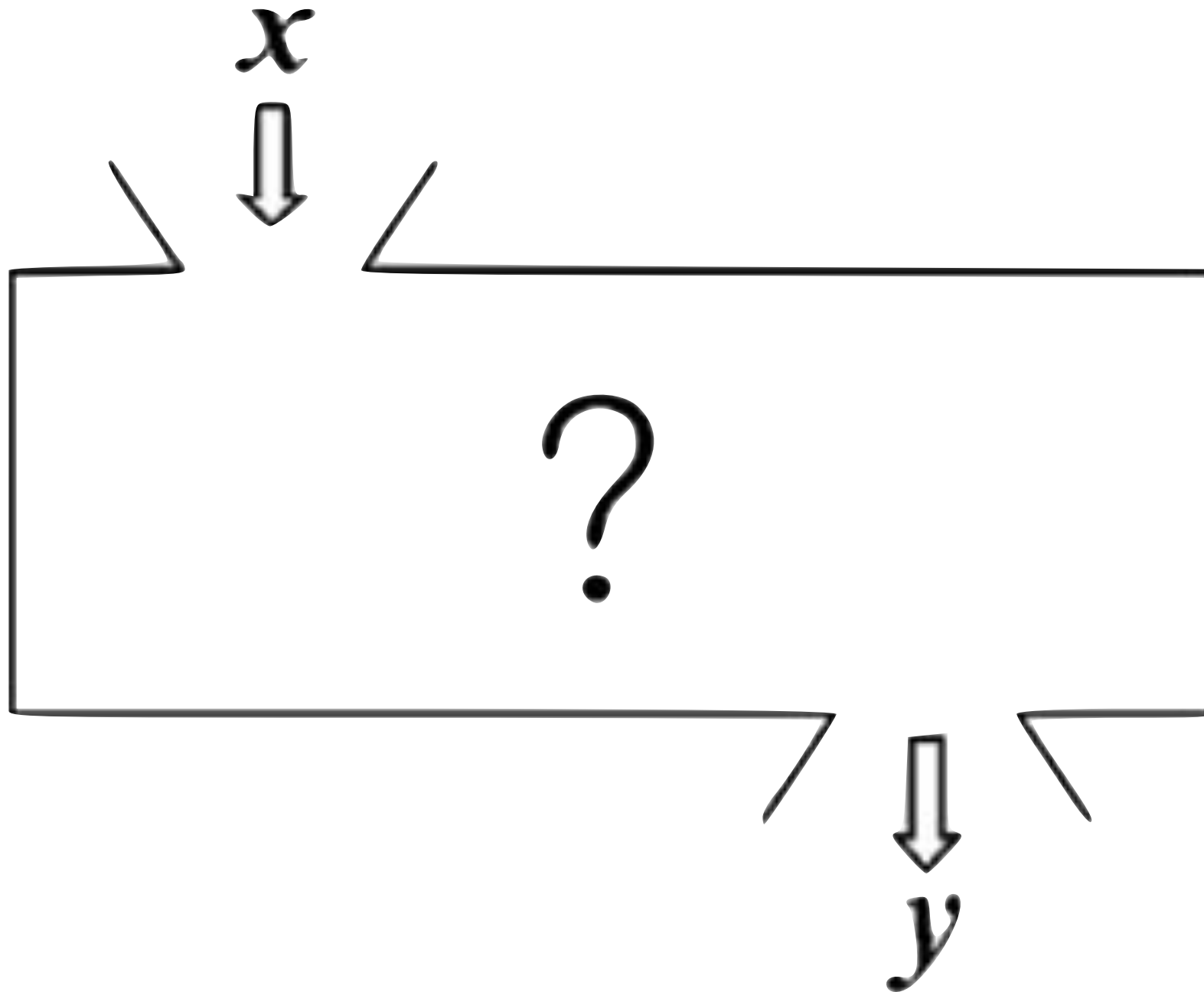
문법 : `var` vName:Any

변수 값 할당

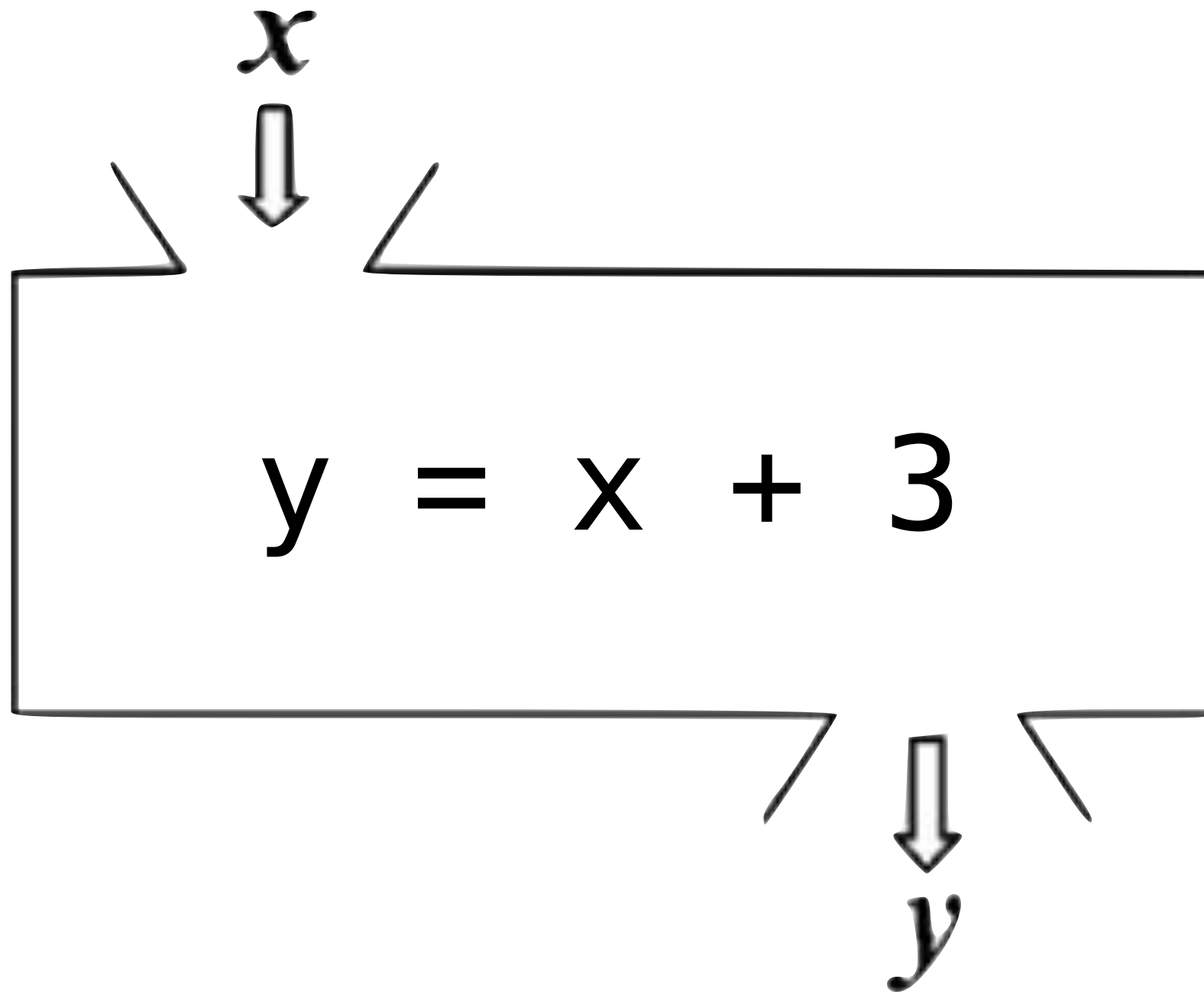
변수 명(Name) = 값(Value)

문법 : `var` vName:Any = 3

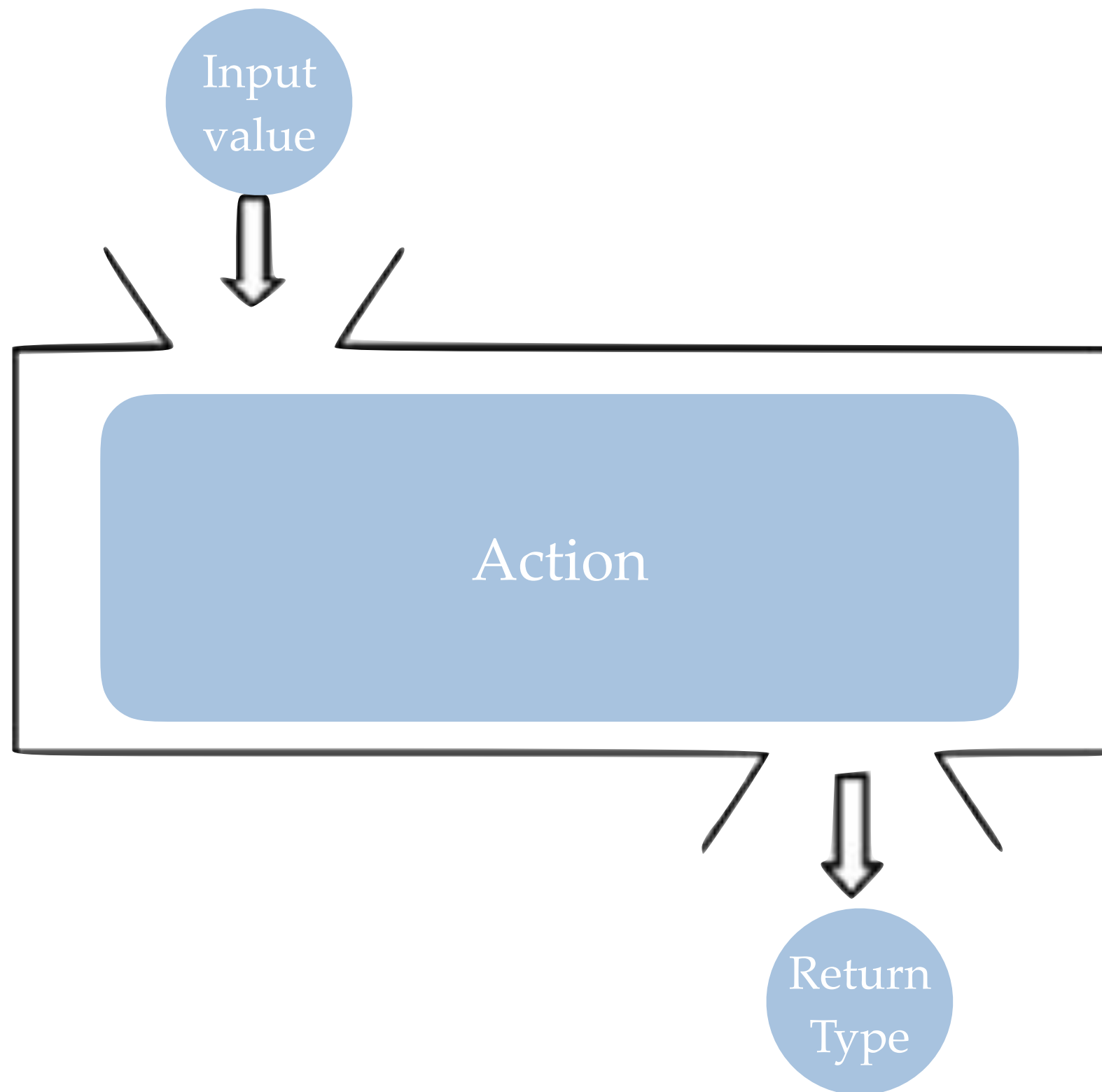
함수



함수



함수



- 함수 만들기 위해 필요한것?

키워드 + 함수명(Name) + 입력값(Input Value) +
함수 내용(Action) + 결과타입(Return Type)

문법 : `func vName(parameter: Any) -> Any`
 {
 //함수 내용
 }

연습해보기

```
class 사람
{
    var 이름: String = ""
    // 추가 속성

    func 달리기()
    {
        print("사람이 달려갑니다.")
    }
    // 추가 함수
}
```

- 사람 클래스를 만들어 보세요
- 나만의 클래스를 하나 만들어 보세요

주석

- 컴파일러가 인식할 수 없는 텍스트
- 메모 및 설명을 작성하는 용도로 사용
- 코드의 실행을 막는 경우로 사용

한줄 주석

- “//” 기호를 사용해서 한줄씩 주석 처리
- command + / 키로 주석 설정/해제 가능

//주석을 작성하세요
//라인별로 표시를 해줘야 합니다.

여러줄 주석

- “/*” 시작기호와 “*/” 끝기호를 사용하여 여러 라인의 줄을 모두 주석 처리

```
/*  
    이 안의 모든  
    내용은  
    주석으로  
    처리  
    가능합니다.  
*/
```

주석으로 quick help 문서 만들기

- https://developer.apple.com/library/content/documentation/Xcode/Reference/xcode_markup_formatting_ref/SymbolDocumentation.html#//apple_ref/doc/uid/TP40016497-CH51-SW1

Class 사용하기

- 인스턴스(객체 만들기) 생성
- 인스턴스 변수에 저장
- 인스턴스의 속성(변수), 함수 접근 및 실행 (닷 . 문법 사용)

Class 사용하기

- 인스턴스(객체 만들기) 생성 - 초기화

```
사람()
```

- 인스턴스 변수에 저장

```
var 영민: 사람 = 사람()
```

- 인스턴스의 속성(변수), 함수 접근 및 실행 (닷 . 문법 사용)

```
영민.이름 = "주영민"  
영민.달리기()
```


한글 어색해요ㅠㅠ

- swift는 한글이 지원됩니다! 하지만 글로벌 사회인 만큼 영어로 만들겠습니다.(지금부터 한글 금지령)
- 클래스명, 변수명, 함수명, 모두 영어로 바꿔서 다시 작업 만들어 봅시다.

실제 동작하는 코드 확인해보기

- 이미 만들어진 코드확인 해보기