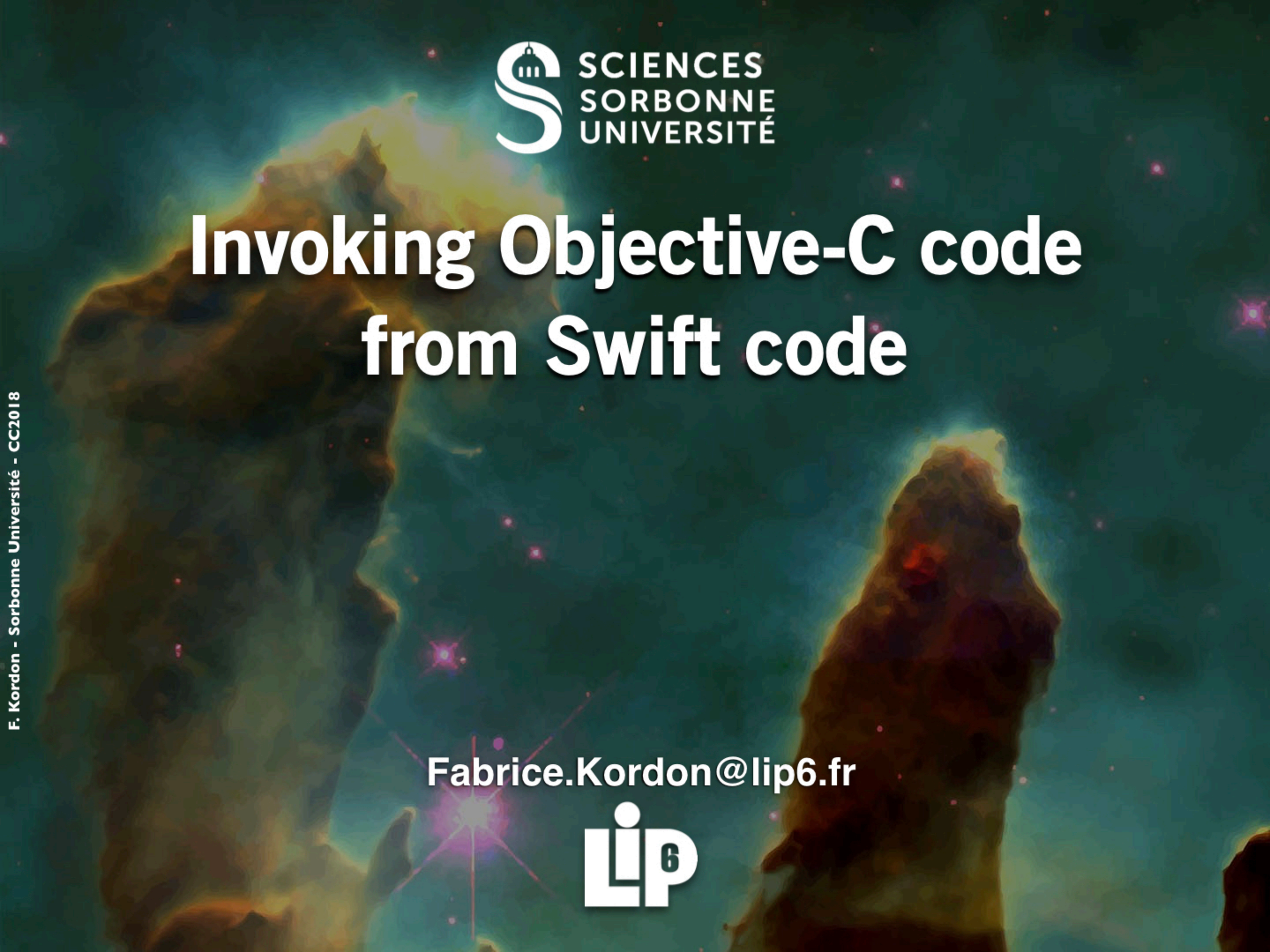

SCIENCES
SORBONNE
UNIVERSITÉ
Invoking Objective-C code
from Swift code
F. Kordon - Sorbonne Université - CC2018
Fabrice.Kordon@lip6.fr
LIP6

As an introduction

Presentation by means of an example

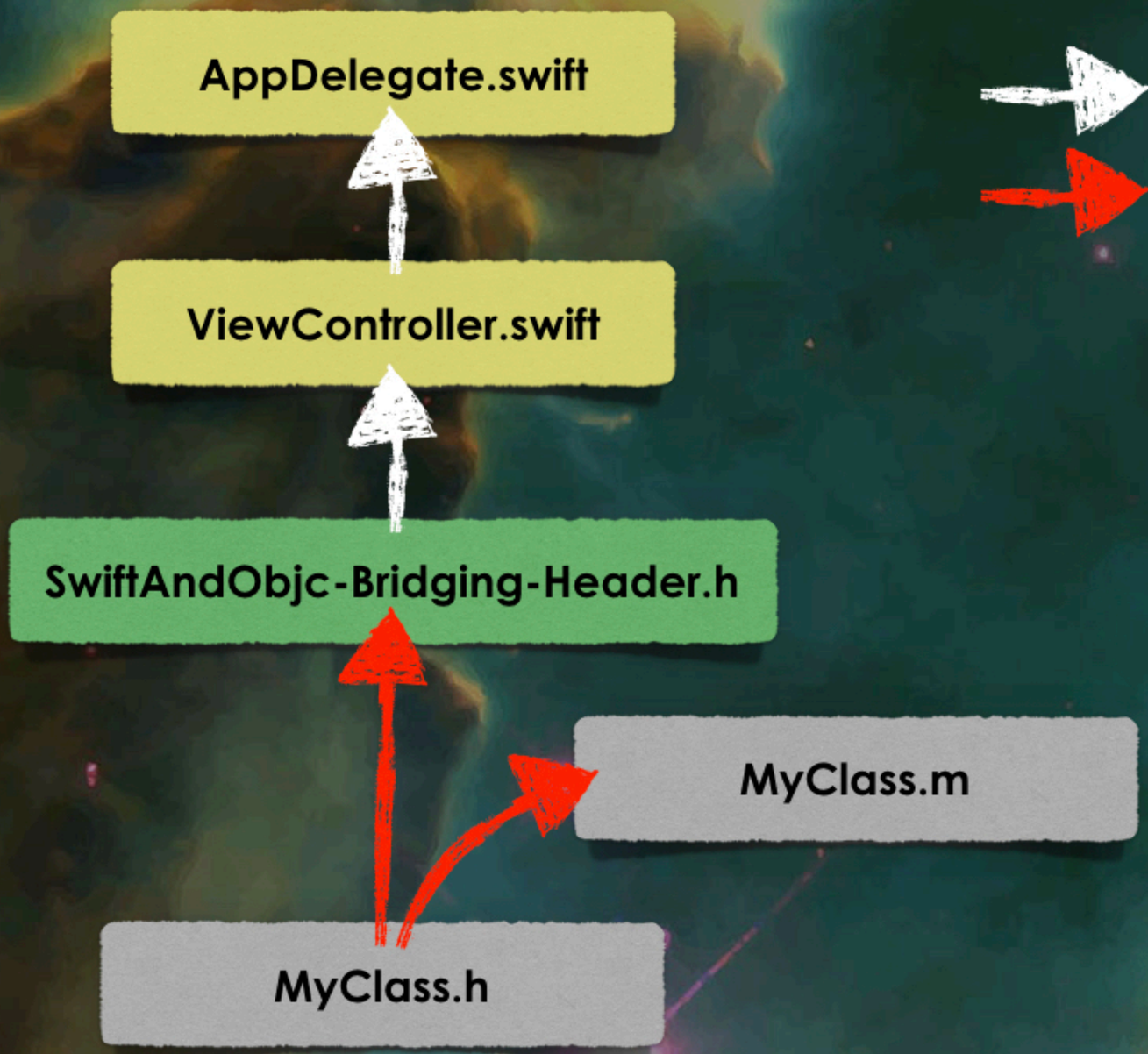
- Creation of objects written in Objective-C
- Inside a Swift project

Operations to complete

- Init this class
- Show various access to attributes
- Show method invocation

Demo

Architecture

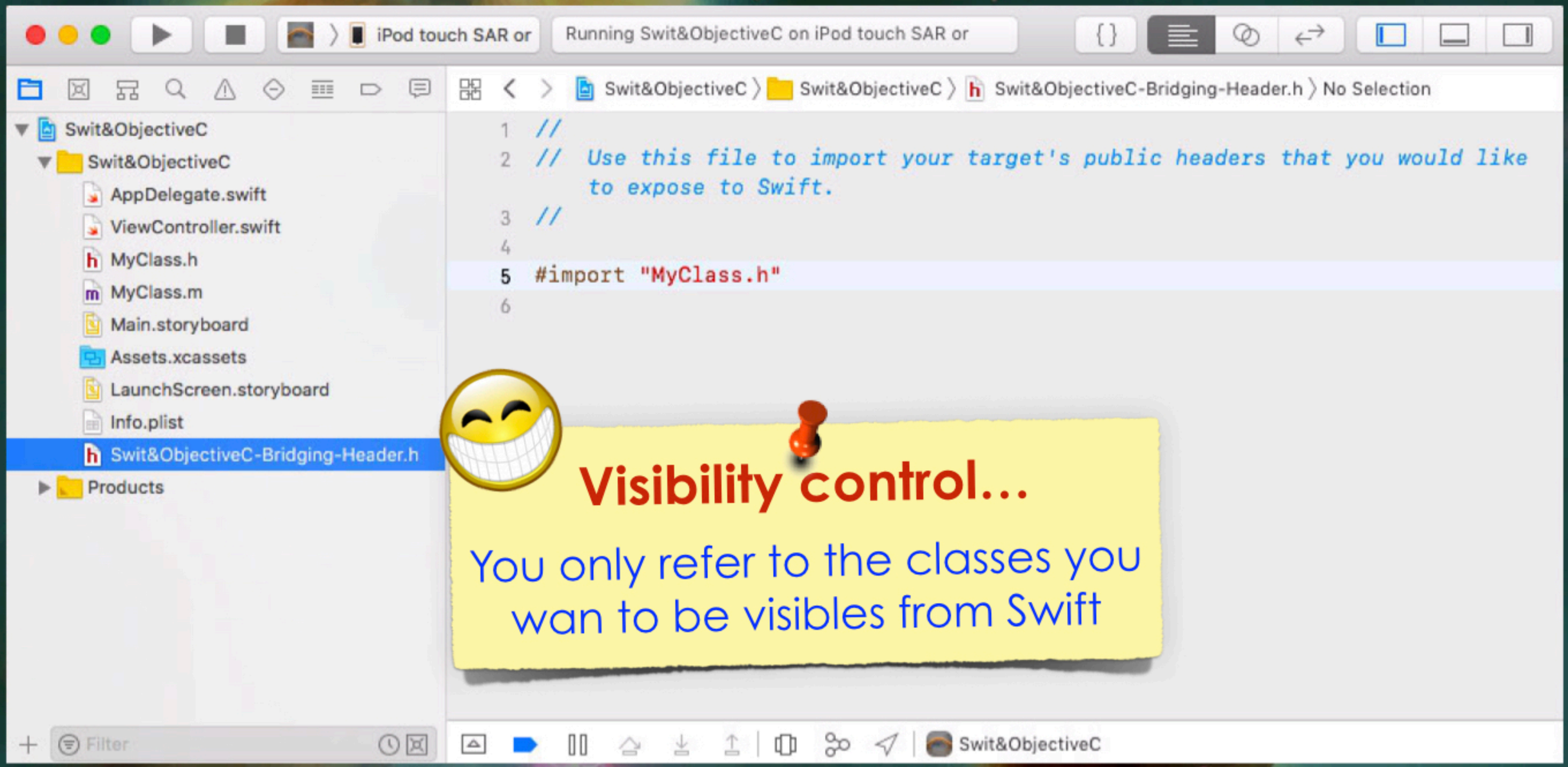


implicit liaison

Explicit liaison (connection)

Creating the bridge

The bridge file



MyClass


```objc
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h> // Added for UIColor

NS_ASSUME_NONNULL_BEGIN

@interface MyClass : NSObject

    @property (readwrite, strong, nonatomic) NSString *myValue;
    @property (readonly, strong, nonatomic) UIColor *myColor;

- (id) initWithValue:(NSString *)initValue andColor:(UIColor *) col;
- (NSString*) compute;

@end

NS_ASSUME_NONNULL_END
```

MyClass


```objc
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h> // Added for UIColor

NS_ASSUME_NONNULL_BEGIN

@interface MyClass : NSObject

    @property (readwrite, strong, nonatomic) NSString *myValue;
    @property (readonly, strong, nonatomic) UIColor *myColor;

- (id) initWithValue:(NSString *)initValue andColor:(UIColor *) col;
- (NSString*) compute;

@end

NS_ASSUME_NONNULL_END
```

```objc
#import "MyClass.h"

@implementation MyClass

- (id) initWithValue:(NSString *)initValue andColor:(UIColor *)col {
    if ([super init]) {
        _myValue = [initValue copy];
        _myColor = [col copy];
    }
    return self;
}

- (NSString*) compute {
    return [NSString stringWithFormat:@"Computed : %@", _myValue];
}

@end
```

ViewController


```
import UIKit

class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        self.view = UIView()
        self.view.backgroundColor = UIColor.white
        let lab1 = UILabel(frame: CGRect(x: 0.0, y: 40.0,
                                         width: UIScreen.main.bounds.width,
                                         height: 30.0))
        lab1.textAlignment = .center
        lab1.textColor = UIColor.blue
        lab1.font = UIFont.boldSystemFont(ofSize: 14.0)

        let lab2 = UILabel(frame: CGRect(x: 0.0, y: 80.0,
                                         width: UIScreen.main.bounds.width,
                                         height: 30.0))
        lab2.textAlignment = .center
        lab2.textColor = UIColor.red
        lab2.font = UIFont.boldSystemFont(ofSize: 14.0)

        let obj1 = MyClass(value: "I am in the Objective-C part",
                           andColor: .red)
        lab1.textColor = obj1.myColor
        lab1.text = obj1.compute() // using the getter

        let obj2 = MyClass()
        obj2.myValue = "Hello from Objective-C"
        lab2.textColor = .blue // myColor attribute is not set
        lab2.text = obj2.myValue // accessing the property

        self.view.addSubview(lab1)
        self.view.addSubview(lab2)
    }
}
```

As a conclusion…

Easy is'n't it?

- Important for reusing Existing Objective-C code
  - **Until it is being ported?**

One more thing…

What about the reverse operation? similar principles

- Less interesting a priori
- Main difference, the bridge is automatic
  - **ARC can be deactivated (for the Objective-C part)**
- There is too a bridge to expose Objective-C class to Swift
  - **Only for your own Objective-C classes**
  - **Those the Swift code is using**
- In the Swift code
  - **Directive @objc to expose entities to Objective-C**
  - **Already required for methods to be called-back**

```swift
import Foundation

@objc class MyClass: NSObject {// Do not forget @objc

    var myValue = ""

    func compute () -> String {
        return "Computed : " + myValue
    }
}
```

One more thing…

What about the reverse operation? similar principles

- Less interesting a priori
- Main difference, the bridge is automatic
  - **ARC can be [illegible]ated (for the Objective-C part)**
- There is too [illegible]ft
  - **Only for your**
  - **Those the Sw**
- In the Swift c[illegible]
  - **Directive @objc to expose entities to Objective-C**
  - **Already required for methods to be called-back**

From Objective-C?

It works as usual (this is the idea)

```
import Foundation

@objc class MyClass: NSObject {// Do not forget @objc

    var myValue = ""

    func compute () -> String {
        return "Computed : " + myValue
    }
}
```