



ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 2B

A CUSTOM OBJECTIVE-C CLASS

A CUSTOM OBJECTIVE-C CLASS

The screenshot shows a Xcode interface with a code editor containing Objective-C code. A red arrow points from the bottom left towards the code editor. A context menu is open over the code, with 'File...' selected. The code editor contains the following:

```
19 // Do any
20 }
21 - (void)didReceiveMemoryWarning {
22     [super didReceiveMemoryWarning];
23     // Dispose of any cached data here
24 }
25
26 @end
27
28 }
```

The Xcode interface includes a top menu bar with 'Tab' and 'Window' options, and a right-hand sidebar with settings for 'Line Endings', 'Indent Using', 'Widths', 'Source Control', and a 'View Controller' section.

File... was selected in the context menu, which typically leads to creating a new file or opening an existing one.

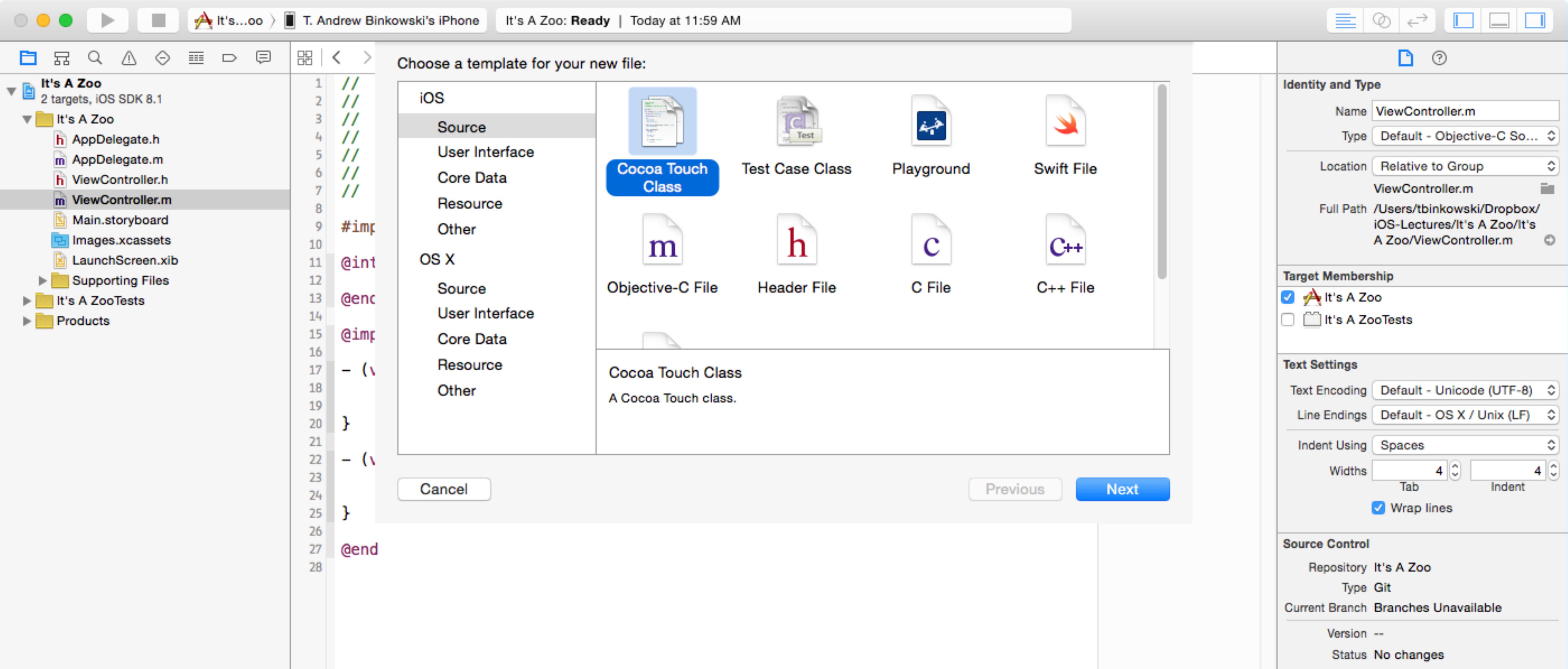
Source Control section details:

- Repository: It's A Zoo
- Type: Git
- Current Branch: Branches Unavailable
- Version: --
- Status: No changes
- Location: /Users/tbinkowski/Dropbox/iOS-Lectures/It's A Zoo/It's A Zoo/ViewController.m

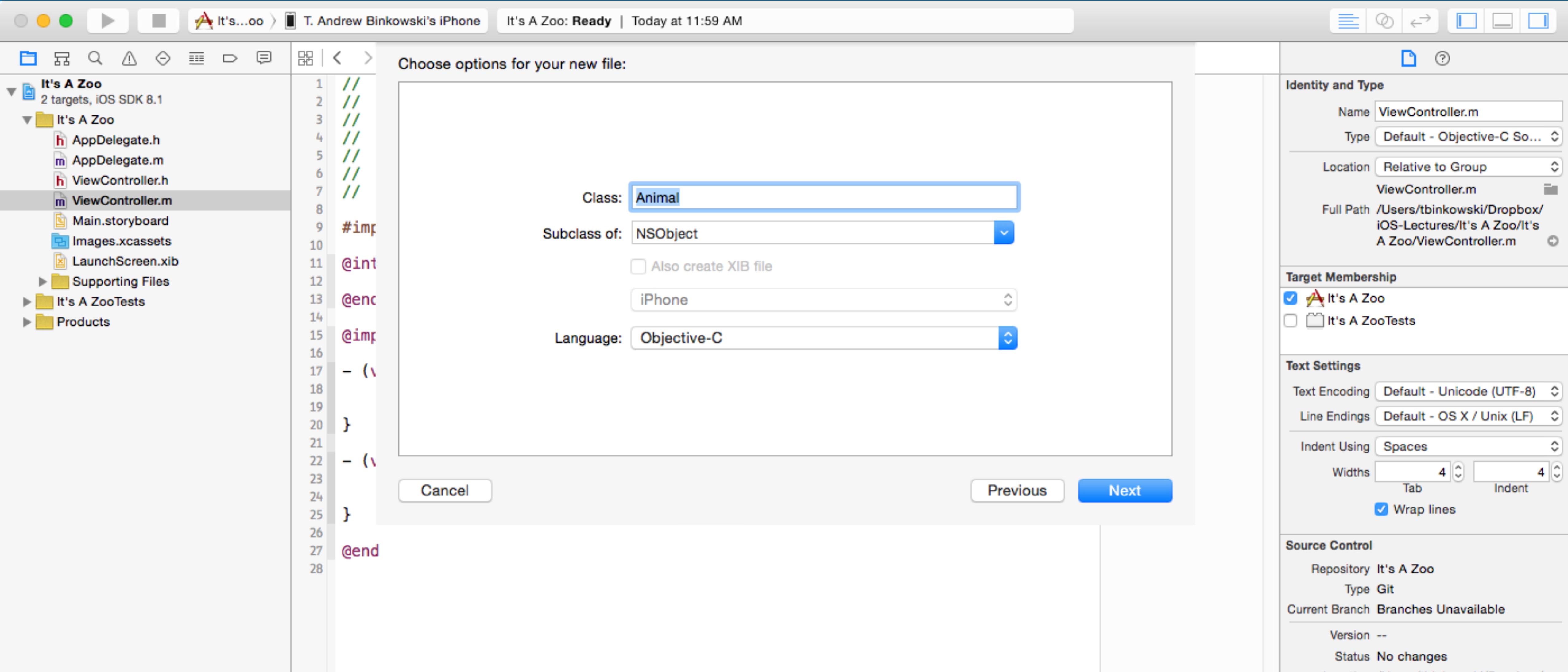
View Controller section details:

- View Controller** - A controller that supports the fundamental view-management model in iOS.
- Navigation Controller** - A controller that manages navigation through a hierarchy of views.
- Table View Controller** - A controller that manages a table view.

A CUSTOM OBJECTIVE-C CLASS



A CUSTOM OBJECTIVE-C CLASS



A CUSTOM OBJECTIVE-C CLASS

Header file

```
1 // Animal.h
2 // It's A Zoo
3 //
4 // Created by T. Andrew Binkowski on 1/12/15.
5 // Copyright (c) 2015 The University of Chicago. All rights reserved.
6 //
7
8
9 #import <Foundation/Foundation.h>
10 @import UIKit;
11
12 @interface Animal : NSObject
13
14 // Declared properties
15 @property (strong, nonatomic) NSString *name;
16 @property (strong, nonatomic) NSString *species;
17 @property (strong, nonatomic) NSNumber *age;
18 @property (strong, nonatomic) UIImage *image;
19
20 // Instance method
21 - (NSString*)describeAnimal;
22
23
24 @end
```

A CUSTOM OBJECTIVE-C CLASS

We don't have to write [init] methods; only if you want to

```
0
9 #import "Animal.h"
10
11 @implementation Animal
12
13
14 - (NSString*)describeAnimal
15 {
16     NSString *descriptionString = [NSString stringWithFormat:@"Animal Name:%@ Species:%@ Age:%@ \n",
17                                     self.name, self.species, self.age];
18     return descriptionString;
19 }
20
21 @end
22
```

A CUSTOM OBJECTIVE-C CLASS

h Animal.h
m Animal.m
Main.storyboard
Images.xcassets
LaunchScreen.xib
▶ Supporting Files
▶ It's A ZooTests
▶ Products

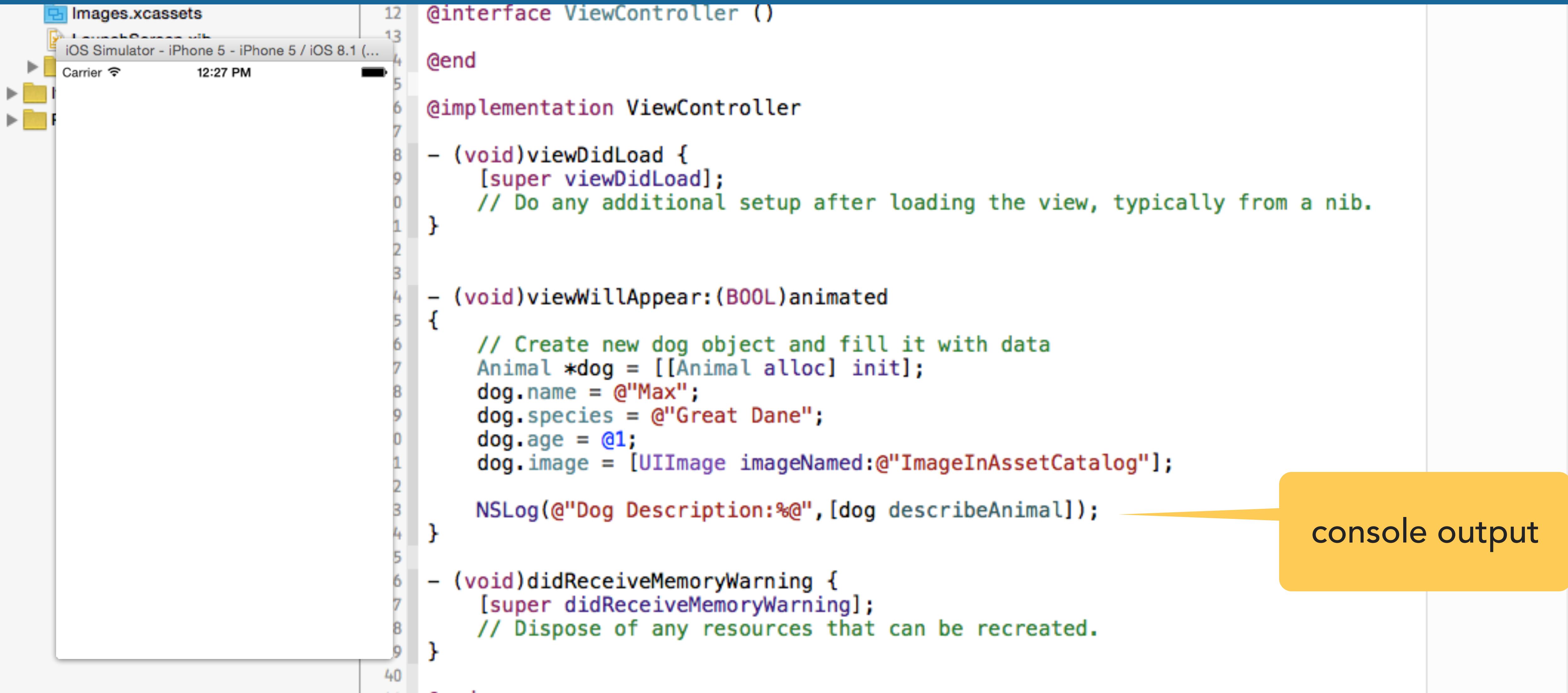
```
9 #import "ViewController.h"
10 #import "Animal.h"
11
12 @interface ViewController : UIViewController
13
14 @end
15
16 @implementation ViewController
17
18 - (void)viewDidLoad {
19     [super viewDidLoad];
20     // Do any additional setup after loading the nib.
21 }
22
23
24 - (void)viewWillAppear:(BOOL)animated
25 {
26     // Create new dog object and fill it with data
27     Animal *dog = [[Animal alloc] init];
28     dog.name = @"Max";
29     dog.species = @"Great Dane";
30     dog.age = @1;
31     dog.image = [UIImage imageNamed:@"ImageInAssetCatalog"];
32
33     NSLog(@"Dog Description:%@", [dog describeAnimal]);
34 }
35
36 - (void)didReceiveMemoryWarning {
37     [super didReceiveMemoryWarning];
```

import class

create new class nib.

call our method;
message our
object

A CUSTOM OBJECTIVE-C CLASS



The screenshot shows the Xcode IDE interface. On the left is the Project Navigator with files like 'Images.xcassets' and 'ViewController.m'. The center is the Editor pane displaying the code for 'ViewController.m'. The code defines an interface and implementation for a View Controller, including methods for view loading, appearance, and memory management. A callout bubble labeled 'console output' points to the bottom right of the code area.

```
12 @interface ViewController : UIViewController
13
14 @end
15
16 @implementation ViewController
17
18 - (void)viewDidLoad {
19     [super viewDidLoad];
20     // Do any additional setup after loading the view, typically from a nib.
21 }
22
23
24 - (void)viewWillAppear:(BOOL)animated
25 {
26     // Create new dog object and fill it with data
27     Animal *dog = [[Animal alloc] init];
28     dog.name = @"Max";
29     dog.species = @"Great Dane";
30     dog.age = @1;
31     dog.image = [UIImage imageNamed:@"ImageInAssetCatalog"];
32
33     NSLog(@"Dog Description:%@", [dog describeAnimal]);
34 }
35
36 - (void)didReceiveMemoryWarning {
37     [super didReceiveMemoryWarning];
38     // Dispose of any resources that can be recreated.
39 }
```

console output

OBJECTIVE-C IN A SWIFT PROJECT

Swift and Objective-C in the Same Project

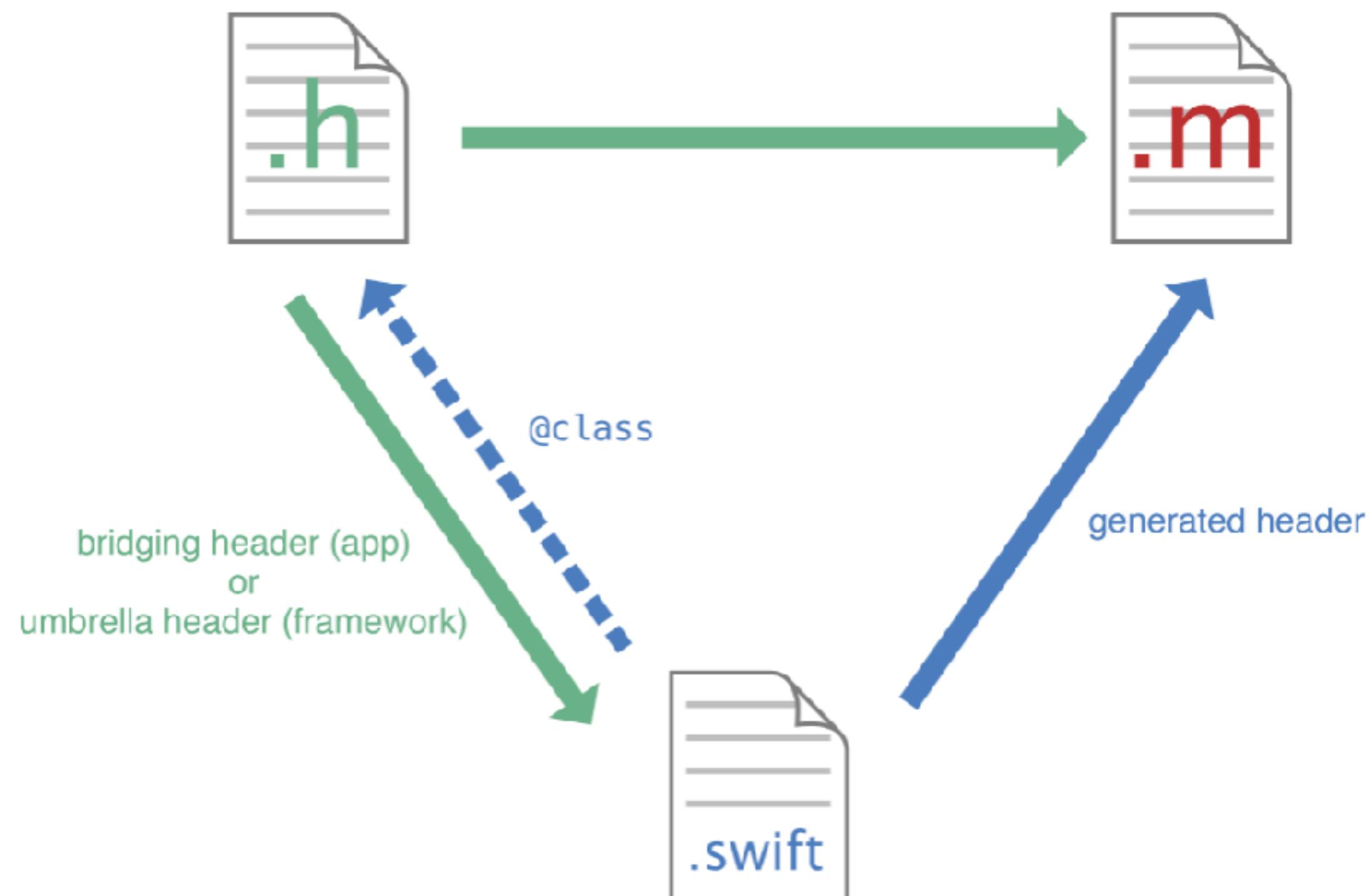
Swift's compatibility with Objective-C lets you create a project that contains files written in either language. You can use this feature, called *mix and match*, to write apps that have a mixed-language codebase. Using mix and match, you can implement part of your app's functionality using the latest Swift features and seamlessly incorporate it back into your existing Objective-C codebase.

Mix and Match Overview

<https://developer.apple.com/library/content/documentation/Swift/Conceptual/BuildingCocoaApps/MixandMatch.html>

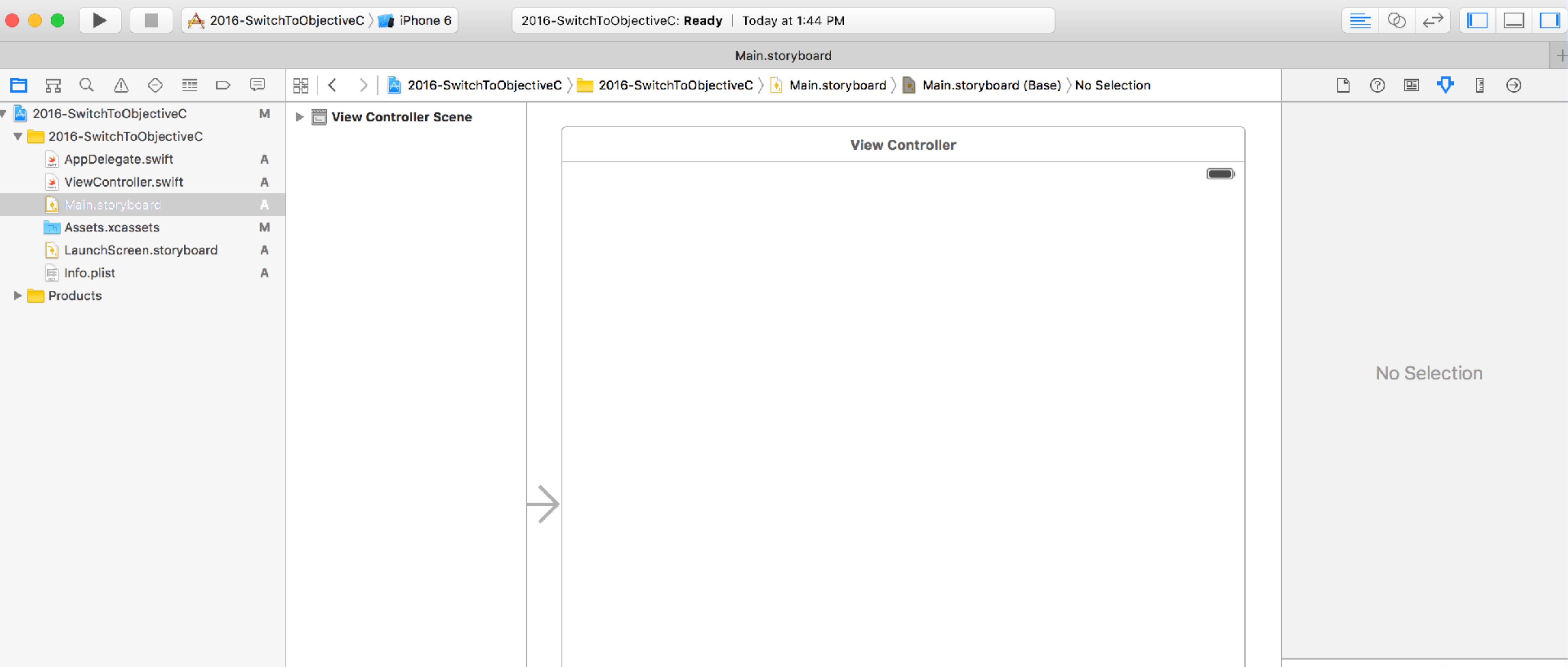
Swift project. You can simply add a file of the other language directly to an existing project. This natural

CREATE A PROJECT



AN OBJECTIVE-C VIEW CONTROLLER

CREATE A PROJECT



AN OBJECTIVE-C VIEW CONTROLLER

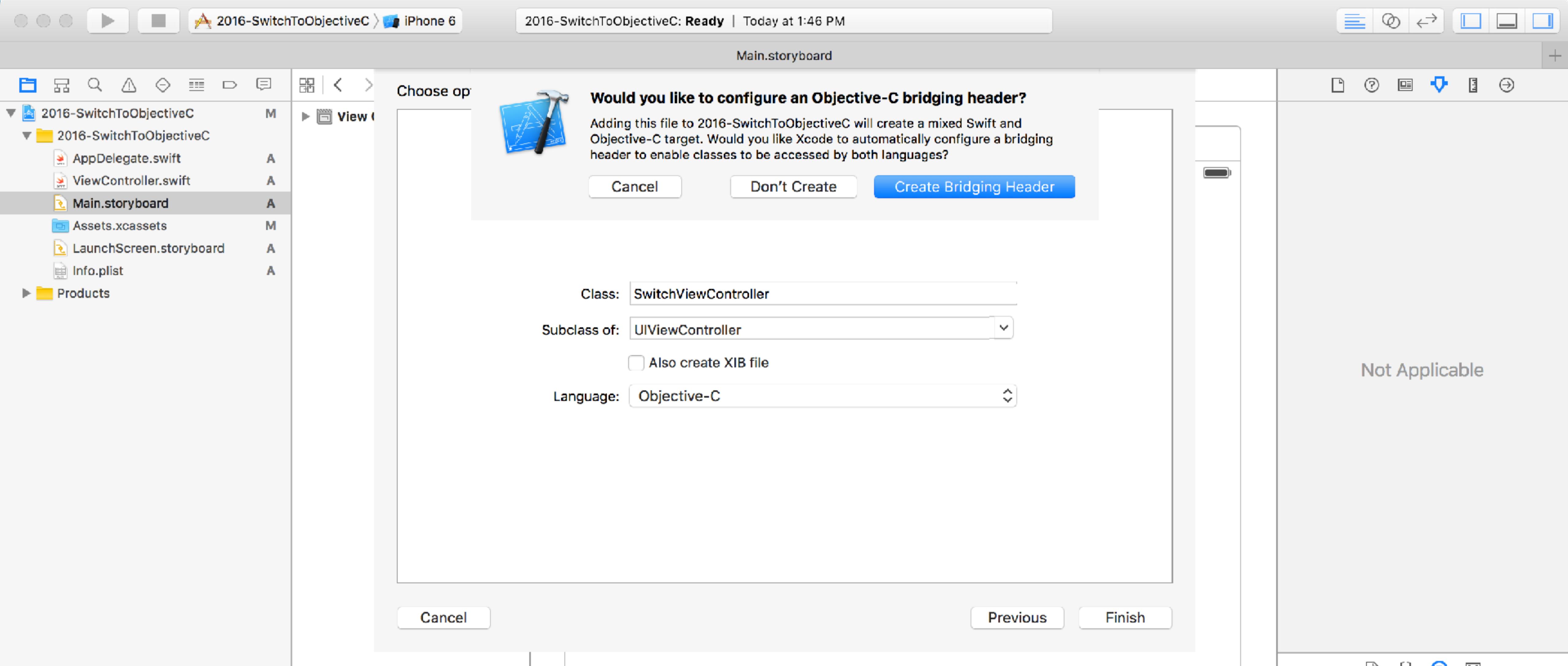
The screenshot shows the Xcode interface with a project named "2016-SwitchToObjectiveC". The file list on the left includes `AppDelegate.swift`, `ViewController.swift`, `Main.storyboard` (which is selected), `Assets.xcassets`, `LaunchScreen.storyboard`, and `Info.plist`. A "Products" folder is also listed.

A modal dialog is open in the center, titled "New View Controller". The "Class" field contains `SwitchViewController`. The "Subclass of:" dropdown is set to `UIViewController`. The "Also create XIB file" checkbox is unchecked. The "Language:" dropdown is set to `Objective-C`. At the bottom of the dialog are "Cancel", "Previous", and "Next" buttons. The "Next" button is highlighted in blue.

To the right of the dialog, there is a sidebar with a "Not Applicable" section and two items at the bottom:

- View Controller** - A controller that manages a view. (Icon: yellow rounded rectangle)
- Storyboard Reference** - Provides a placeholder for a view controller in an (Icon: orange rounded rectangle)

AN OBJECTIVE-C VIEW CONTROLLER



AN OBJECTIVE-C VIEW CONTROLLER

The screenshot shows two Xcode editor panes side-by-side, illustrating the transition from Swift to Objective-C.

Left Pane (Swift View Controller):

- Project Navigator: Shows the project structure with files like AppDelegate.swift, ViewController.swift, Main.storyboard, SwitchViewController.h, SwitchViewController.m, Assets.xcassets, LaunchScreen.storyboard, Info.plist, and 2016-SwitchT...dging-Header.h.
- Editor: Displays the Swift code for `SwitchViewController`. It includes imports for `UIKit`, defines the `SwitchViewController` class as a `UIViewController`, and ends with an `@end` directive.

```
// SwitchViewController.h
// 2016-SwitchToObjectiveC
//
// Created by T. Andrew Binkowski on 4/3/16.
// Copyright © 2016 The University of Chicago, Department of Economics.

#import <UIKit/UIKit.h>

@interface SwitchViewController : UIViewController
@end
```

Right Pane (Objective-C View Controller):

- Project Navigator: Shows the project structure with files like AppDelegate.swift, ViewController.swift, Main.storyboard, SwitchViewController.h, SwitchViewController.m, Assets.xcassets, LaunchScreen.storyboard, Info.plist, and 2016-SwitchT...dging-Header.h.
- Editor: Displays the Objective-C code for `SwitchViewController`. It includes an import for `"SwitchViewController.h"`, defines the `SwitchViewController` class, and implements methods for `viewDidLoad` and `didReceiveMemoryWarning`.

```
// SwitchViewController.m
// 2016-SwitchToObjectiveC
//
// Created by T. Andrew Binkowski on 4/3/16.
// Copyright © 2016 The University of Chicago, Department of Economics.

#import "SwitchViewController.h"

@interface SwitchViewController ()
```

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view.
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(
    // Get the new view controller using [segue destinationViewController].
    // Pass the selected object to the new view controller.
```

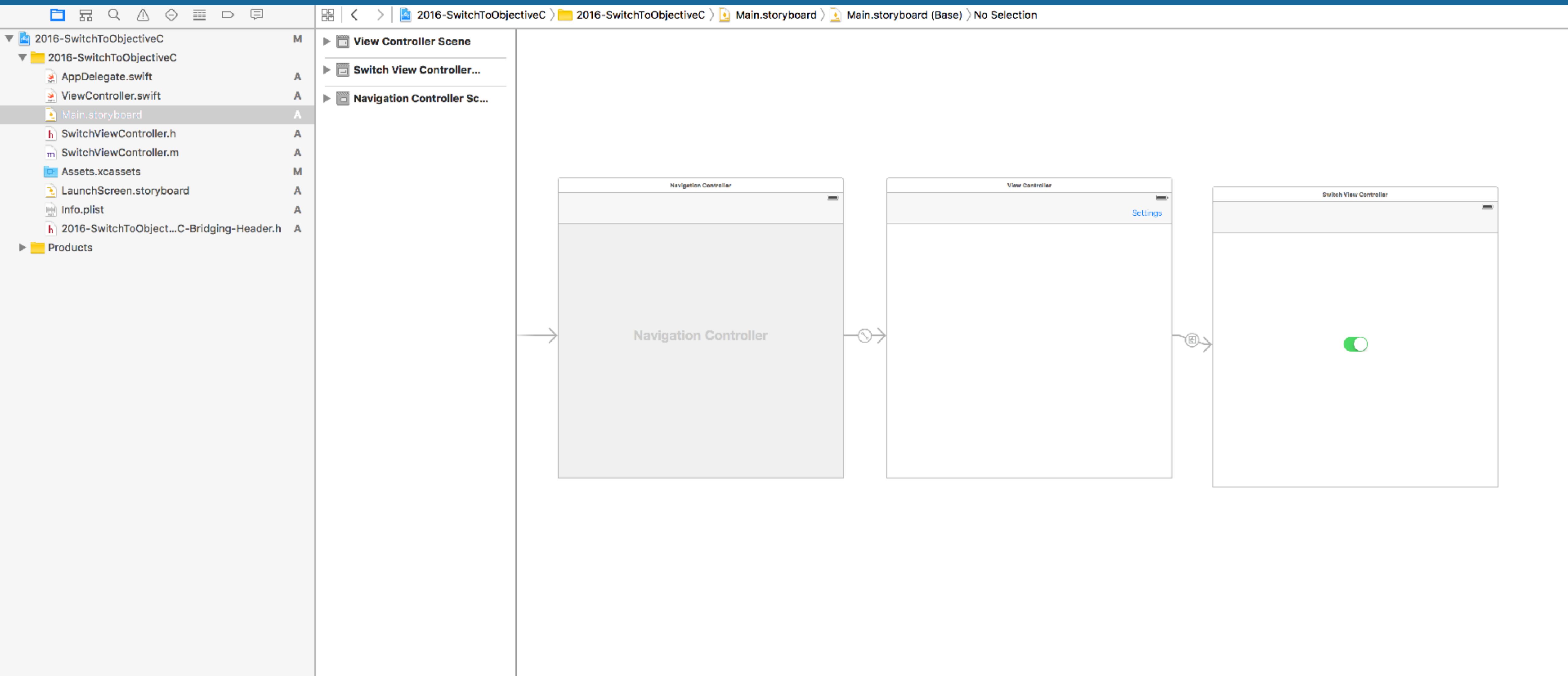
AN OBJECTIVE-C VIEW CONTROLLER

The image shows a screenshot of an Xcode project. On the left, the file navigator displays a project named "2016-SwitchToObjectiveC" containing a folder "2017-SwitchToObjectiveC" which includes files like AppDelegate.swift, ViewController.swift, SwitchViewController.h, SwitchViewController.m, Assets.xcassets, Main.storyboard, LaunchScreen.storyboard, Info.plist, 2016-SwitchToObjectiveC-Bridging-Header.h, GreetingObject.h, and GreetingObject.m. A yellow arrow points from the "GreetingObject.h" file in the navigator to the code editor on the right. The code editor contains the following Objective-C header:

```
//  
// Use this file to import your target's public headers that  
// you can then #import in your header files.  
//  
#import "GreetingObject.h"
```

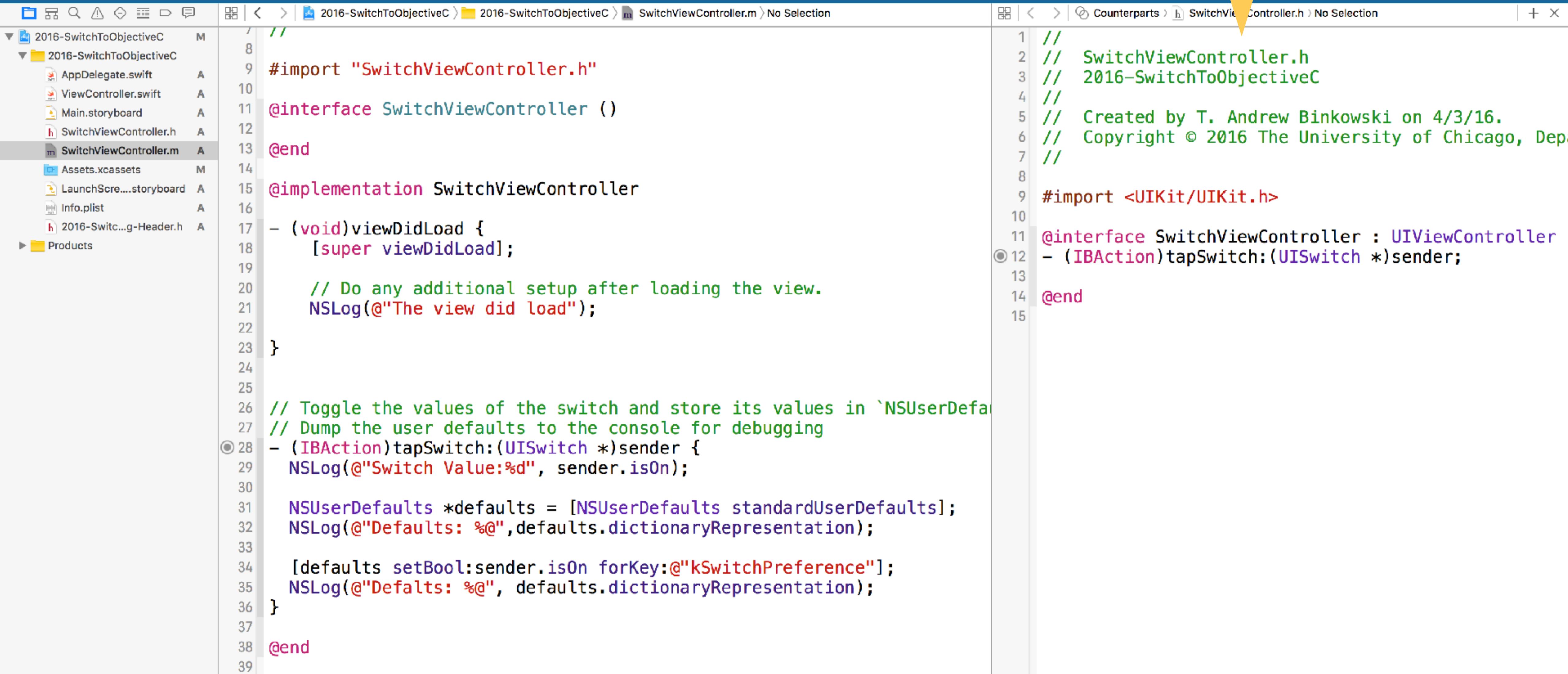
A yellow callout box with a black border and text "IMPORT THE OBJECTIVE-C HEADER INTO A SWIFT PROJECT FOR CUSTOM CLASSES" is positioned to the right of the code editor, with a yellow arrow pointing towards the "#import" line.

AN OBJECTIVE-C VIEW CONTROLLER



AN OBJECTIVE-C VIEW CONTROLLER

IBOUTLETS WILL AUTOMATICALLY CREATE METHOD DEFINITIONS DURING DRAG/DROP



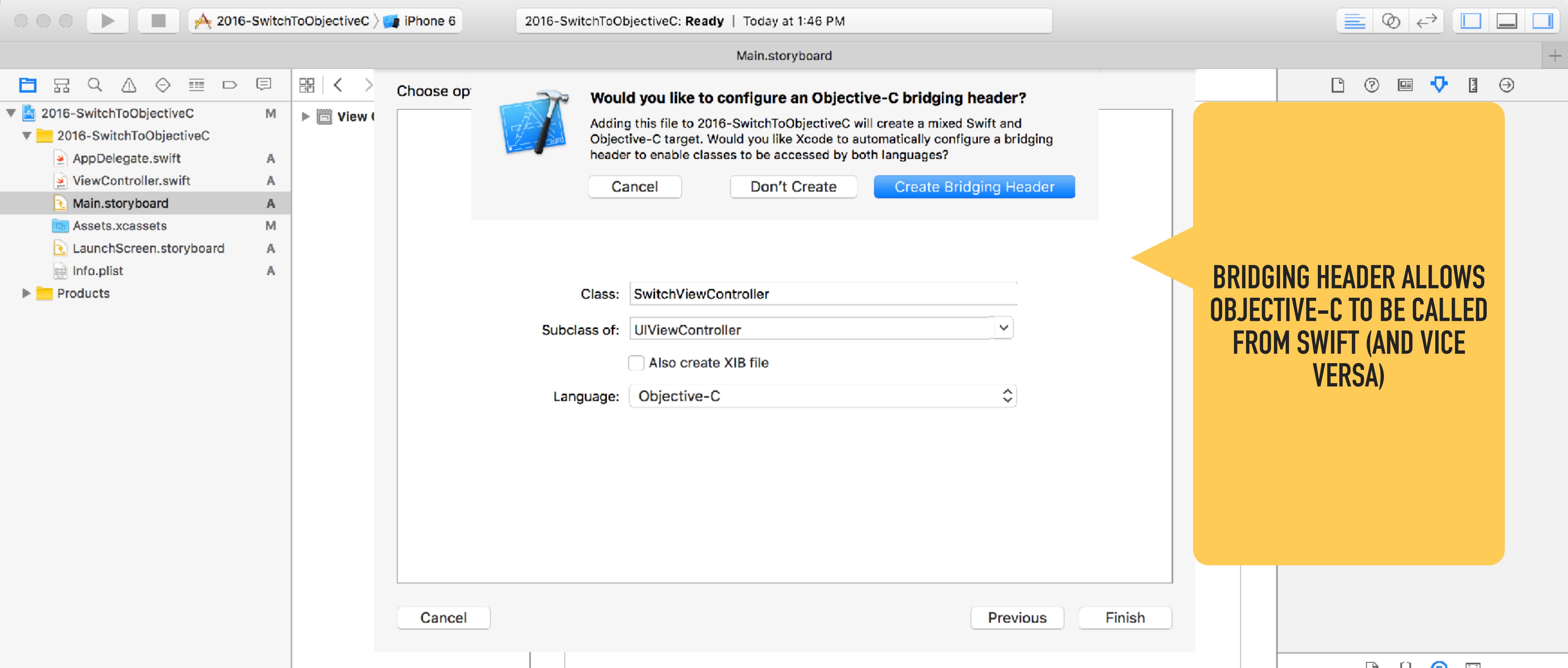
The screenshot shows two Xcode code editors side-by-side. The left editor displays `SwitchViewController.m` with Objective-C code. The right editor displays `SwitchViewController.h` with Objective-C header code. A yellow arrow points from the `tapSwitch:` method in the `.h` file to its corresponding implementation in the `.m` file.

```
2016-SwitchToObjectiveC 2016-SwitchToObjectiveC SwitchViewController.m No Selection
1 // SwitchViewController.h
2 // 2016-SwitchToObjectiveC
3 // Created by T. Andrew Binkowski on 4/3/16.
4 // Copyright © 2016 The University of Chicago, Department of Economics.
5
6 #import <UIKit/UIKit.h>
7
8 @interface SwitchViewController : UIViewController
9
10 - (IBAction)tapSwitch:(UISwitch *)sender;
11
12 @end

2016-SwitchToObjectiveC 2016-SwitchToObjectiveC SwitchViewController.m No Selection
1 // SwitchViewController.m
2 // 2016-SwitchToObjectiveC
3 // Created by T. Andrew Binkowski on 4/3/16.
4 // Copyright © 2016 The University of Chicago, Department of Economics.
5
6 #import "SwitchViewController.h"
7
8 @interface SwitchViewController () {
9     UISwitch *switchView;
10 }
11
12 @implementation SwitchViewController
13
14 - (void)viewDidLoad {
15     [super viewDidLoad];
16
17     // Do any additional setup after loading the view.
18     NSLog(@"The view did load");
19
20 }
21
22
23
24
25
26 // Toggle the values of the switch and store its values in `NSUserDefaults`
27 // Dump the user defaults to the console for debugging
28 - (IBAction)tapSwitch:(UISwitch *)sender {
29     NSLog(@"Switch Value:%d", sender.isOn);
30
31     NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
32     NSLog(@"Defaults: %@", defaults.dictionaryRepresentation);
33
34     [defaults setBool:sender.isOn forKey:@"kSwitchPreference"];
35     NSLog(@"Defaults: %@", defaults.dictionaryRepresentation);
36 }
37
38
39 @end
```

USING CUSTOM CLASSES

CREATE A PROJECT



CREATE A PROJECT

```
2016-SwitchToObjectiveC 2016-SwitchToObjectiveC GreetingObject.h @interface GreetingObject
2016-SwitchToObjectiveC 2016-SwitchToObjectiveC GreetingObject.m @implementation GreetingObject
```

```
// GreetingObject.h
// 2016-SwitchToObjectiveC
//
// Created by T. Andrew Binkowski on 4/3/16.
// Copyright © 2016 The University of Chicago, Department of Computer Science.

#import <Foundation/Foundation.h>

@interface GreetingObject : NSObject

@property (strong, nonatomic) NSString *message;
@property (strong, nonatomic) NSDate *today;

- (id)initWithMessage:(NSString*)message;
- (void)greetWithDate;

@end
```

```
// GreetingObject.m
// 2016-SwitchToObjectiveC
//
// Created by T. Andrew Binkowski on 4/3/16.
// Copyright © 2016 The University of Chicago, Department of Computer Science.

#import "GreetingObject.h"

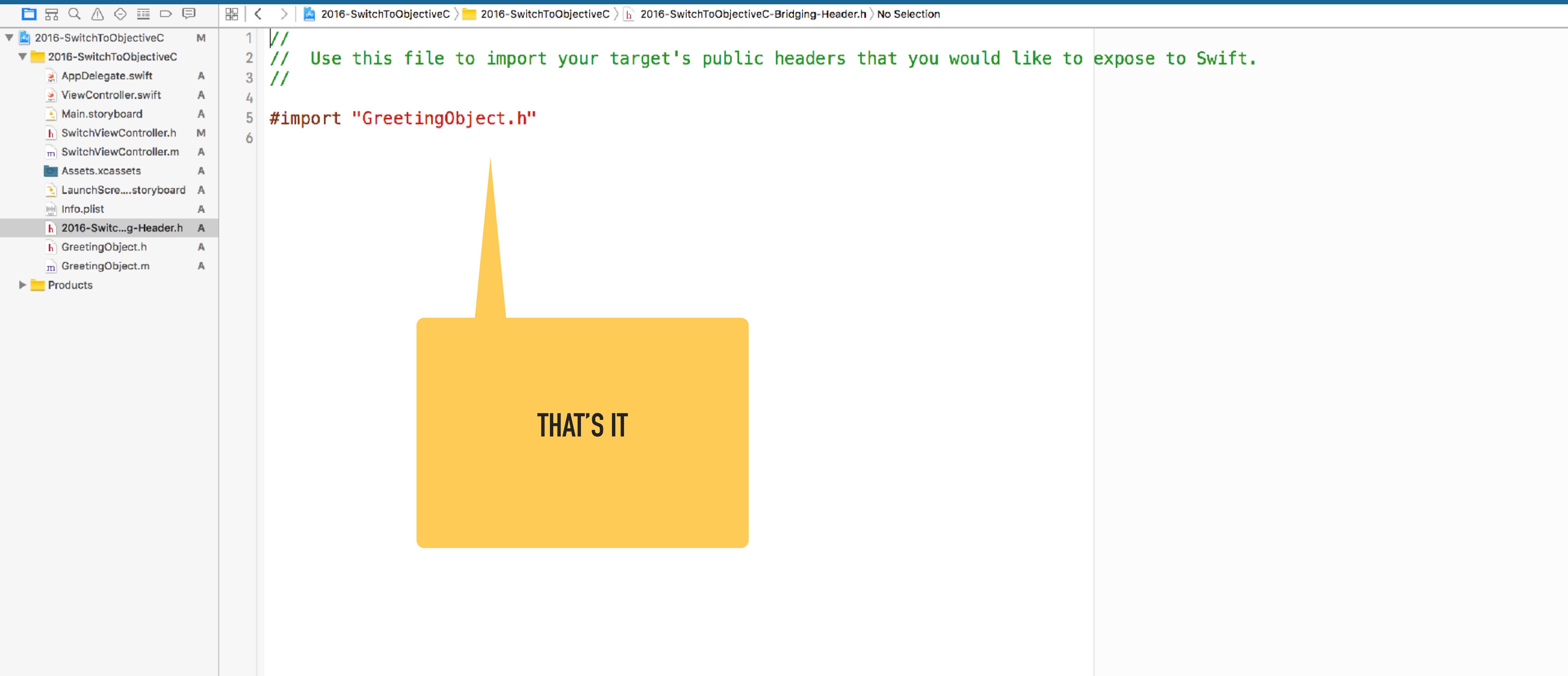
@implementation GreetingObject

- (id)initWithMessage:(NSString*)message {
    if ( self = [super init] ) {
        self.message = message;
        self.today = [NSDate date];
    }
    return self;
}

- (void)greetWithDate {
    NSLog(@"%@", self.message, self.today);
}
```

CUSTOM INITIALIZATION METHOD

CREATE A PROJECT



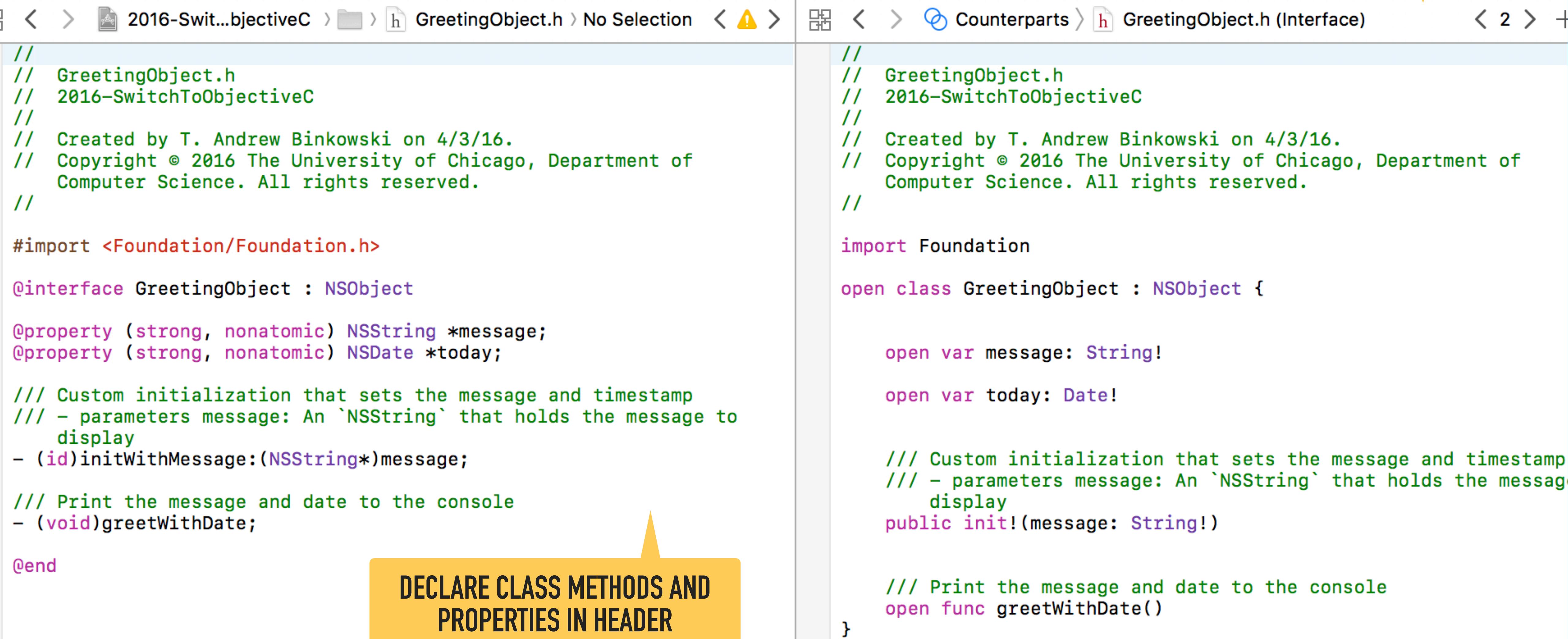
The screenshot shows the Xcode interface with a project named "2016-SwitchToObjectiveC". The left sidebar displays the project structure, including files like AppDelegate.swift, ViewController.swift, Main.storyboard, SwitchViewController.h, SwitchViewController.m, Assets.xcassets, LaunchScreen.storyboard, Info.plist, and two files named "2016-Swift...g-Header.h" (highlighted in grey) and "GreetingObject.h". The main editor area shows a bridging header file "2016-SwitchToObjectiveC-Bridging-Header.h" containing the following code:

```
// Use this file to import your target's public headers that you would like to expose to Swift.  
//  
#import "GreetingObject.h"
```

A large yellow speech bubble graphic is overlaid on the bottom left of the editor area, containing the text "THAT'S IT".

CREATE A PROJECT

THE BRIDGING HEADER CONVERTS
TO SWIFT (DOES NOT ACTUALLY
GENERATE THE FILE)



```
// GreetingObject.h
// 2016-SwitchToObjectiveC
//
// Created by T. Andrew Binkowski on 4/3/16.
// Copyright © 2016 The University of Chicago, Department of
// Computer Science. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface GreetingObject : NSObject

@property (strong, nonatomic) NSString *message;
@property (strong, nonatomic) NSDate *today;

/// Custom initialization that sets the message and timestamp
/// - parameters message: An `NSString` that holds the message to
///   display
- (id)initWithMessage:(NSString*)message;
/// Print the message and date to the console
- (void)greetWithDate;

@end

// GreetingObject.h
// 2016-SwitchToObjectiveC
//
// Created by T. Andrew Binkowski on 4/3/16.
// Copyright © 2016 The University of Chicago, Department of
// Computer Science. All rights reserved.
//

import Foundation

open class GreetingObject : NSObject {

    open var message: String!
    open var today: Date!

    /// Custom initialization that sets the message and timestamp
    /// - parameters message: An `NSString` that holds the message to
    ///   display
    public init!(message: String!)

    /// Print the message and date to the console
    open func greetWithDate()
}
```

DECLARE CLASS METHODS AND PROPERTIES IN HEADER

CREATE A PROJECT

```
class ViewController: UIViewController {  
  
    //  
    // MARK: - Lifecycle  
    //  
  
    override func viewDidAppear(_ animated: Bool) {  
        super.viewDidAppear(animated)  
  
        // Load up the defaults and print them the console (we should be able to see  
        // that we changed the values from Objective-C  
        let defaults = UserDefaults.standard  
        print("Defaults (from Swift ViewController):\n \(defaults.dictionaryRepresentation())")  
  
        // Call our objective-c code  
        let message = GreetingObject(message: "This is from Swift")  
        message?.greetWithDate()  
  
    }  
  
}
```

- `(id)initWithMessage:(NSString*)message;`
- `(void)greetWithDate;`

CREATE A PROJECT

```
let d = Dog()
let numberOfDogs = d.number(ofDogs: 4)
print("Number of Dogs: \(numberOfDogs)")
```

```
1 // 
2 // Dog.m
3 // 2018-SwitchToObjectiveC
4 //
5 // Created by T. Andrew Binkowski on 4/9/18.
6 // Copyright © 2018 The University of Chicago,
    Department of Computer Science. All rights reserved.
7 //
8
9 #import "Dog.h"
10
11 @implementation Dog
12 - (int)numberOfDogs:(int)number {
13     return number;
14 }
15 @end
16
```

```
1 // 
2 // Dog.h
3 // 2018-SwitchToObjectiveC
4 //
5 // Created by T. Andrew Binkowski on 4/9/18.
6 // Copyright © 2018 The University of Chicago,
    Department of Computer Science. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10
11 @interface Dog : NSObject
12 @property (strong, nonatomic) NSString *message;
13
14 - (int)numberOfDogs:(int)number;
15
16 @end
17
```

CREATE A PROJECT

- https://developer.apple.com/library/content/documentation/Swift/Conceptual/BuildingCocoaApps/InteractingWithObjective-CAPIs.html#/apple_ref/doc/uid/TP40014216-CH4-ID35

```
@property (nullable) id nullableProperty;  
@property (nonnull) id nonNullProperty;  
@property id unannotatedProperty;  
  
var nullableProperty: Any?  
var nonNullProperty: Any  
var unannotatedProperty: Any!
```



ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 2B