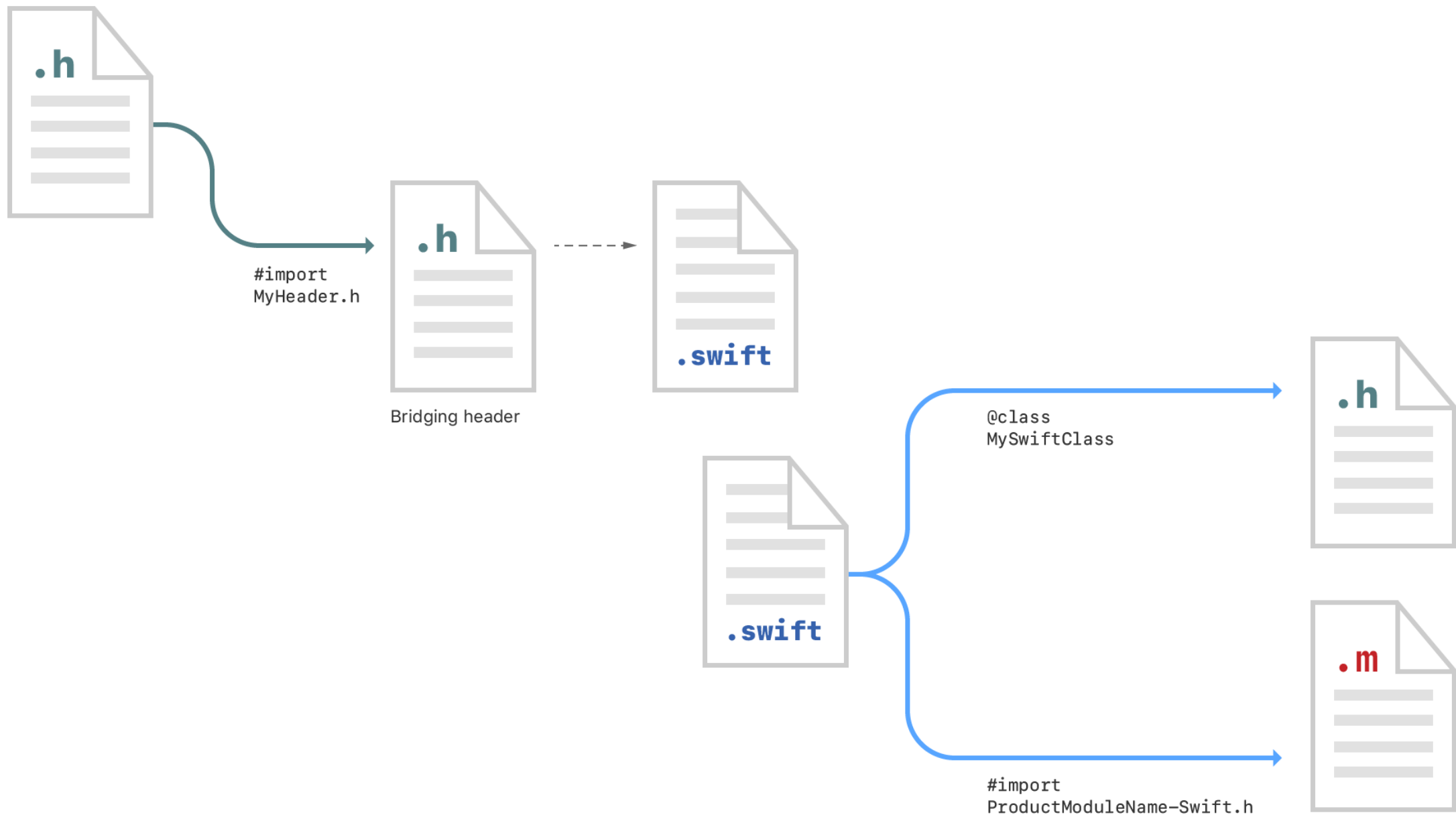


# Swift and Objective-C



# Bridging-header file

Choose a target

iOS

wa

Source



## Would you like to configure an Objective-C bridging header?

Adding this file to Test will create a mixed Swift and Objective-C target. Would you like Xcode to automatically configure a bridging header to enable classes to be accessed by both languages?

Cancel

Don't Create

Create Bridging Header

```
1 //  
2 // Use this file to import your target's public headers that you would like to expose to Swift.  
3 //  
4  
5
```

# Objective-C to Swift

```
1  //
2  //  Use this file to import your target's public headers that you would like to expose to Swift.
3  //
4
5  #import "AppDelegate.h"
6  #import "ViewController.h"
7
```

```
1  //
2  //  MyClass.swift
3  //  Test
4  //
5  //  Created by Aleksandr Sychev on 10/03/2019.
6  //  Copyright © 2019 Aleksandr Sychev. All rights reserved.
7  //
8
9  import Foundation
10
11 class Swifty : NSObject {
12     let viewController = ViewController()
13 }
14
```

# Swift to Objective-C

▼ <b>Packaging</b>	
Setting	Test
Convert Copied Files	No ⚙
Create Info.plist Section in Binary	No ⚙
Defines Module	No ⚙
Executable Extension	
Executable Prefix	
Expand Build Settings in Info.plist File	Yes ⚙
Force Package Info Generation	Yes ⚙
Framework Version	A
<b>Info.plist File</b>	<b>Test/Info.plist</b>
Info.plist Other Preprocessor Flags	
Info.plist Output Encoding	binary ⚙
► Info.plist Preprocessor Definitions	
Info.plist Preprocessor Prefix File	
Module Map File	
Preprocess Info.plist File	No ⚙
Preserve HFS Data	No ⚙
Private Headers Folder Path	Test.app/PrivateHeaders
Private Module Map File	
<b>Product Bundle Identifier</b>	<b>com.test.Test</b>
► <b>Product Module Name</b>	<b>Test</b>
<b>Product Name</b>	<b>Test</b>
Property List Output Encoding	binary ⚙
Public Headers Folder Path	Test.app/Headers
Strings File Output Encoding	binary ⚙
Wrapper Extension	app

# Swift to Objective-C

```
1  //
2  //  ViewController.m
3  //  Test
4  //
5  //  Created by Aleksandr Sychev on 10/03/2019.
6  //  Copyright © 2019 Aleksandr Sychev. All rights reserved.
7  //
8
9  #import "ViewController.h"
10
11  #import "Test-Swift.h"
12
13  @interface ViewController ()
14
15  - (Swift *)mySwiftObject;
16
17  @end
```

# Swift to Objective-C

```
1  //
2  // AppDelegate.h
3  // Test
4  //
5  // Created by Aleksandr Sychev on 10/03/2019.
6  // Copyright © 2019 Aleksandr Sychev. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10
11  @class Swifty;
12
13  @interface AppDelegate : UIResponder <UIApplicationDelegate>
14
15  @property (strong, nonatomic) UIWindow *window;
16
17  @end
```

# Swift to Objective-C

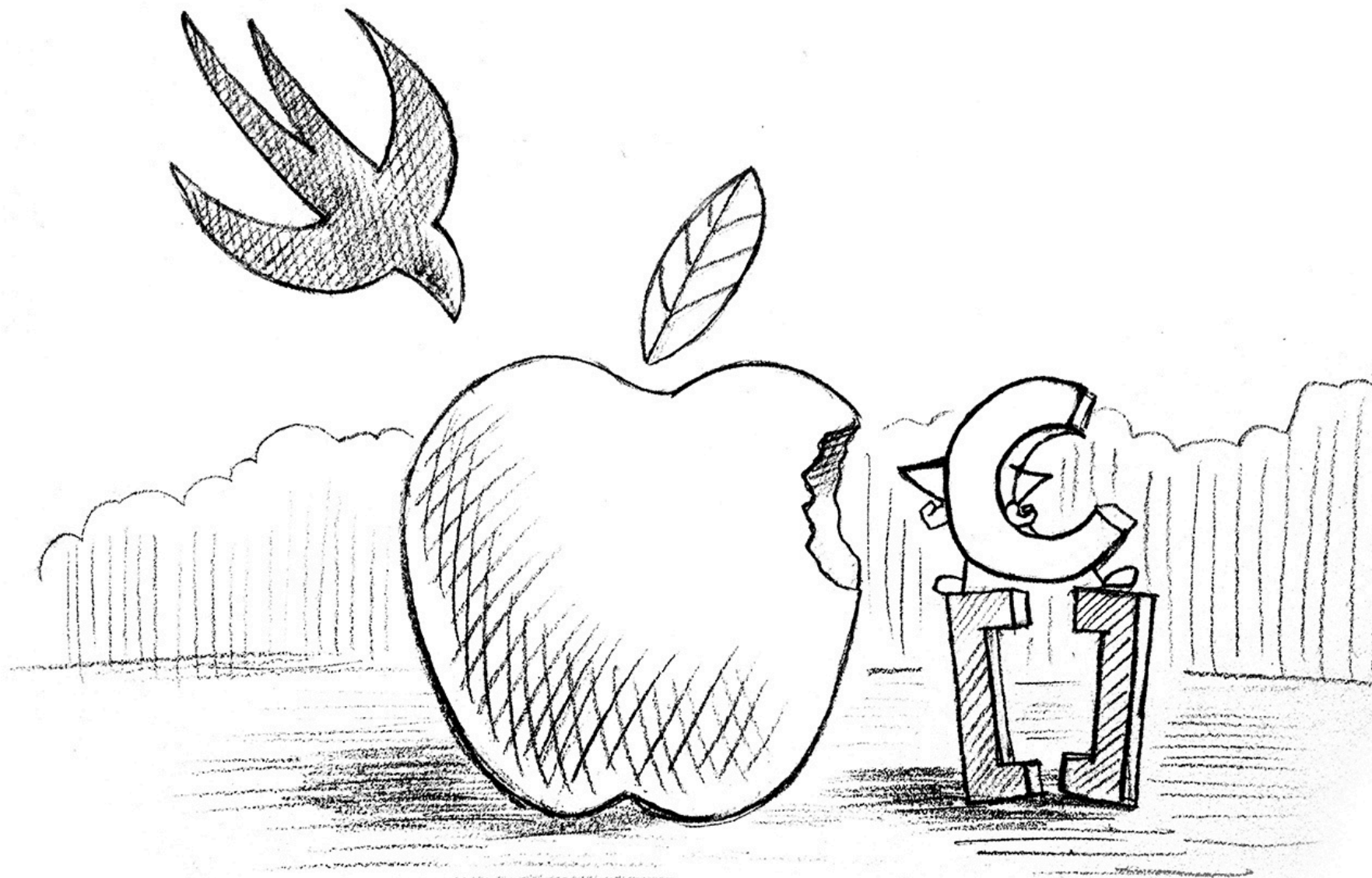
*A Swift class must be a descendant of an Objective-C class to be accessible and usable in Objective-C.*



# Swift to Objective-C

```
185 SWIFT_CLASS( "_TtC4Test19SwiftyForObjectiveC" )
186 @interface SwiftyForObjectiveC : NSObject
187 - (NSInteger)sum SWIFT_WARN_UNUSED_RESULT;
188 - (nonnull instancetype)init OBJC_DESIGNATED_INITIALIZER;
189 @end
```

**Пример**



# Objective-C Types in Swift

Objective-C Type	Swift Equivalent
BOOL	Bool
NSInteger	Int
SEL	Selector
id	AnyObject!
Class	AnyClass!
NSString *	String!
NSArray *	AnyObject[]!

# Objective-C Nullability

```
@interface MyList : NSObject
- (MyListItem *)itemWithName:(NSString *)name;
- (NSString *)nameForItem:(MyListItem *)item;
@property (copy) NSArray<MyListItem *> *allItems;
@end
```

```
class MyList: NSObject {
    func item(withName name: String!) -> MyListItem!
    func name(for item: MyListItem!) -> String!
    var allItems: [MyListItem]!
}
```

# Objective-C Nullability

- **Nonnullable** — as nonoptionals
- **Nullable** — as optionals
- Without a nullability or with a **null\_resettable** — as implicitly unwrapped optionals

# Objective-C Nullability

```
@interface MyList : NSObject
- (nullable MyListItem *)itemWithName:(nonnull NSString *)name;
- (nullable NSString *)nameForItem:(nonnull MyListItem *)item;
@property (copy, nonnull) NSArray<MyListItem *> *allItems;
@end
```

```
class MyList: NSObject {
    func item(withName name: String) -> MyListItem?
    func name(for item: MyListItem) -> String?
    var allItems: [MyListItem]
}
```

# Objective-C Nullability

**\_Nullable id \* \_Nonnull**



# Objective-C Nullability

```
NS_ASSUME_NONNULL_BEGIN
```

```
@interface MyList : NSObject  
- (nullable MyListItem *)itemWithName:(NSString *)name;  
- (nullable NSString *)nameForItem:(MyListItem *)item;  
@property (copy) NSArray<MyListItem *> *allItems;  
@end
```

```
NS_ASSUME_NONNULL_END
```

# Read

- <https://medium.com/ios-os-x-development/swift-and-objective-c-interoperability-2add8e6d6887>
- <https://developer.apple.com/swift/blog/?id=25>
- [https://developer.apple.com/documentation/swift/objective-c and c code customization](https://developer.apple.com/documentation/swift/objective-c_and_c_code_customization)
- [https://developer.apple.com/documentation/swift/imported c and objective-c apis](https://developer.apple.com/documentation/swift/imported_c_and_objective-c_apis)
- <https://www.hackingwithswift.com/articles/114/objective-c-to-swift-conversion-cheat-sheet>
- <https://youtu.be/WL9i7tBf3vk>