

# Workshop :: Automated testing





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata



Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

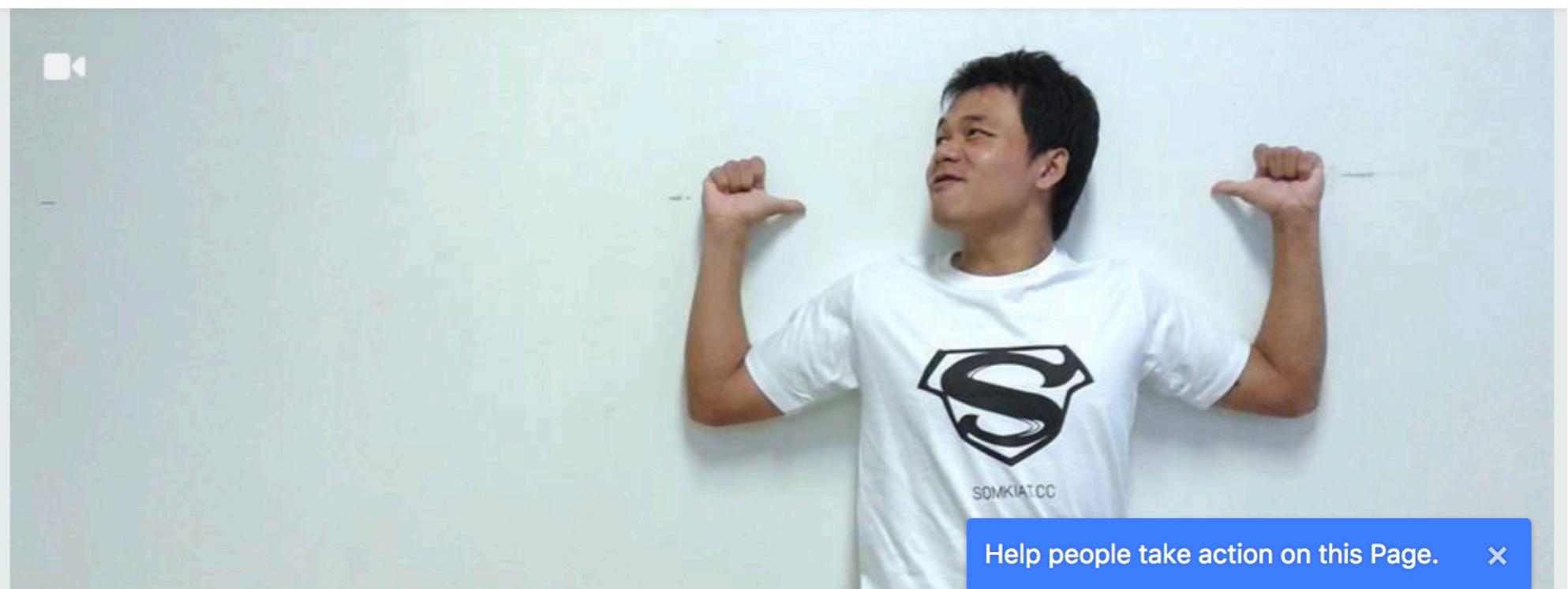
@somkiat.cc

Home

Posts

Videos

Photos



# Agenda

- Introduction of testing
- Why we need to test ?
- Types of tests
- Testing pyramid concept
- iOS app testing
- Workshop (step-by-step)



**<https://github.com/up1/workshop-ios-testing>**



# How to write first unit tests ?

Working with XCTest

Add tests to your project

Running unit tests in XCode

Write unit tests

<https://developer.apple.com/documentation/xctest>



# Create project with Unit tests

Choose options for your new project:

Product Name: MyMovies

Team: Somkiat Puisungnoen (Personal Team) 

Organization Name: Somkiat Puisungnoen

Organization Identifier: test 

Bundle Identifier: test.MyMovies

Language: Swift 

Use Core Data

Include Unit Tests

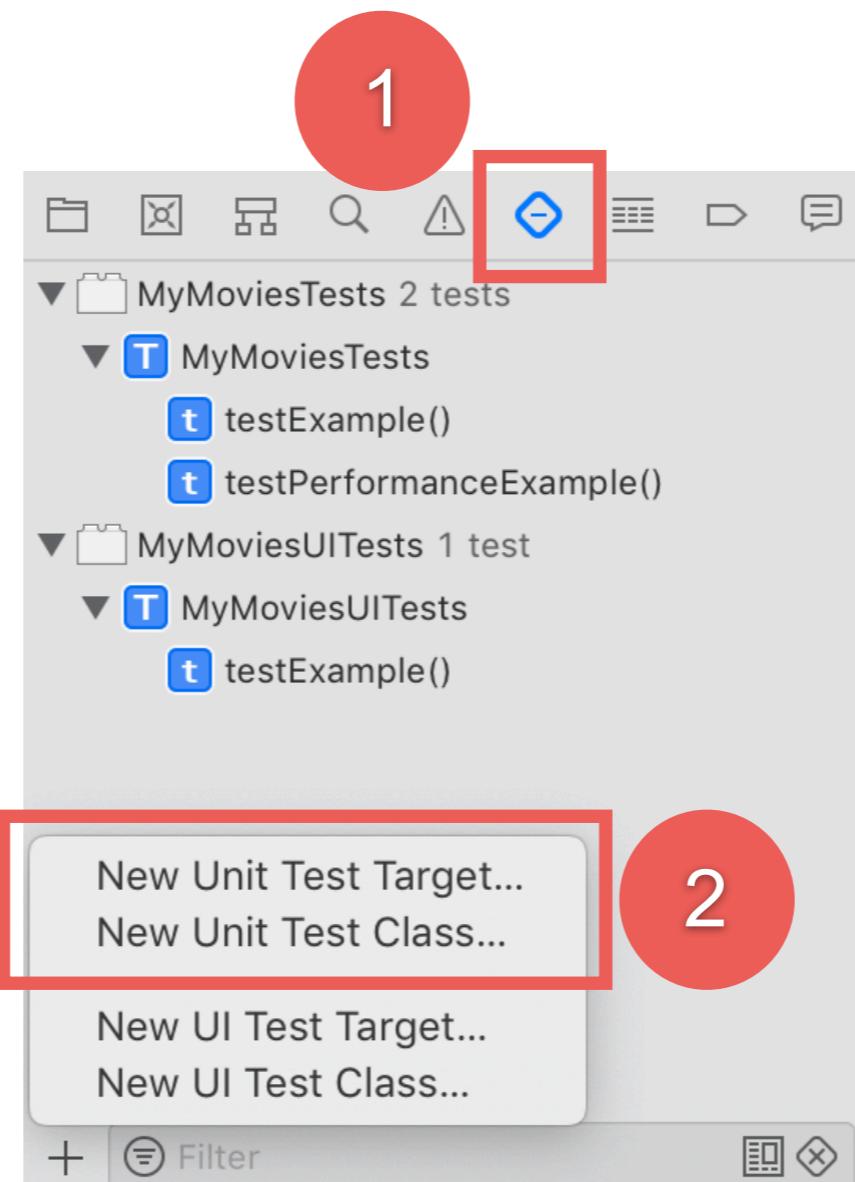
Include UI Tests



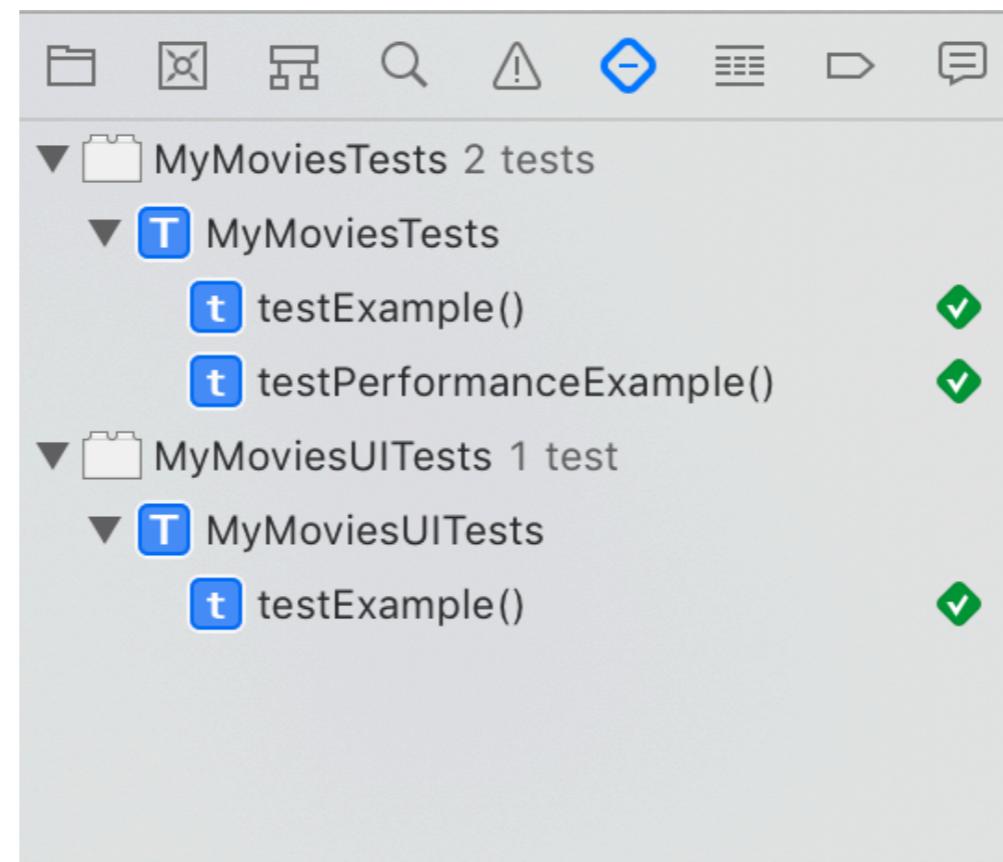
# Add unit tests to existing project

Test navigation -> New Unit Test Target



# Running unit tests in XCode

Product -> Test (command + U)



# What should you test first ?



# What should you test first ?

Requirements ?

You won't be able to test all the requirements



# What should you test first ?

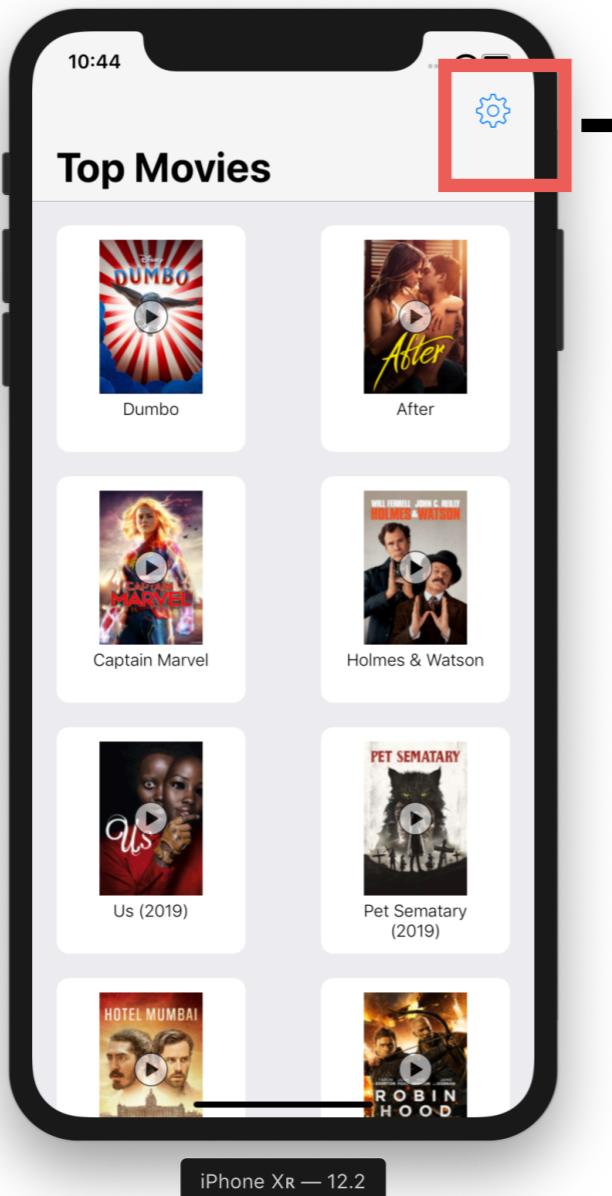
But if you can test some of them,  
You'll able to quickly verify your app meet  
requirement



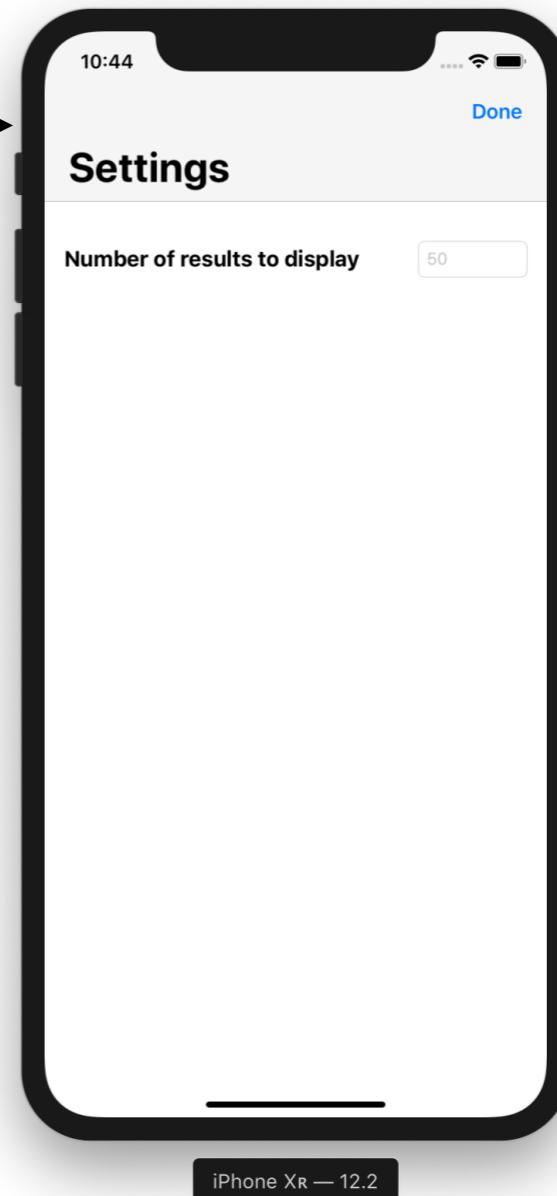
# Requirements(1)

## My Movies App from iTunes

Tops movies page



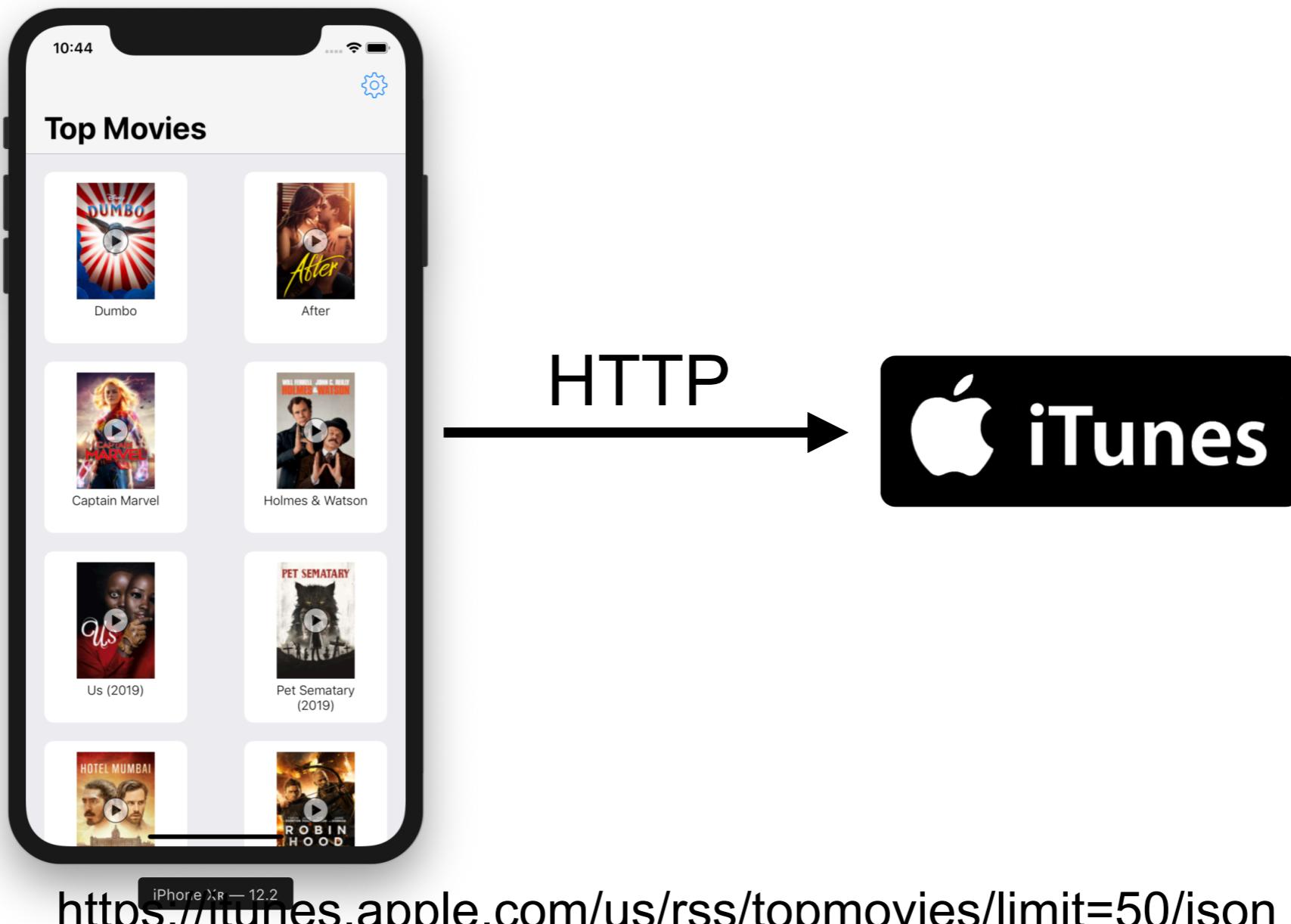
Settings page



# Requirements(2)

## My Movies App from iTunes

Tops movies page

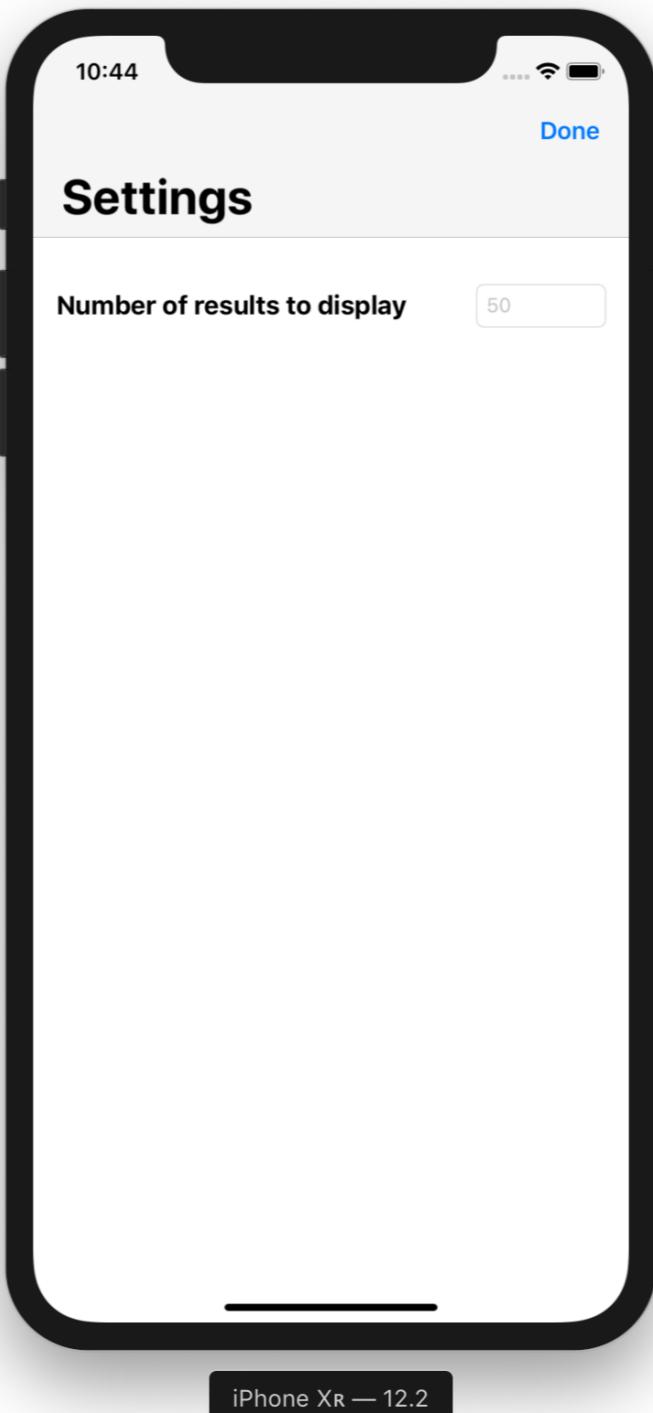


# Let's start with easy part

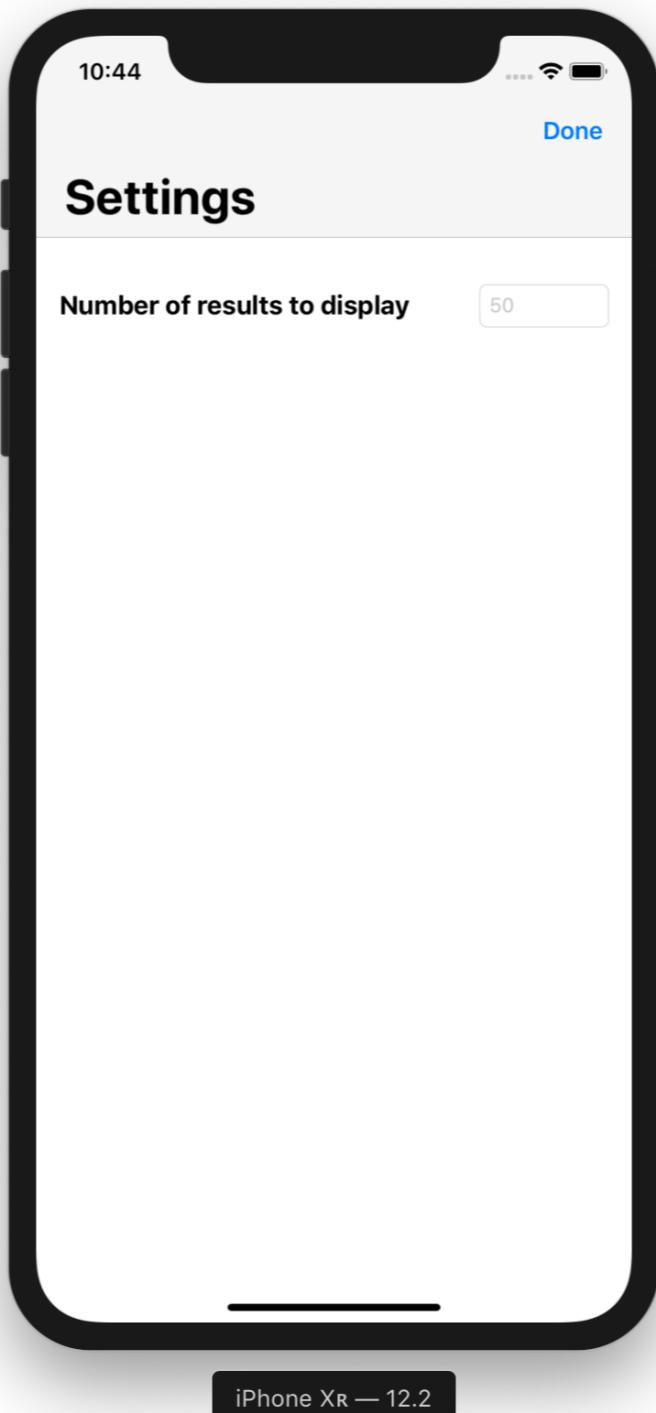
## Settings page



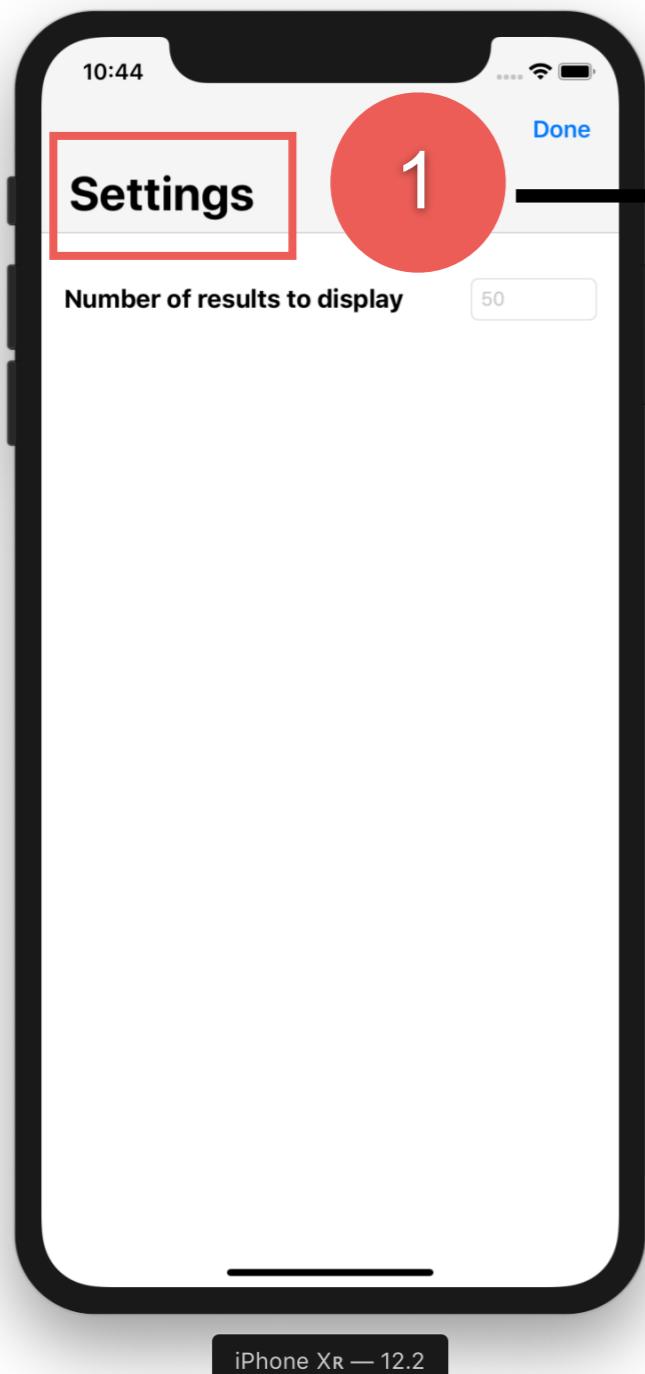
# Settings page



# What should we test ?



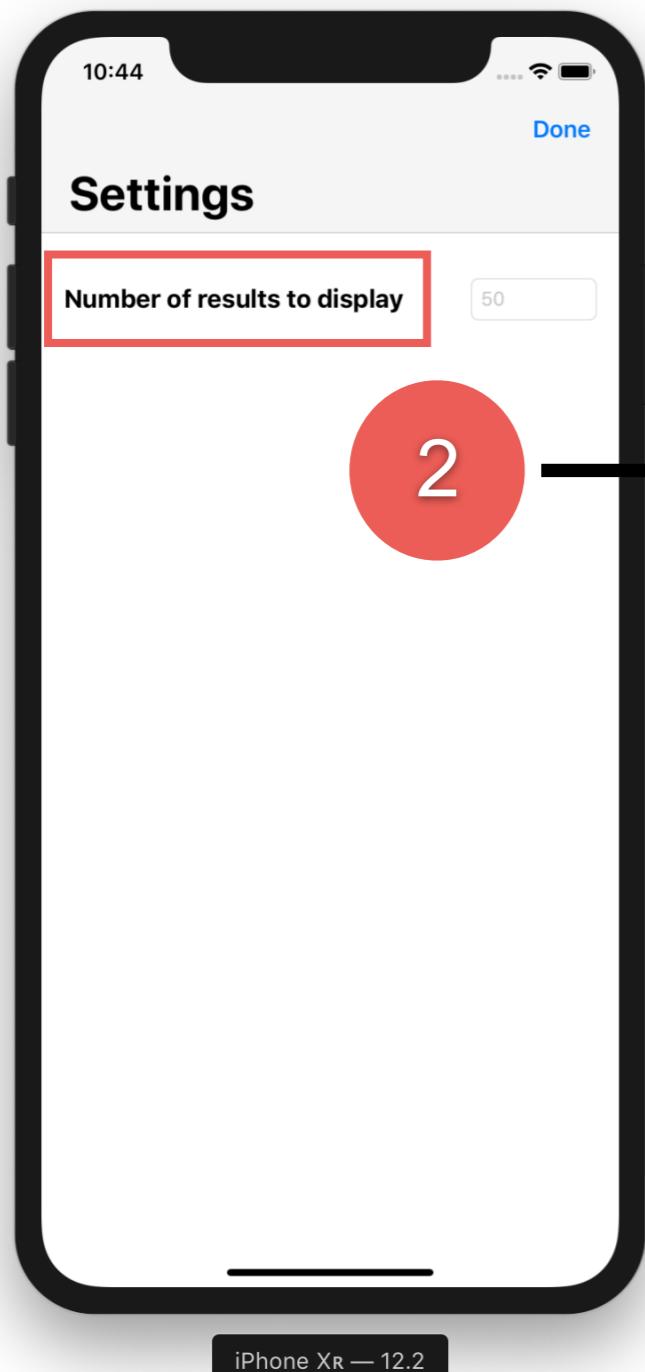
# What should we test ?



Title of screen is **Settings**



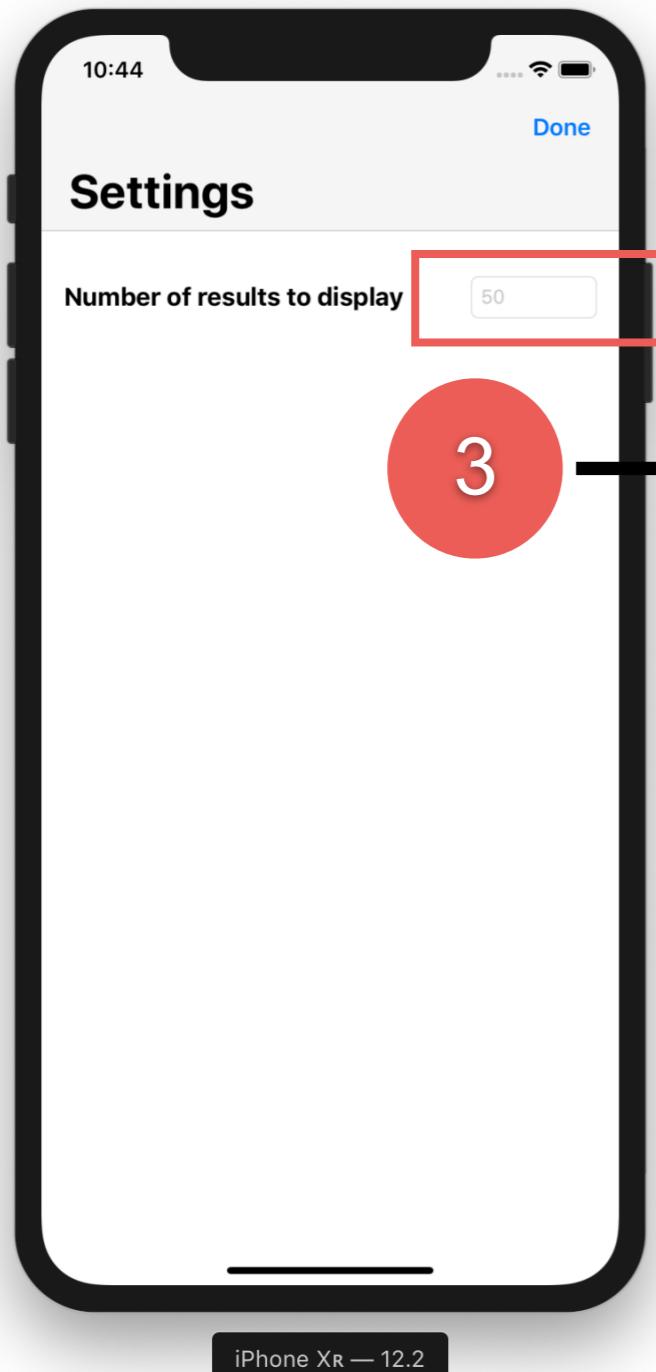
# What should we test ?



Label should display  
**“Number of results to display”**



# What should we test ?



Verify placeholder text in text field  
Should be **50**



# Write tests...

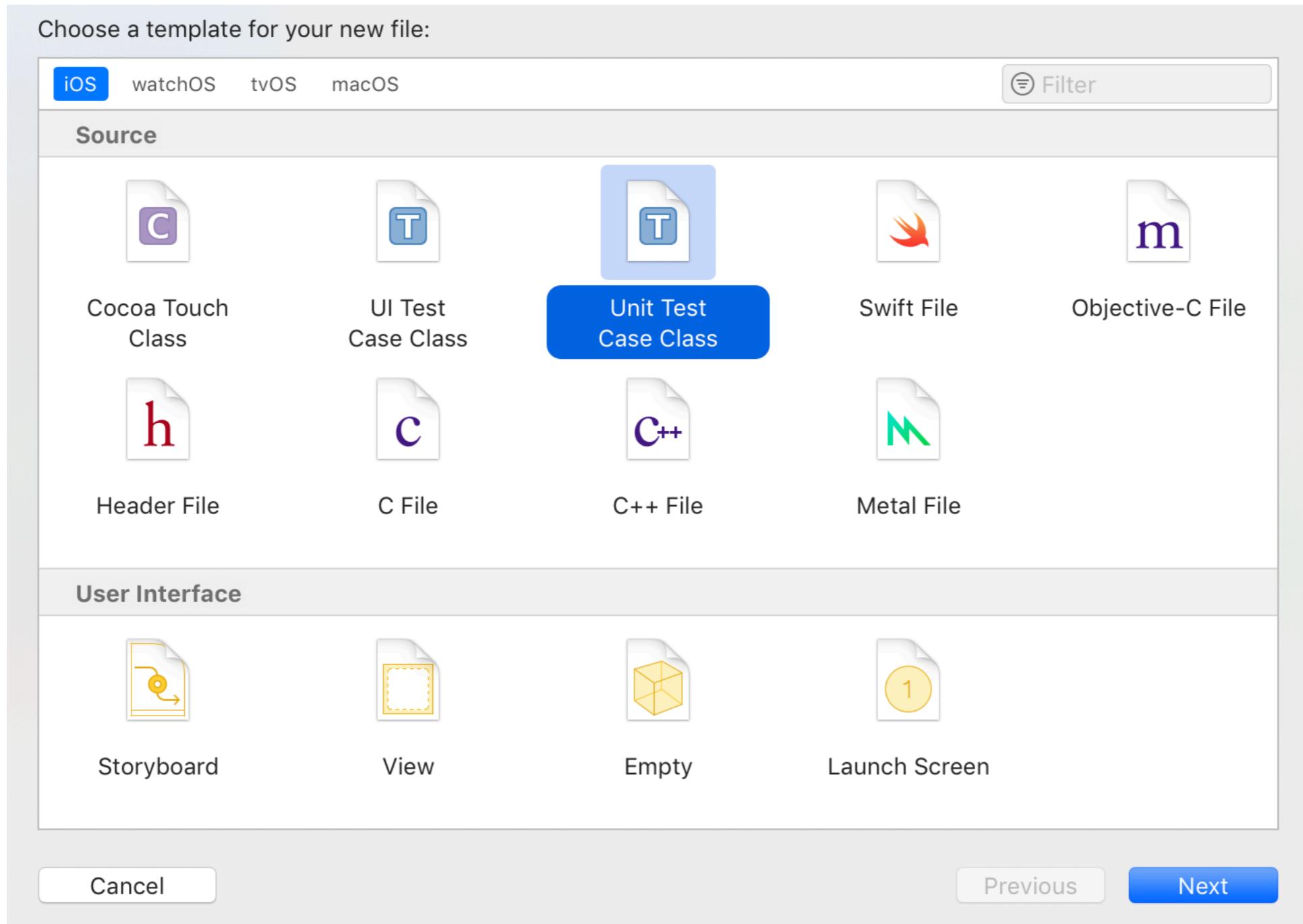


# Create Unit Tests

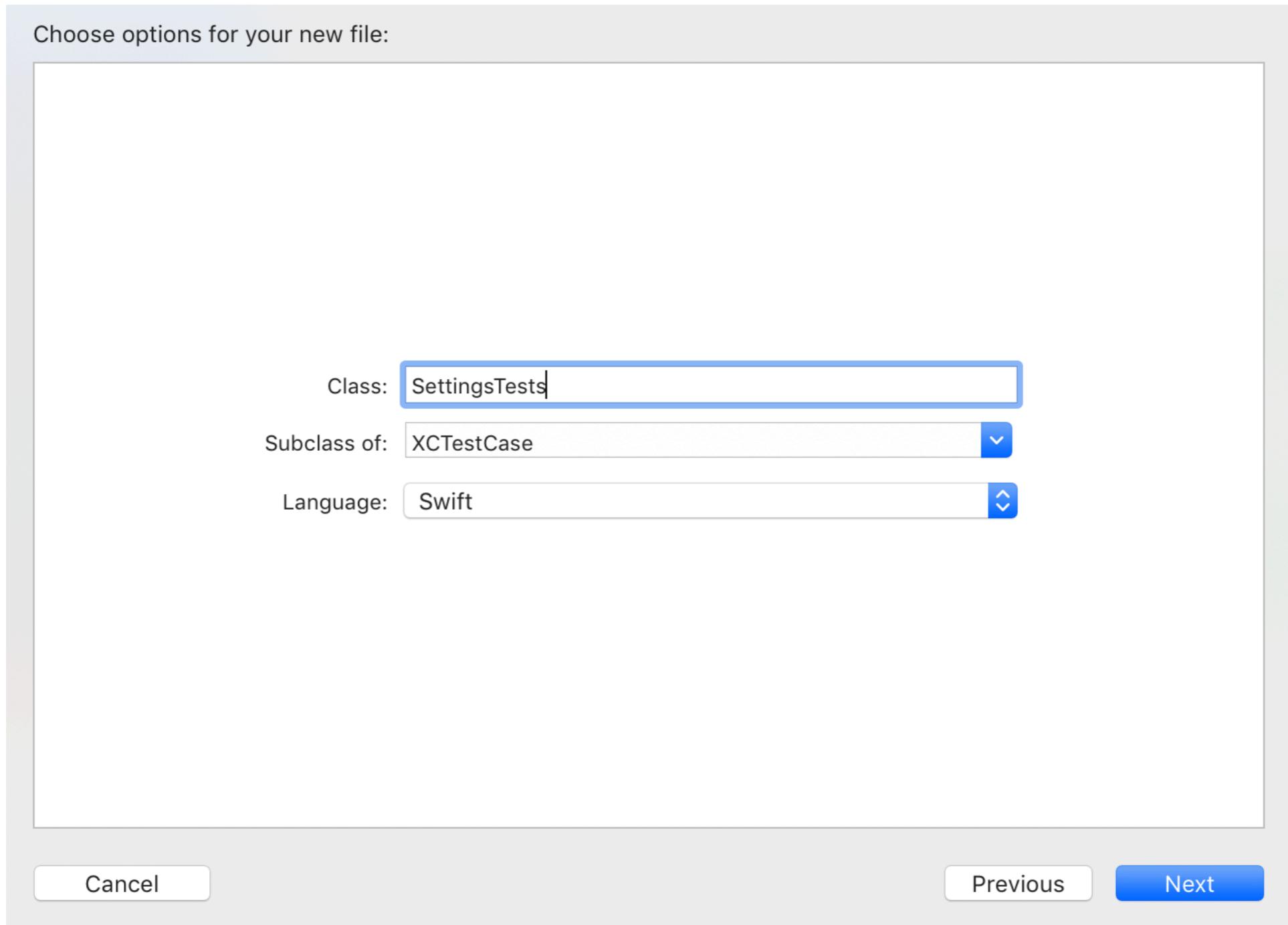
File name = SettingsTests.swift



# Create Unit Tests (1)



# Create Unit Tests (2)



# Ready !!

```
import XCTest
@testable import MyMovies

class SettingsTests: XCTestCase {

    func testExample() {
        // Arrange

        // Act

        // Assert

    }

}
```



# First test case

## Title of screen is **Settings**

```
class SettingsTests: XCTestCase {

    func test_title_is_Settings() {
        // Arrange
        let storyboard = UIStoryboard(name: "Main", bundle: nil)

        // Act
        let vc = storyboard.instantiateViewController(withIdentifier:
            "Settings")

        let _ = vc.view

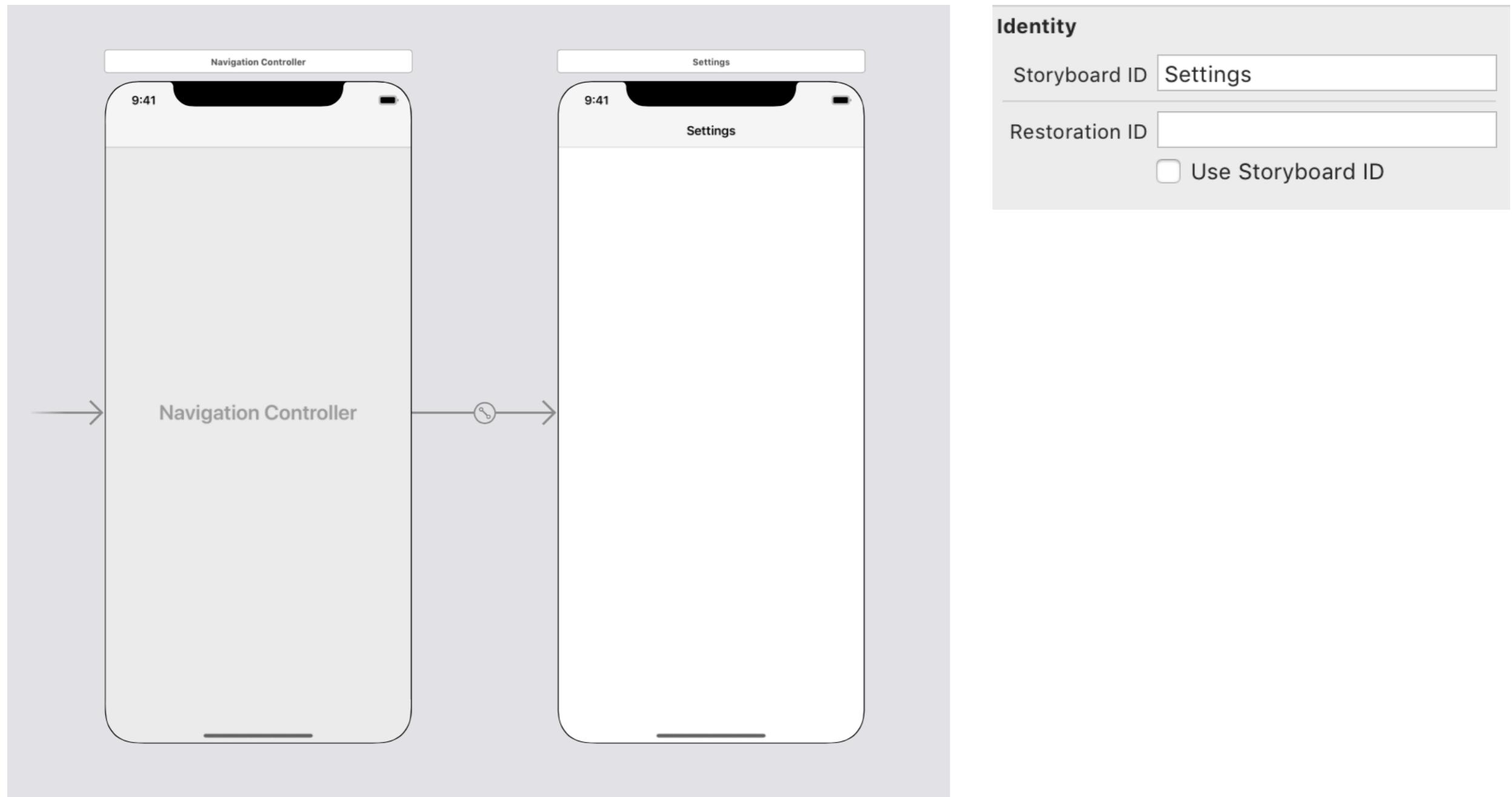
        // Assert
        XCTAssertEqual(vc.navigationItem.title, "Settings")
    }

}
```



# Make test pass !!

Create new ViewController in Main storyboard



# Second test case

## Verify text on Label

```
func test_label_text_is_Number_of_results_to_display() {  
    // Arrange  
    let storyboard = UIStoryboard(name: "Main", bundle: nil)  
  
    // Act  
    let vc = storyboard.instantiateViewController(withIdentifier:  
        "Settings") as! SettingsViewController  
  
    let _ = vc.view // Load view in unit test (called ViewDidLoad)  
  
    XCTAssertEqual(vc.label.text!, "Number of results to display")  
}
```



# Make test pass !!

Create new **SettingsViewController.swift**

```
import UIKit
class SettingsViewController: UIViewController {

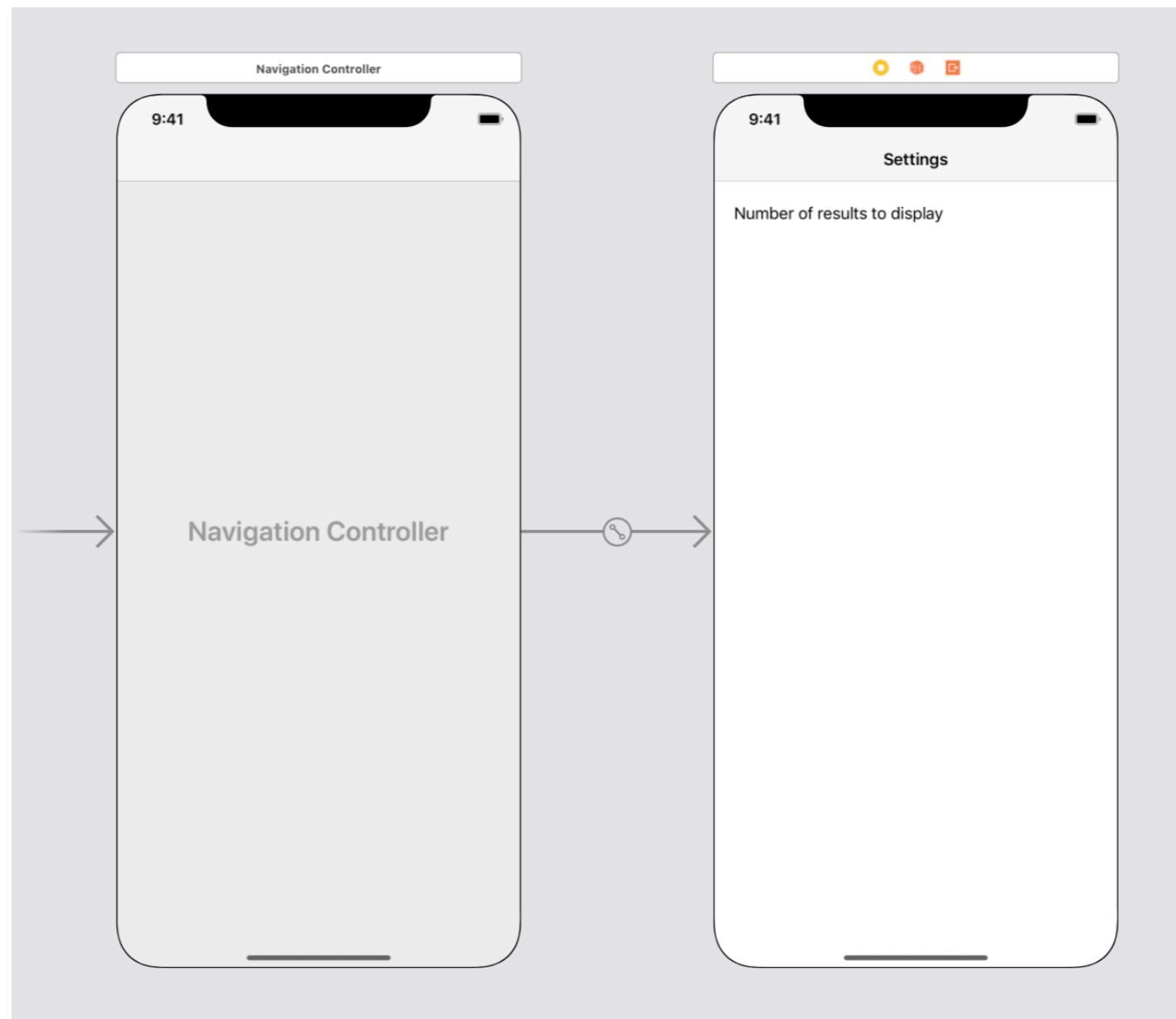
    @IBOutlet weak var label: UILabel!

}
```



# Make test pass !!

Add **UILabel** in storyboard



# Stop !!



# Remove duplication code !!



# Duplication code

```
func test_title_is_Settings() {
    // Arrange
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
    // Act
    let vc = storyboard.instantiateViewController(withIdentifier: "Settings")
    let _ = vc.view // Load view in unit test (called ViewDidLoad)

    // Assert
    XCTAssertEqual(vc.navigationItem.title, "Settings")
}

func test_label_text_is_Number_of_results_to_display() {
    // Arrange
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
    // Act
    let vc = storyboard.instantiateViewController(withIdentifier: "Settings") as!
        SettingsViewController
    let _ = vc.view // Load view in unit test (called ViewDidLoad)

    XCTAssertEqual(vc.label.text!, "Number of results to display")
}
```



# Remove Duplication code

```
func test_title_is_Settings() {
    let vc = settingsViewController()
    XCTAssertEqual(vc.navigationItem.title, "Settings")
}

func test_label_text_is_Number_of_results_to_display() {
    let vc = settingsViewController()
    XCTAssertEqual(vc.label.text!, "Number of results to display")
}

func settingsViewController() -> SettingsViewController {
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
    let vc = storyboard.instantiateViewController(withIdentifier: "Settings") as!
        SettingsViewController
    let _ = vc.view
    return vc
}
```



# Third test case

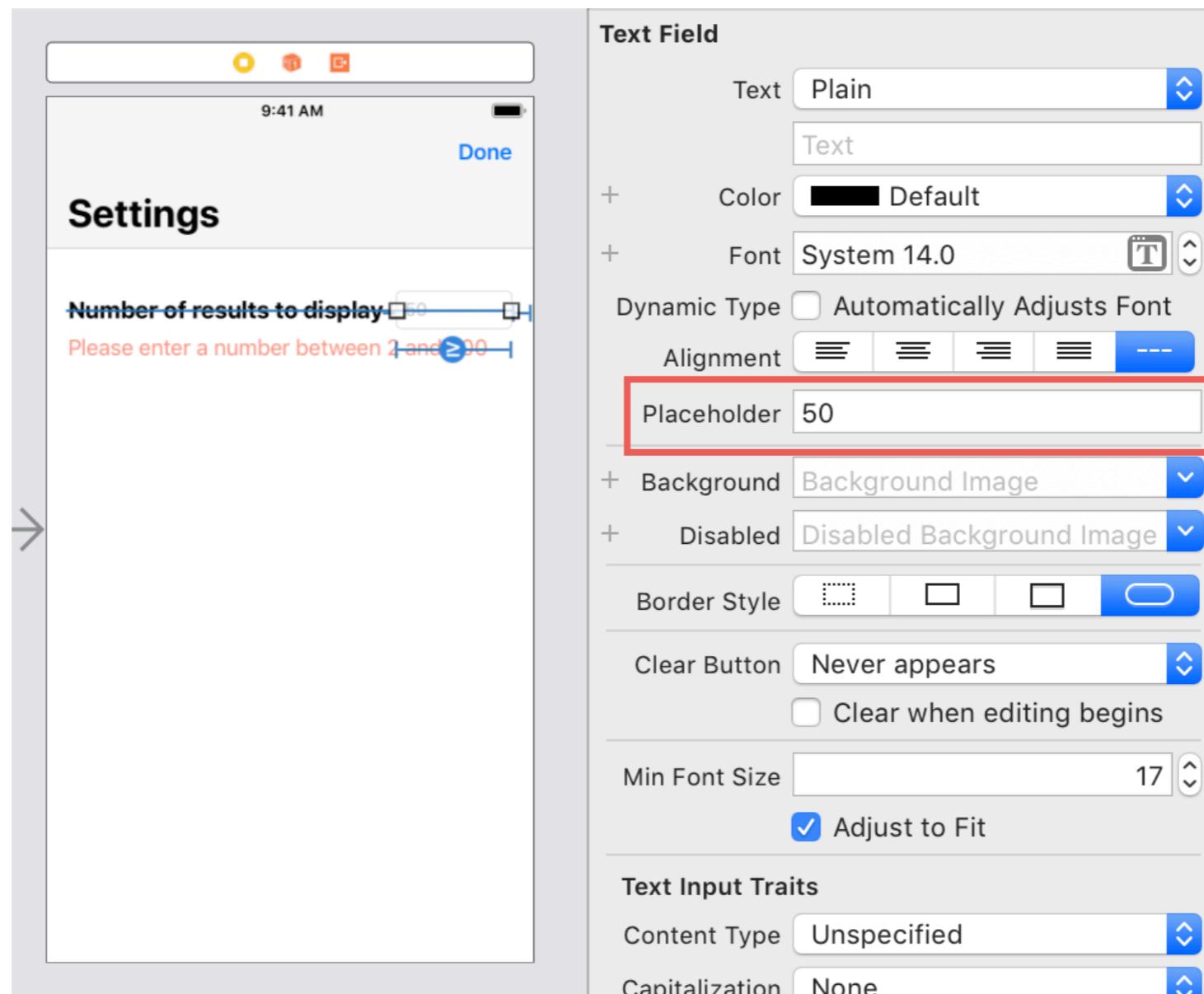
Verify placeholder text on text field = 50

```
func test_number_placeholder_is_50() {  
    let vc = settingsViewController()  
    XCTAssertEqual(vc.number.placeholder!, "50")  
}
```



# Make test pass !!

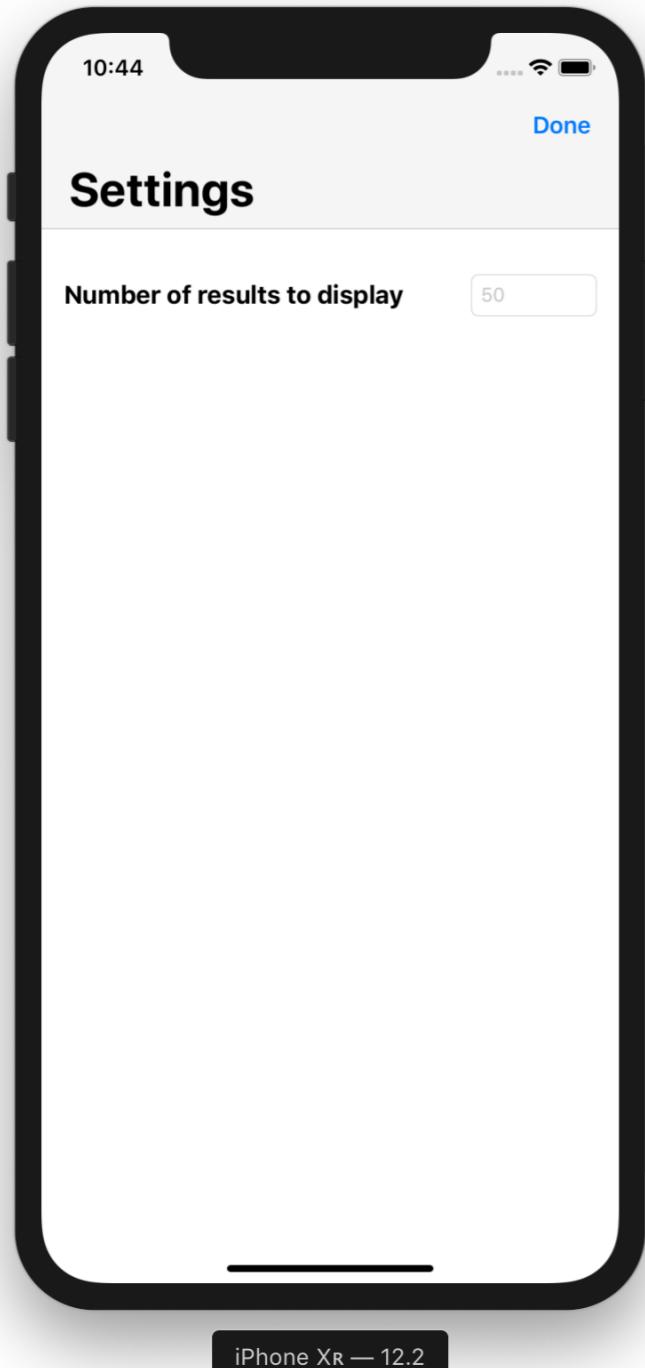
## Add UITextField in storyboard



# More tests



# What should we test ?



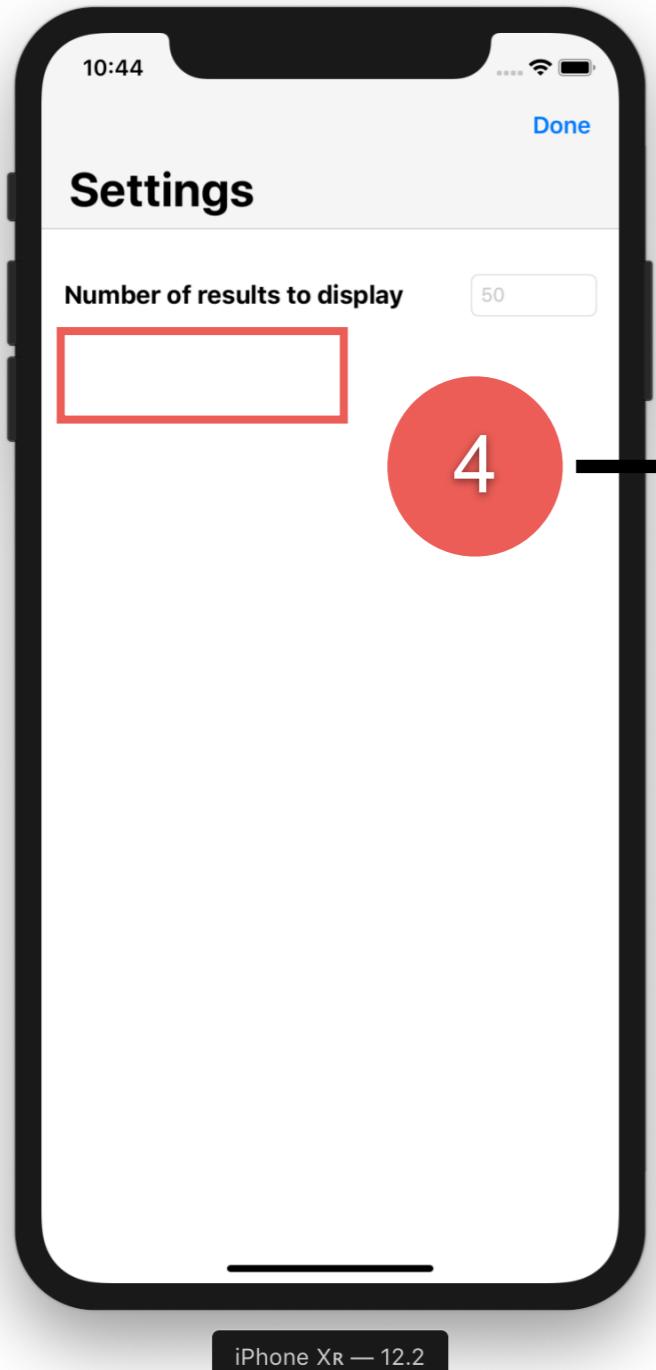
Enter number and save ?  
Range of number ?  
Validate input is number or not ?  
Show error message if validate failed ?



# Validate text entry



# Error message



Show error message  
**“Please enter a number between 2 and 200”**



# Forth test case

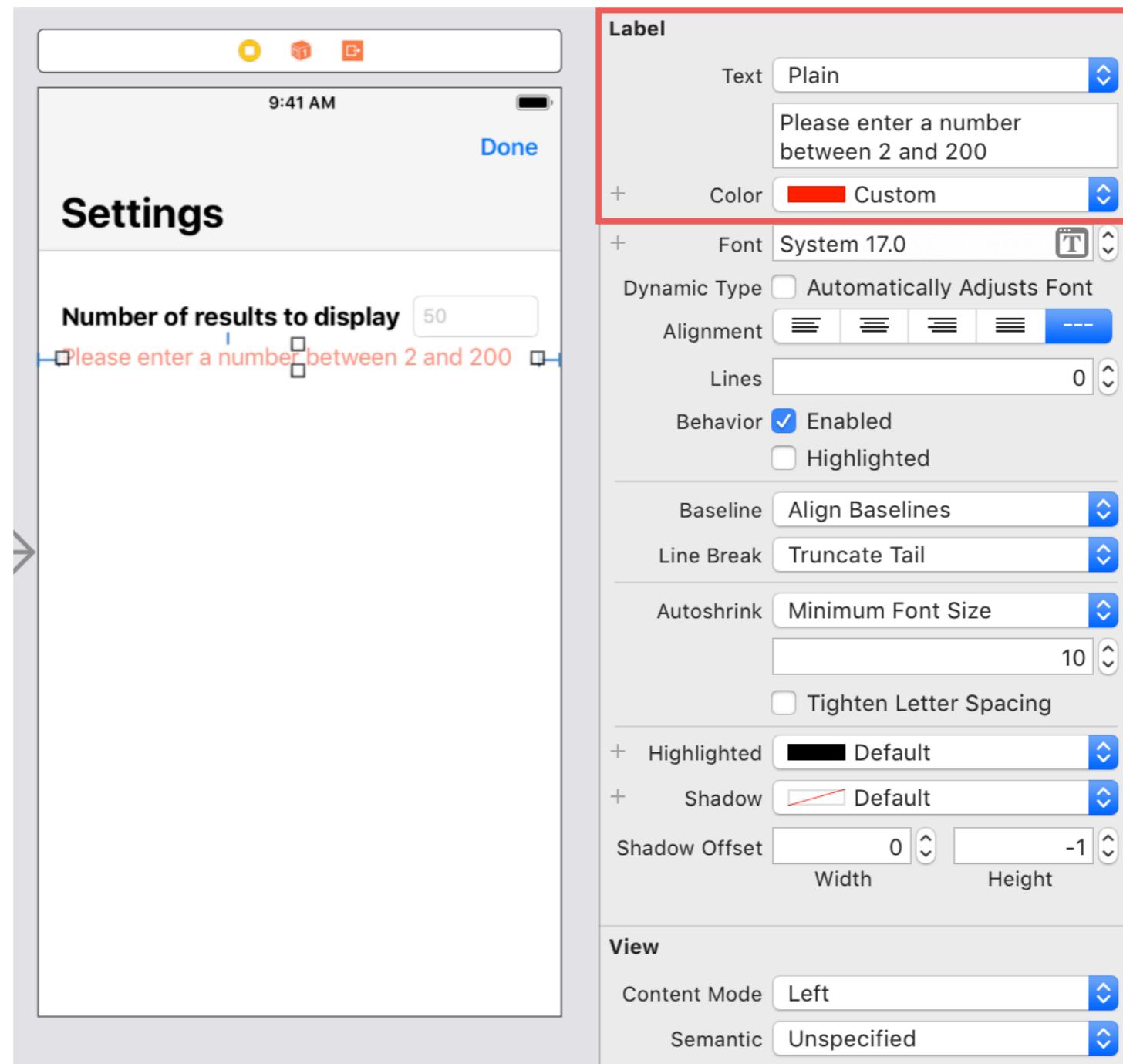
## Verify text entry with error message

```
func test_error_text_is_Please_enter_a_number_between_2_and_200() {  
    let vc = settingsViewController()  
    XCTAssertEqual(vc.error.text!,  
                  "Please enter a number between 2 and 200")  
}
```



# Make test pass !!

## Add UILabel in storyboard



# Waiting ...



# Stage of error message



# State of error message

Open settings page, error is hidden

When user enters text in number field that not a number, should **display error message**



# Error message is hidden

```
func test_error_is_hidden() {  
    let vc = settingsViewController()  
    XCTAssertTrue(vc.error.isHidden)  
}
```



# Not show error when input is correct

Error message not show

```
func test_number_allows_2() {  
    let vc = settingsViewController()  
    let _ = vc.textField(vc.number,  
        shouldChangeCharactersIn: NSRange(location: 0, length: 0),  
        replacementString: "2")  
    XCTAssertTrue(vc.error.isHidden)  
}
```



# Not show error when input is correct

Error message not show

```
func test_numberAllows200() {  
    let vc = settingsViewController()  
    let _ = vc.textField(vc.number,  
        shouldChangeCharactersIn: NSRange(location: 0, length: 0),  
        replacementString: "200")  
    XCTAssertTrue(vc.error.isHidden)  
}
```



# **Stop !!**



# Remove duplication code !!



# Duplication code

```
func test_numberAllows_2() {  
    let vc = settingsViewController()  
    let _ = vc.textField(vc.number,  
        shouldChangeCharactersIn: NSRange(location: 0, length: 0),  
        replacementString: "2")  
    XCTAssertTrue(vc.error.isHidden)  
}  
  
func test_numberAllows_200() {  
    let vc = settingsViewController()  
    let _ = vc.textField(vc.number,  
        shouldChangeCharactersIn: NSRange(location: 0, length: 0),  
        replacementString: "200")  
    XCTAssertTrue(vc.error.isHidden)  
}
```



# Remove Duplication code

Create function `enterText()` with extension

```
extension SettingsViewController {
    func enterText(_ text: String) {
        _ = textField(number, shouldChangeCharactersIn: NSRange(location:
            number.text!.count, length: 0), replacementString: text)
    }
}
```



# Remove Duplication code

Modify in test cases

```
func test_number_allows_2() {  
    let vc = settingsViewController()  
    vc.enterText("2")  
    XCTAssertTrue(vc.error.isHidden)  
}
```

```
func test_number_allows_200() {  
    let vc = settingsViewController()  
    vc.enterText("200")  
    XCTAssertTrue(vc.error.isHidden)  
}
```



# Out-of-range

“Please enter a number between 2 and 200”

```
func test_number_does_not_allow_1() {  
    let vc = settingsViewController()  
    vc.enterText("1")  
    XCTAssertFalse(vc.error.isHidden)  
}
```

```
func test_number_does_not_allow_201() {  
    let vc = settingsViewController()  
    vc.enterText("201")  
    XCTAssertFalse(vc.error.isHidden)  
}
```



# Error when user enter incorrect text

“Please enter a number between 2 and 200”

```
func test_number_does_not_allow_x2() {  
    let vc = settingsViewController()  
    vc.enterText("x2")  
    XCTAssertFalse(vc.error.isHidden)  
}
```

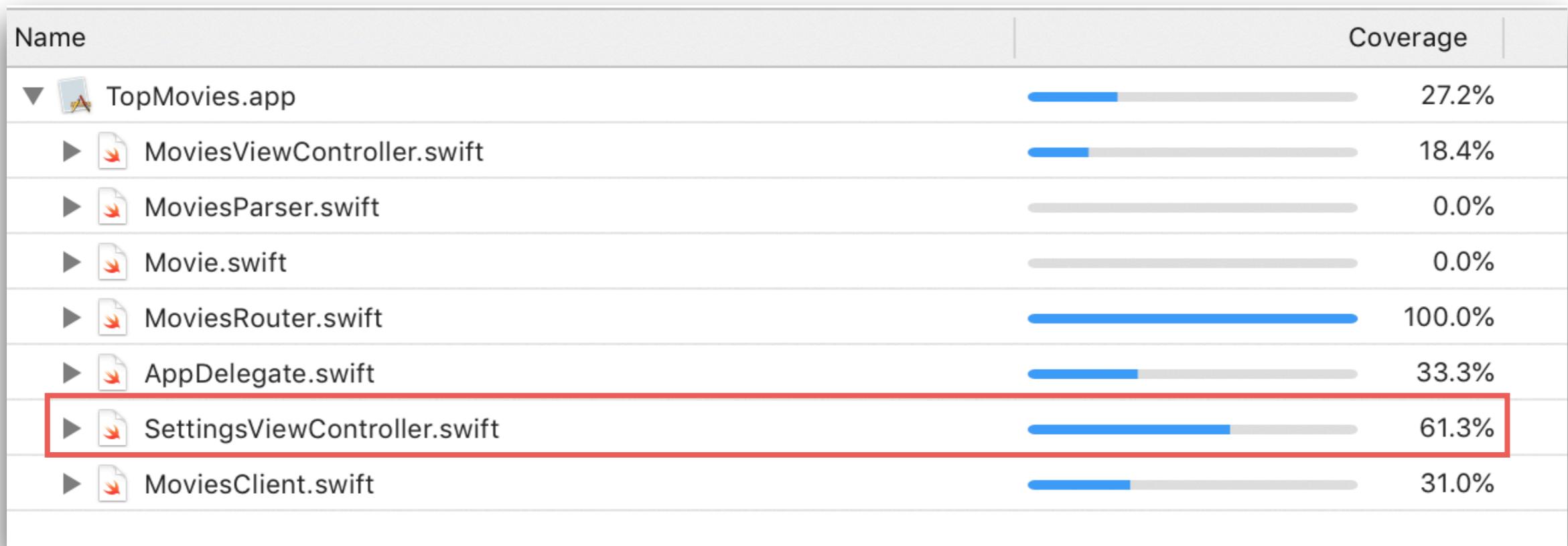


# Test results

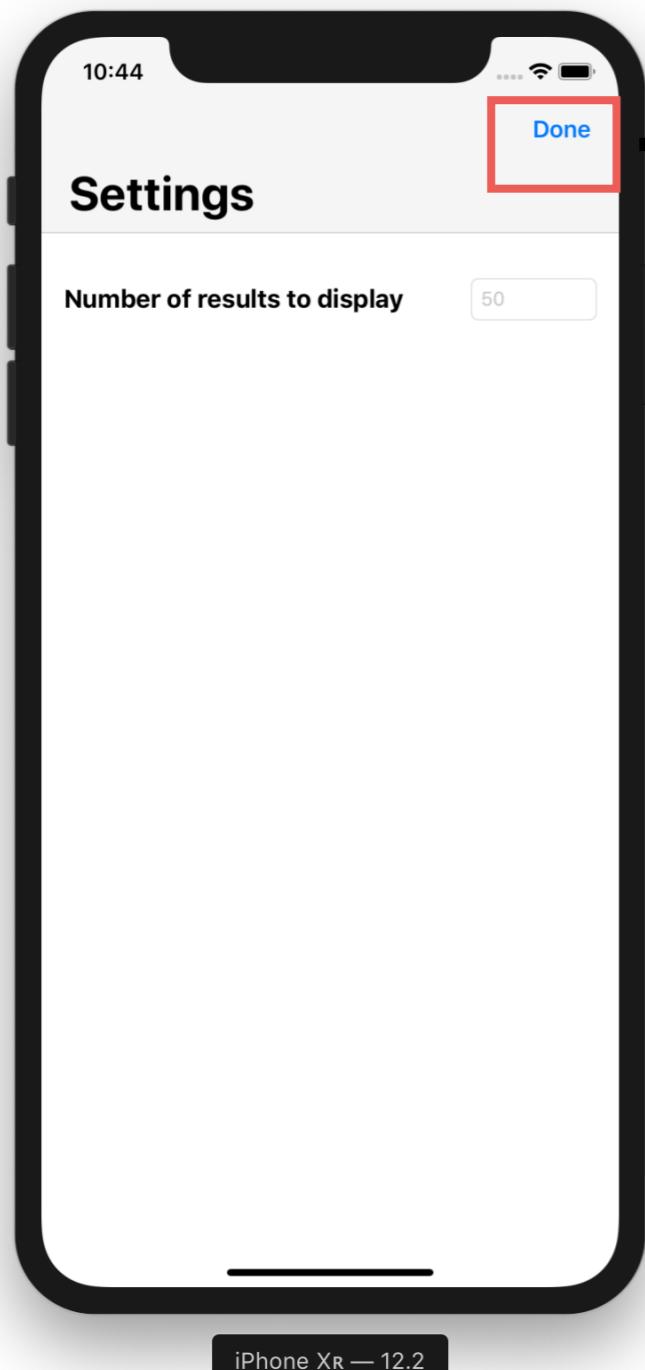
```
▼ └── TopMoviesTests 9 tests
    ▼ └── SettingsTests
        └── t test_title_is_Settings() ✓
        └── t test_label_text_i...ults_to_display() ✓
        └── t test_number_placeholder_is_50() ✓
        └── t test_error_text_i...en_2_and_200() ✓
        └── t test_numberAllows_2() ✓
        └── t test_numberAllows_200() ✓
        └── t test_number_does_not_allow_1() ✓
        └── t test_number_do...not_allow_201() ✓
        └── t test_number_does_not_allow_x2() ✓
```



# Code coverage



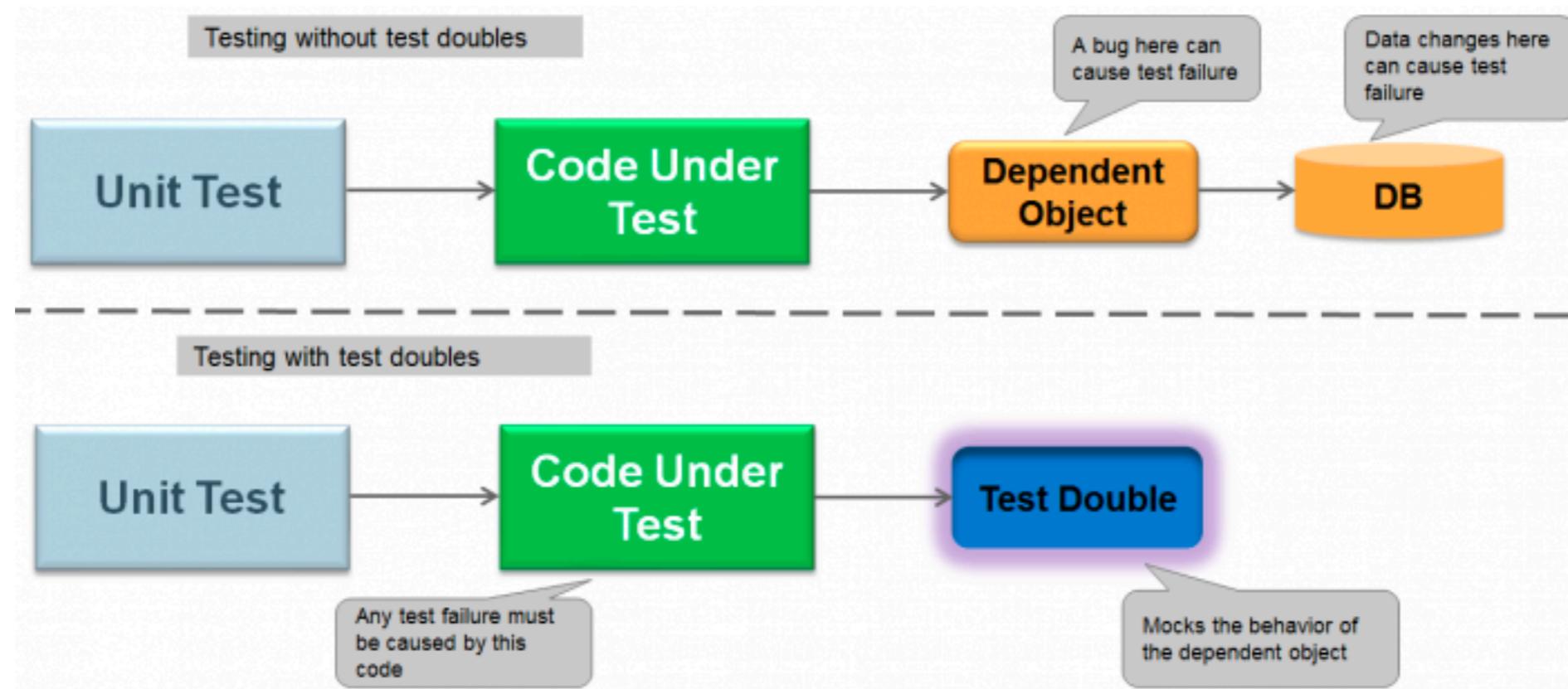
# Working with dependencies



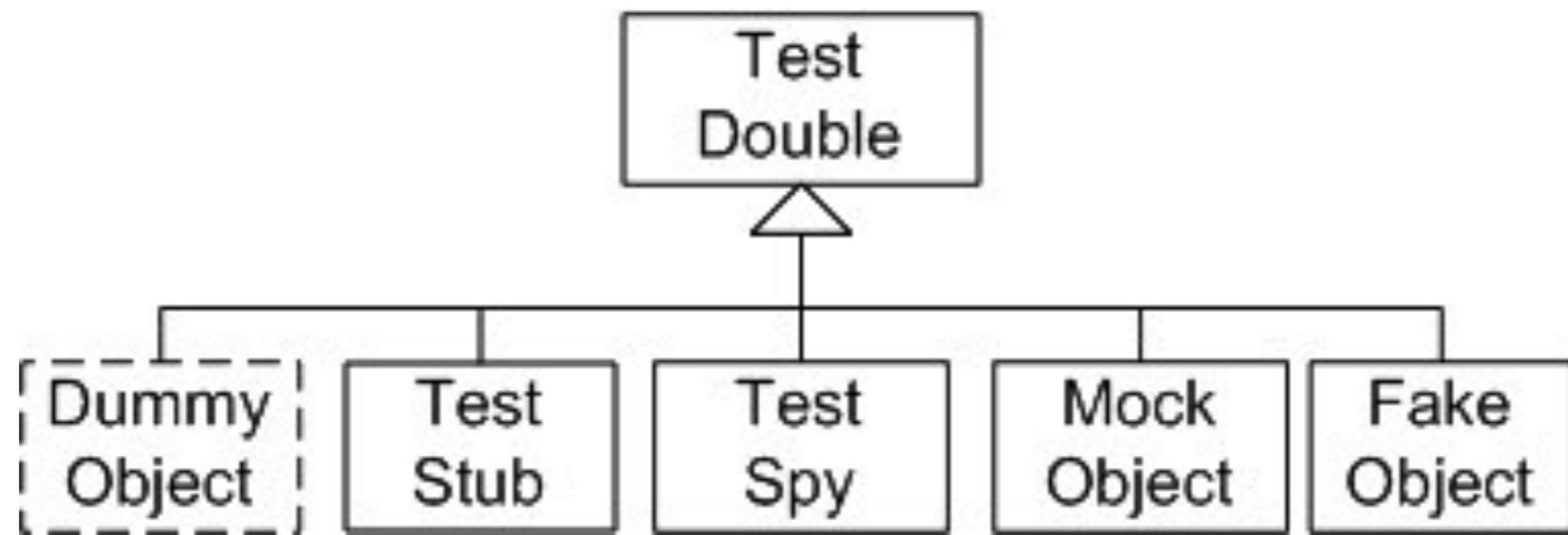
Save data in **UserDefault**s !!



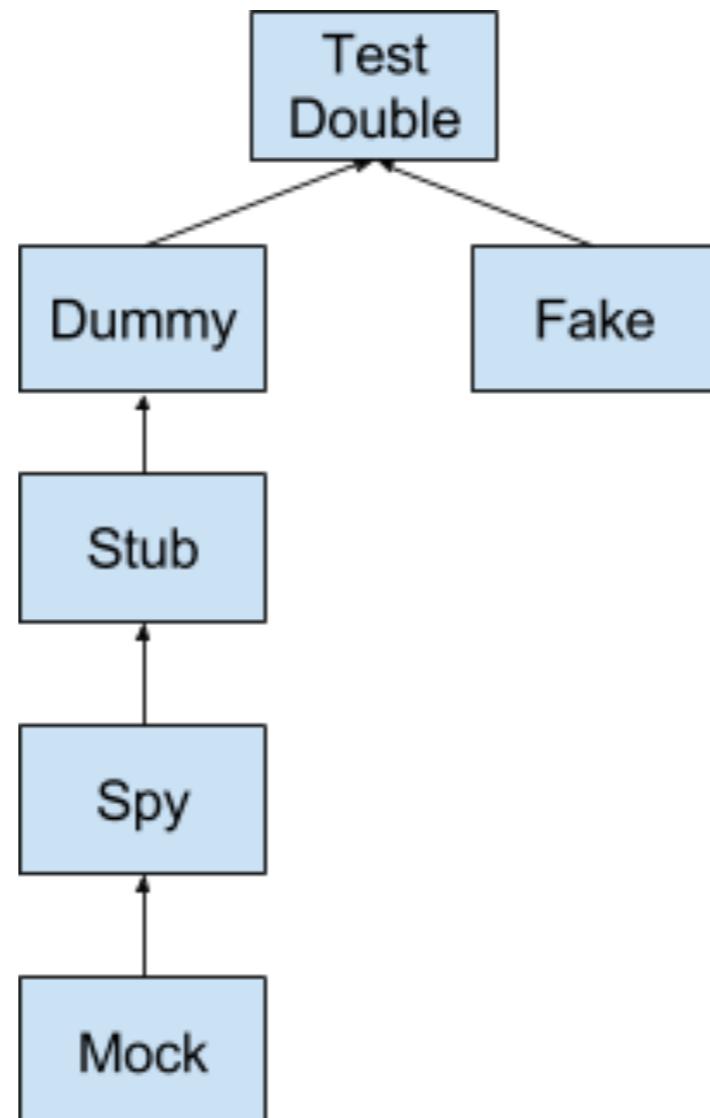
# Test Double



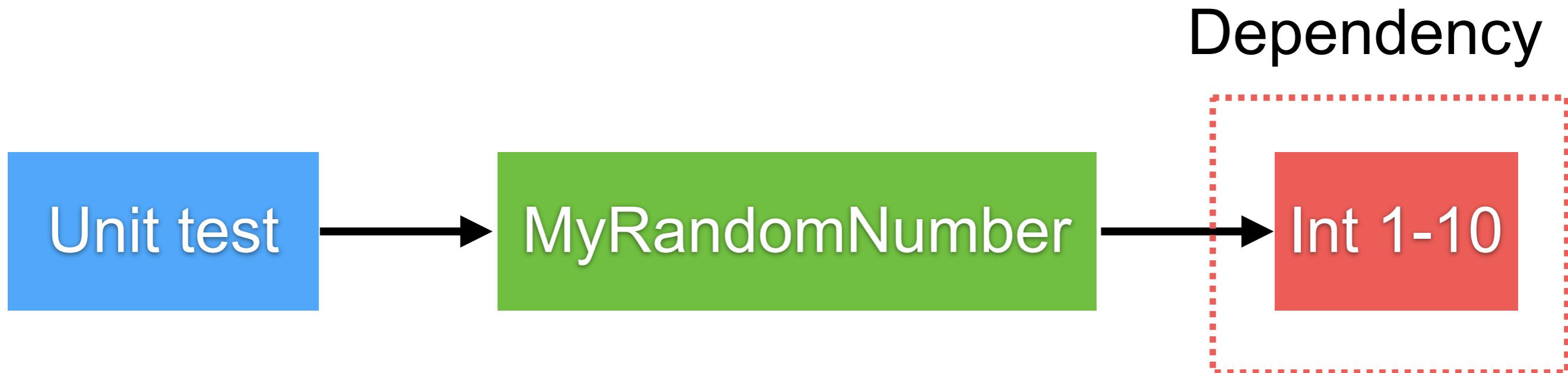
# Test Double



# Test Double

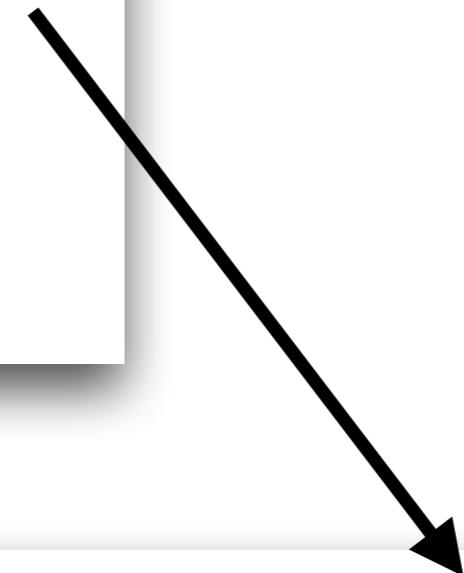


# Demo :: Random number



# How to control a dependency ?

```
class MyRandomNumberTest: XCTestCase {  
  
    func test_random_should_be_5() {  
        let my = MyRandomNumber()  
        let result = my.getNumber()  
        XCTAssertEqual(5, result)  
    }  
  
}
```

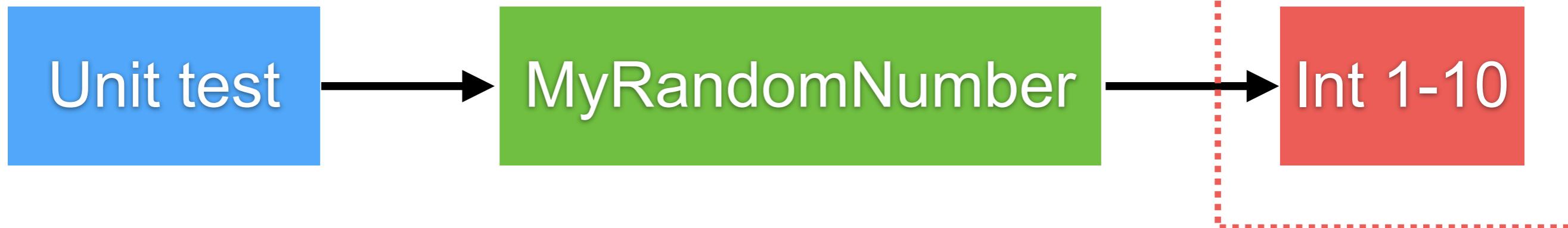


```
class MyRandomNumber {  
    func getNumber() -> Int {  
        return Int.random(in: 0...10)  
    }  
}
```

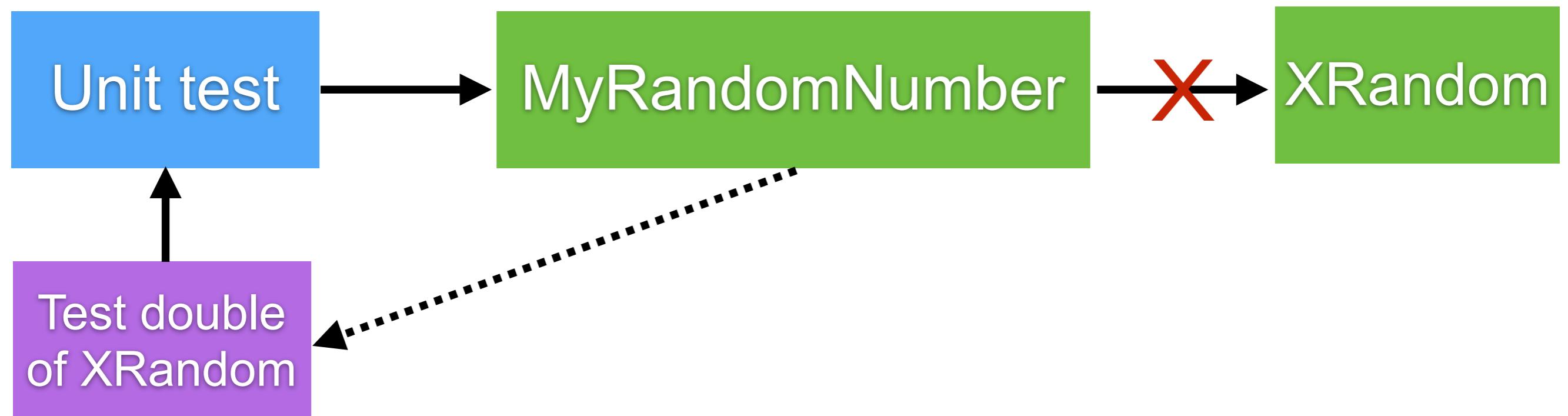


# Demo :: Random number

Dependency



# Demo :: Random number



# Control dependency

```
class MyRandomNumber {  
  
    var xRandom: XRandom!  
  
    func getNumber() -> Int {  
        return xRandom.get()  
    }  
}  
  
class XRandom {  
    func get() -> Int {  
        return Int.random(in: 0...10)  
    }  
}
```



# Testing again

```
class MyRandomNumberTest: XCTestCase {

    func test_random_should_be_5() {
        let my = MyRandomNumber()
        my.xRandom = StubXRandom5()
        let result = my.getNumber()
        XCTAssertEqual(5, result)
    }

}

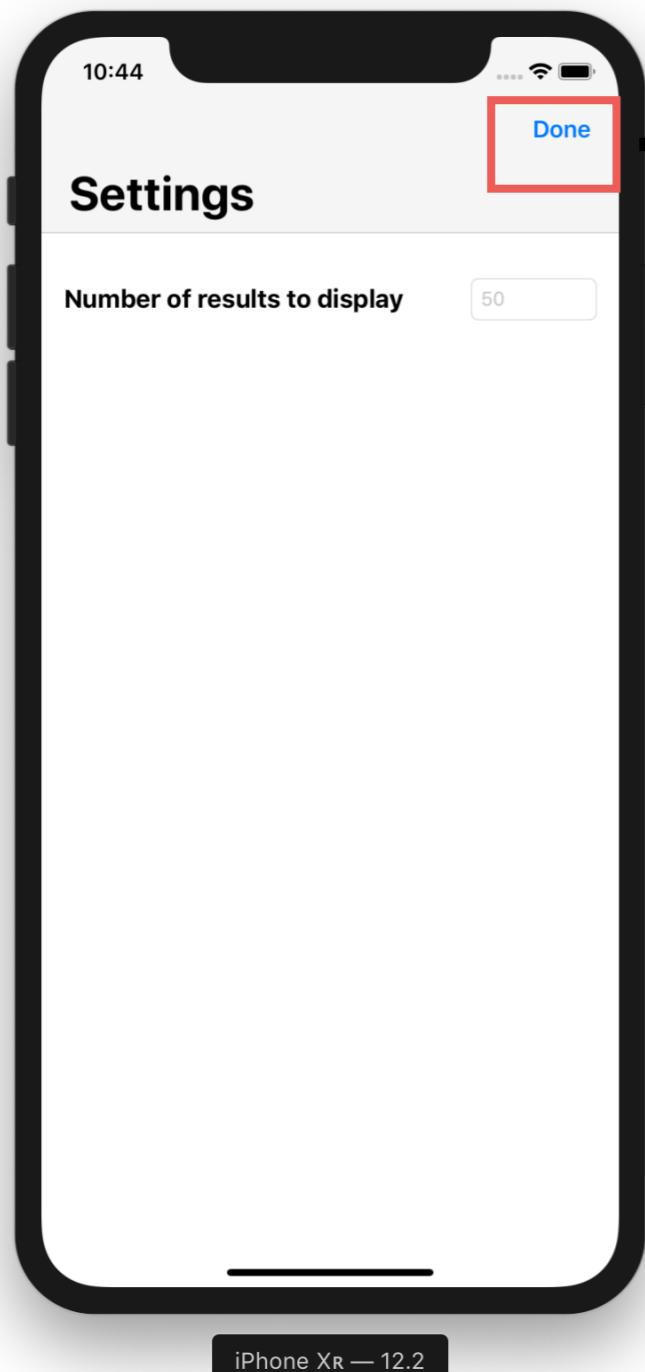
class StubXRandom5: XRandom {
    override func get() -> Int {
        return 5
    }
}
```



# Working with UserDefaults



# Working with dependencies



Save data in **UserDefault**s !!



# Fake UserDefaults

```
class FakeUserDefaults: UserDefaults {
    var store = [String: Any]()
    
    override func setValue(_ value: Any?, forKey key: String) {
        store[key] = value
    }
    
    override func value(forKey key: String) -> Any? {
        if let value = store[key] {
            return value
        }
        return nil
    }
}
```



# Save data into UserDefaults

```
class SettingsWithUserDefaultsTests: SettingsTests {

    func test_save_number_fromUserDefaults() {
        // Arrange
        let vc = settingsViewController()
        let defaults = FakeUserDefaults()
        vc.userDefaults = defaults

        // Act = Save
        vc.enterText("2")
        XCTAssertEqual(2, defaults.value(forKey:
            UserDefaultKeys.numberOfResults.rawValue) as! Int)
    }
}
```



# Remove data in UserDefaults

```
func test_remove_number_fromUserDefaults() {
    // Arrange
    let vc = settingsViewController()
    let defaults = FakeUserDefaults()
    vc.userDefaults = defaults

    // Act = Remove
    vc.number.text = ""
    vc.enterText("")
    XCTAssertNil(defaults.value(forKey: UserDefaultsKeys.numberOfResults.rawValue))
}
```



# Save and Remove

```
func test_save_and_remove_number_fromUserDefaults() {
    // Arrange
    let vc = settingsViewController()
    let defaults = FakeUserDefaults()
    vc.userDefaults = defaults

    // Act = Save
    vc.enterText("2")
    XCTAssertEqual(2, defaults.value(forKey:
        UserDefaultsKeys.numberOfResults.rawValue) as! Int)

    // Act = Remove
    vc.number.text = ""
    vc.enterText("")
    XCTAssertNil(defaults.value(forKey: UserDefaultsKeys.numberOfResults.rawValue))
}
```



# Code coverage

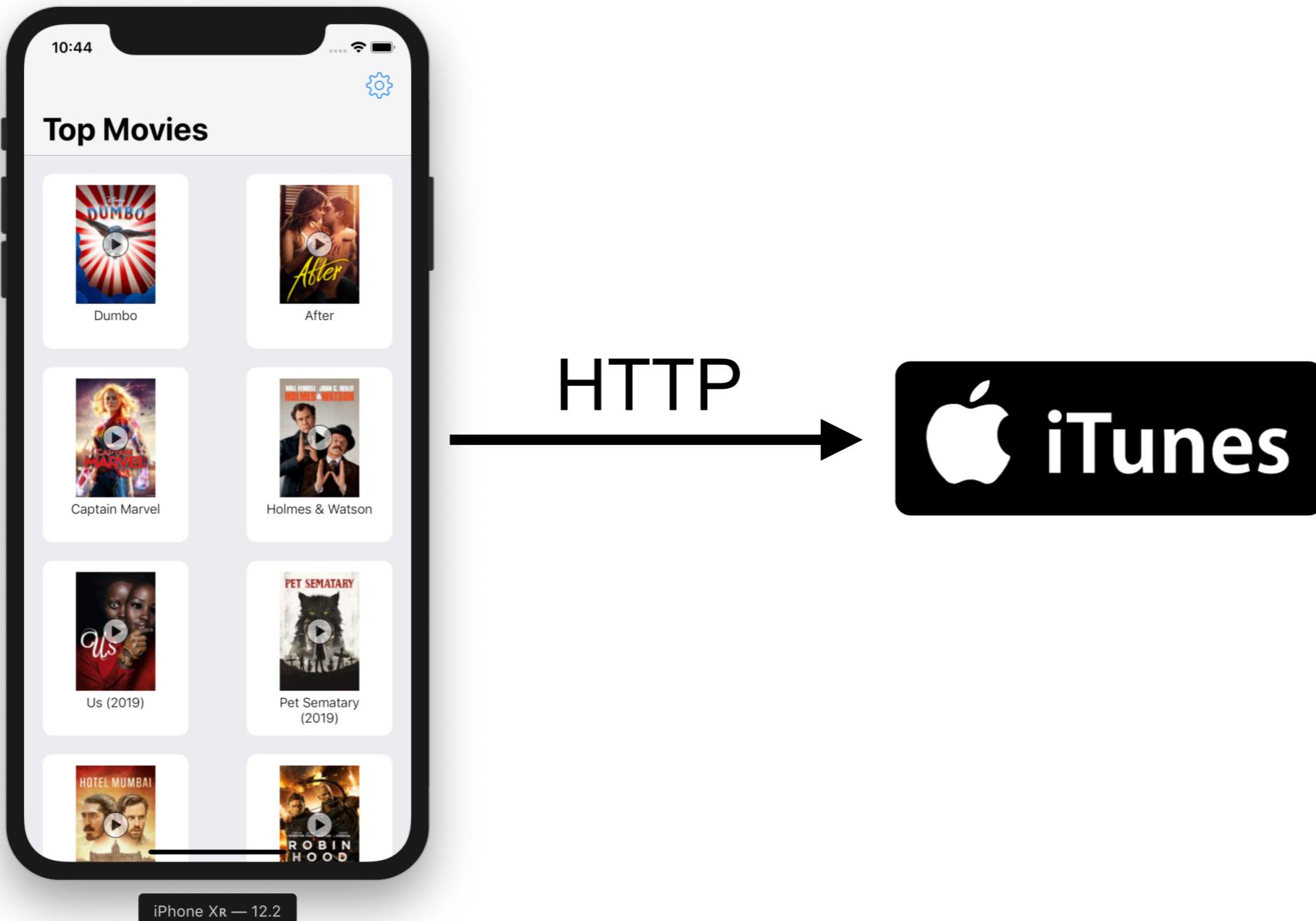
Name	Coverage
▼ TopMovies.app	29.0%
▶ MoviesViewController.swift	18.4%
▶ MyRandomNumber.swift	50.0%
▶ MoviesParser.swift	0.0%
▶ Movie.swift	0.0%
▶ MoviesRouter.swift	100.0%
▶ AppDelegate.swift	33.3%
▶ SettingsViewController.swift	71.0%
▶ MoviesClient.swift	31.0%



# Top movies page



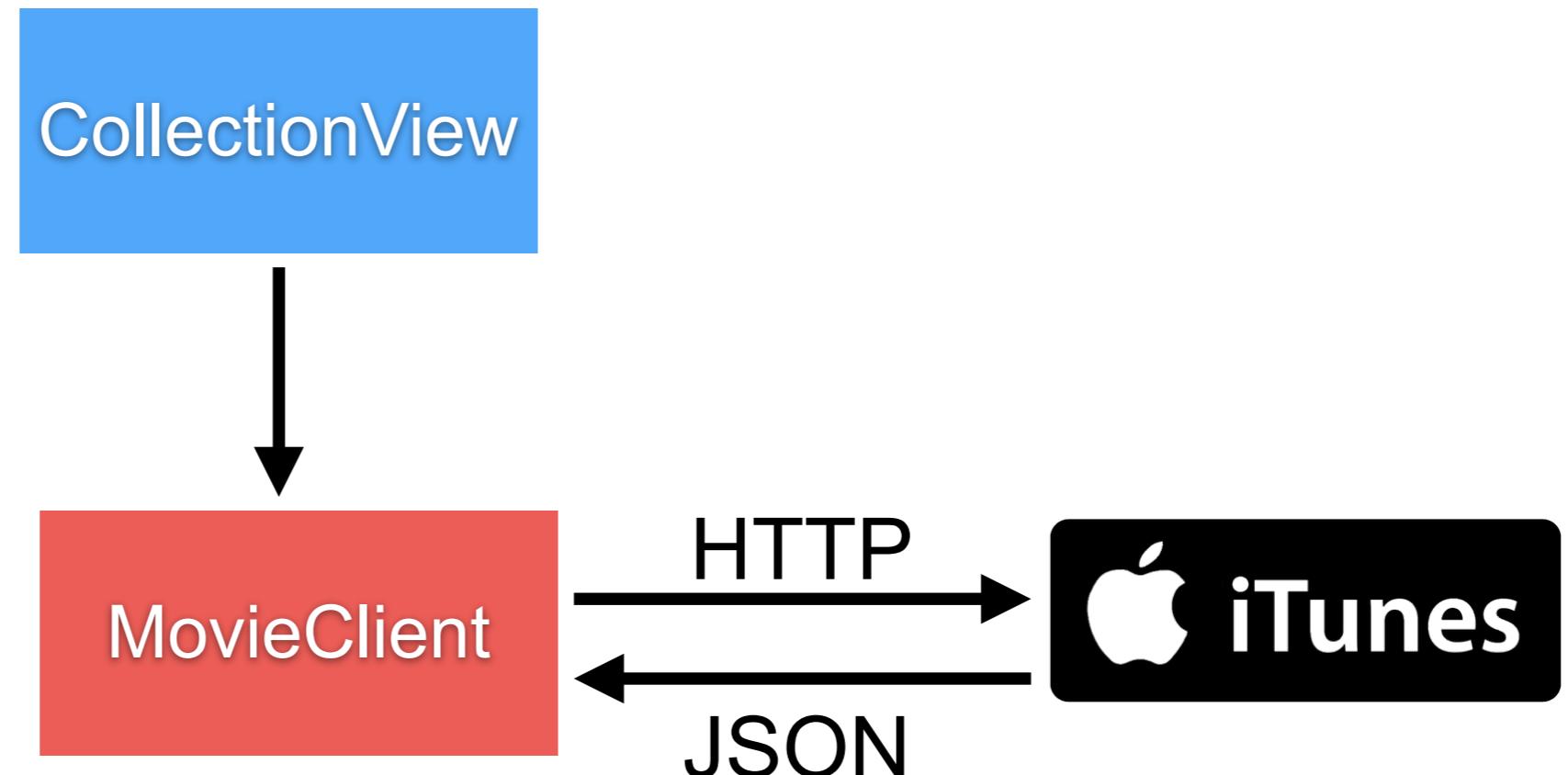
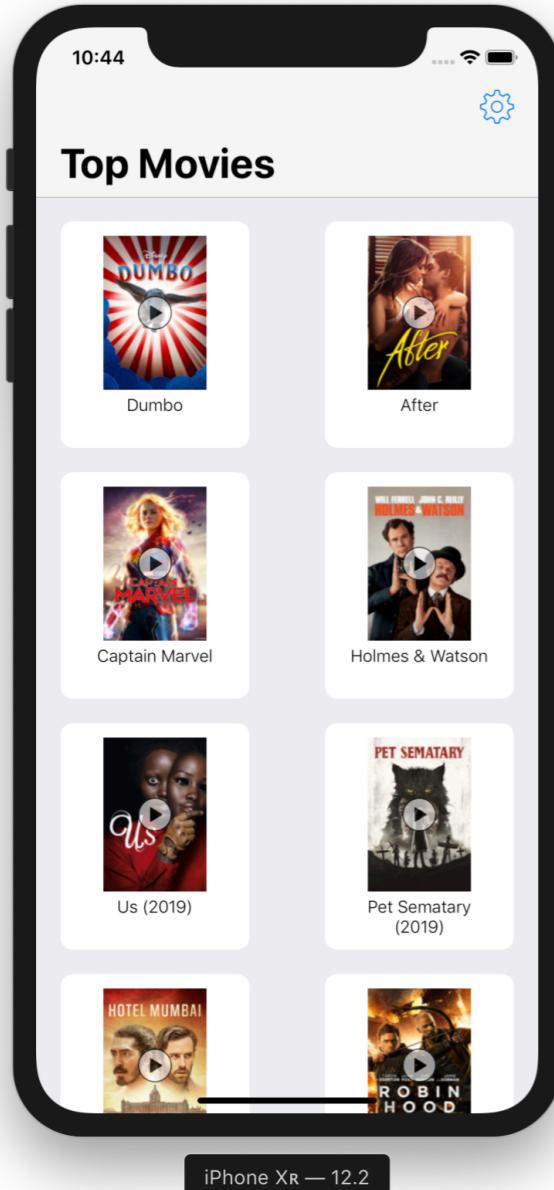
# Top movies page



<https://itunes.apple.com/us/rss/topmovies/limit=50/json>



# Top movies page



# Test cases ?



# Test cases ?

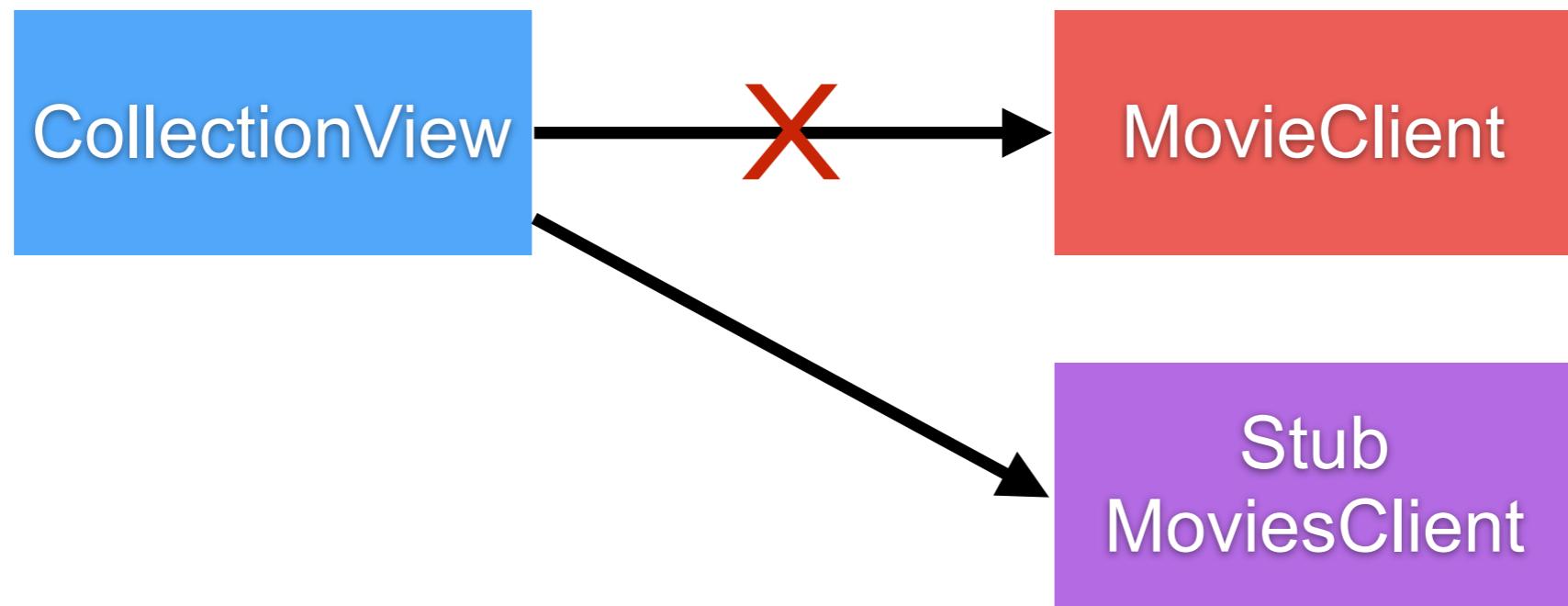
Title of page = **Top Movies**

Test the collection view

Working with APIs



# Manage dependency



# StubMoviesClient.swift

```
class StubMoviesClient: MoviesClient {
    var movies: [Movie]
    init(movies: [Movie]) {
        self.movies = movies
    }

    override func fetchMovies(completion: (([Movie]) -> Void)?) {
        completion!(movies)
    }
}
```



# Title of page = Top Movies

```
class MoviesViewControllerTests: XCTestCase {

    func moviesViewController(
        client: MoviesClient = StubMoviesClient(movies: []))
    -> MoviesViewController {
        let storyboard = UIStoryboard(name: "Main", bundle: nil)
        let vc = storyboard.instantiateViewController(withIdentifier: "Movies")
            as! MoviesViewController
        vc.moviesClient = client
        let _ = vc.view
        return vc
    }

    func test_title_is_Top_Movies() {
        let vc = moviesViewController()
        XCTAssertEqual(vc.navigationItem.title!, "Top Movies")
    }
}
```



# Test the collection view

No movies

2 movies

Collection view cell title name

Refresh when back to this page



# No movie to show

```
func test_collection_view_has_zero_items_when_there_are_no_movies() {  
    let vc = moviesViewController()  
    let numberOfItems = vc.collectionView(vc.collectionView,  
        numberOfRowsInSection: 0)  
    XCTAssertEqual(numberOfItems, 0)  
}
```



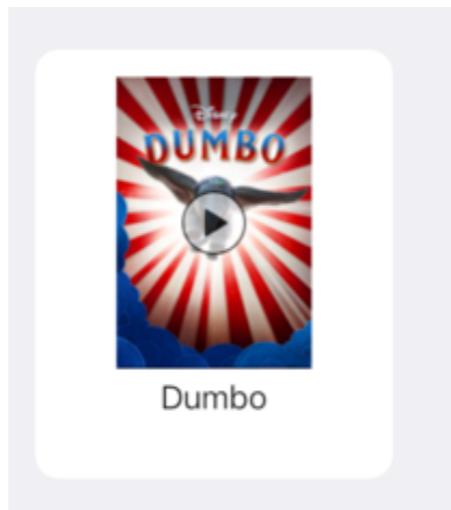
# 2 movies

```
func test_collection_view_has_two_items_when_there_are_two_movies() {  
    let movies = [createMovie(), createMovie()]  
    let vc = moviesViewController(client: StubMoviesClient(movies: movies))  
    let numberOfRowsInSection = vc.collectionView(vc.collectionView,  
        numberOfRowsInSectionInSection: 0)  
    XCTAssertEqual(numberOfItems, 2)  
}  
  
func createMovie(title: String = "TEST") -> Movie {  
    return Movie(title: title)  
}
```



# Collection view cell title name

```
func test_first_cell_title_is_Jumanji_when_movie_is_Jumanji() {  
    let movie = createMovie(title: "Jumanji: Welcome to the Jungle")  
    let movies = [movie]  
    let vc = moviesViewController(client: StubMoviesClient(movies: movies))  
  
    let cell = vc.collectionView(vc.collectionView, cellForItemAt:  
        IndexPath(row: 0, section: 0)) as! MovieCell  
  
    XCTAssertEqual(movie.title, cell.title.text!)  
}
```



# Refresh when back to this page

```
func test_movies_are_set_when_client_returns_movies() {  
    let movies = [createMovie(), createMovie(), createMovie()]  
    let vc = moviesViewController(client: StubMoviesClient(movies: movies))  
    vc.refresh()  
    XCTAssertEqual(vc.movies.count, 3)  
}
```



# Code coverage

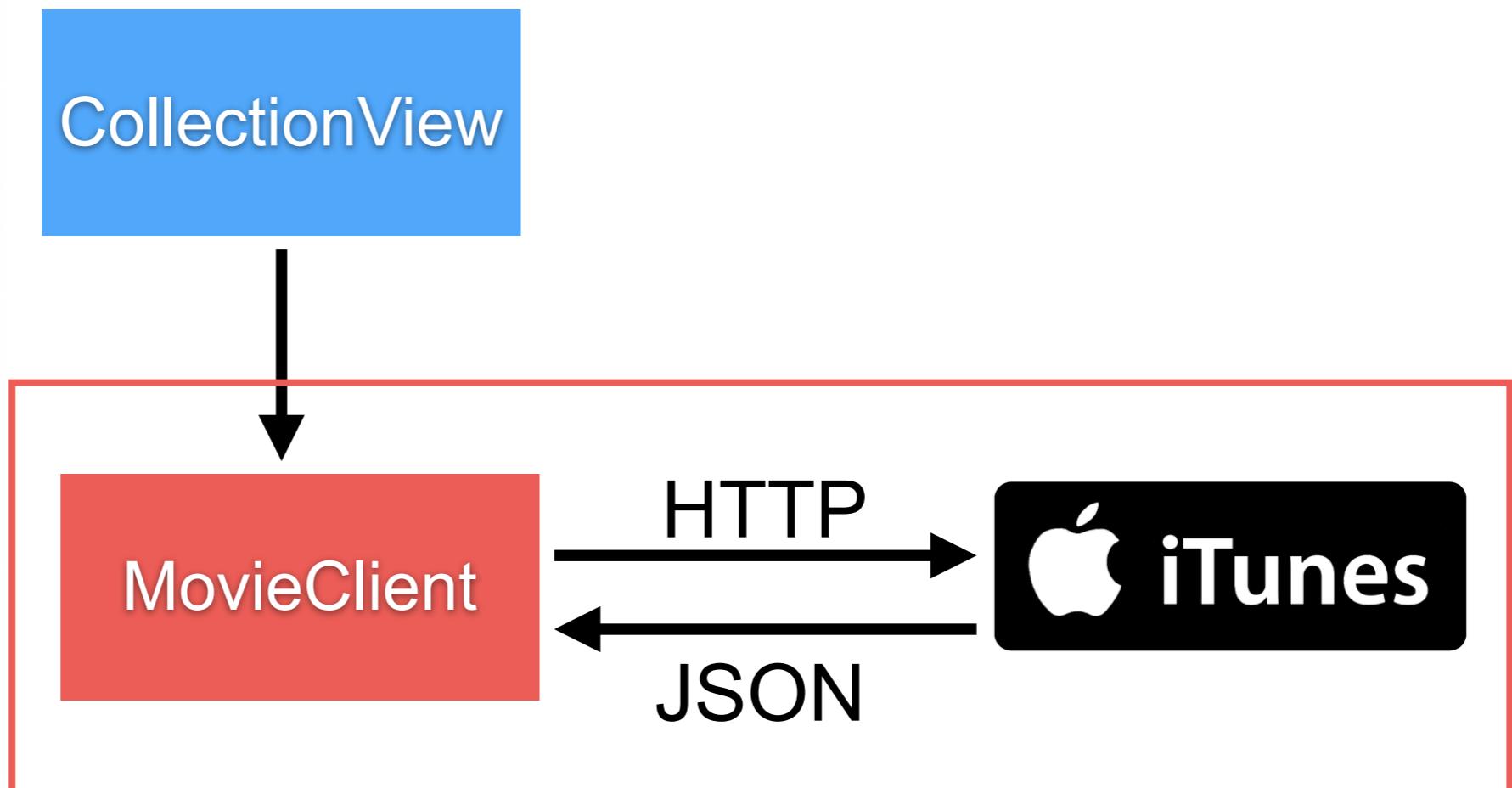
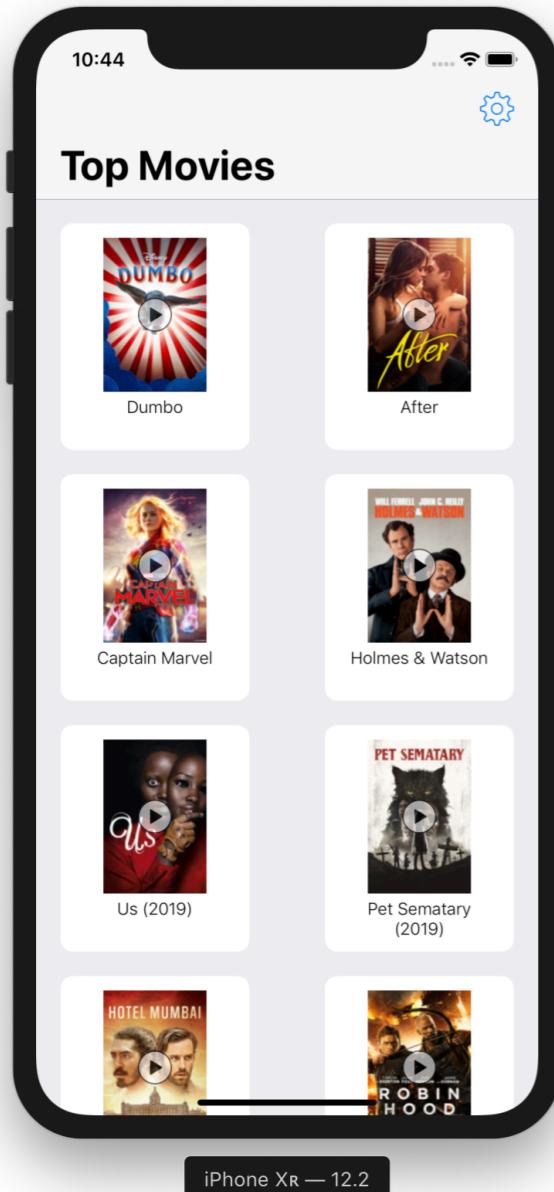
Name	Coverage
▼ TopMovies.app	49.4%
► MoviesViewController.swift	71.1%
► MyRandomNumber.swift	50.0%
► MoviesParser.swift	0.0%
► Movie.swift	14.7%
► MoviesRouter.swift	100.0%
► AppDelegate.swift	33.3%
► SettingsViewController.swift	71.0%
► MoviesClient.swift	39.7%



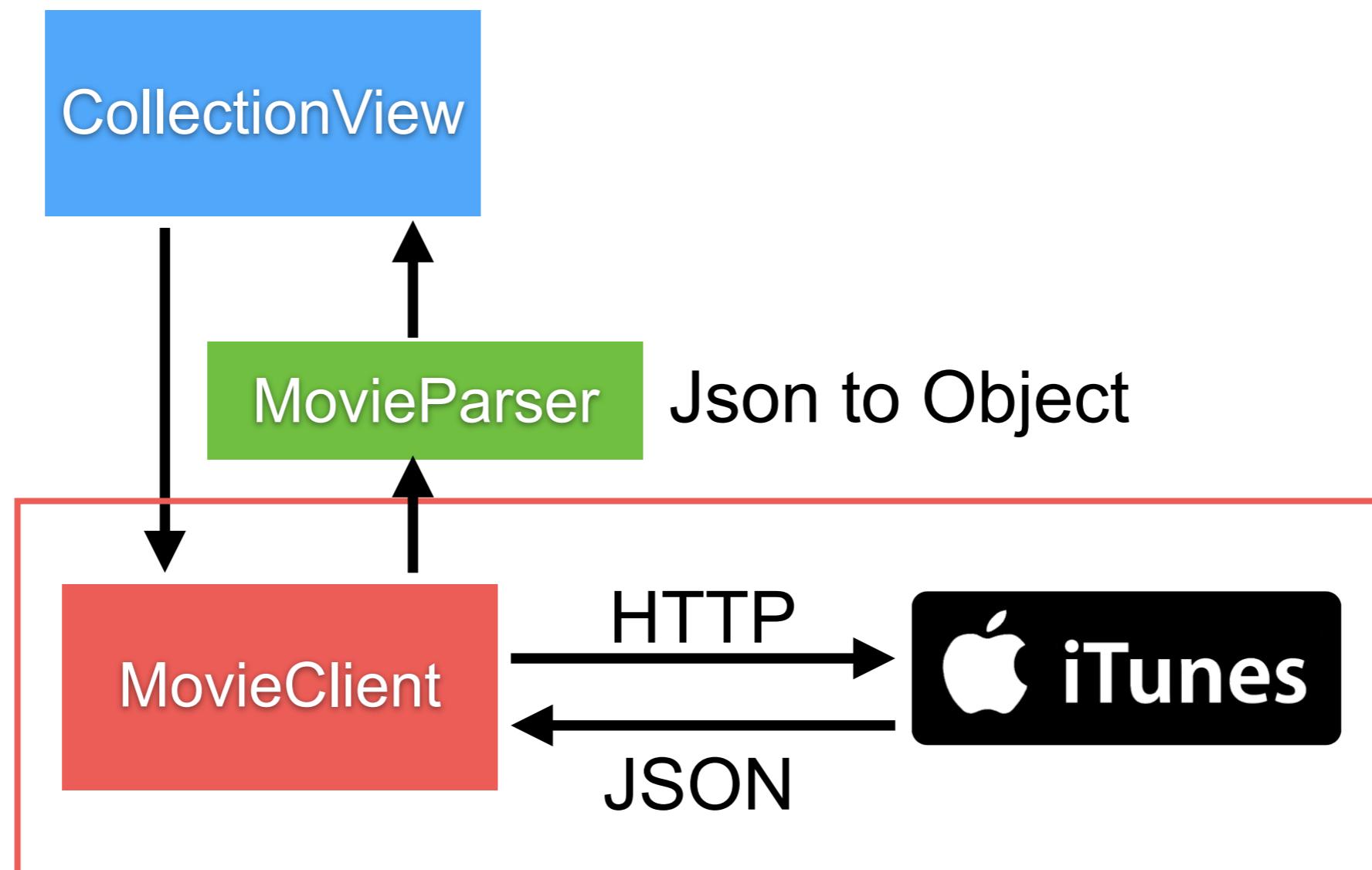
# Working with APIs



# Working with APIs



# Working with APIs



# Try with Stub

```
class MoviesClientTests: XCTestCase {

    func test_fetch_calls_completion() {
        let client = MoviesClient()
        client.router = StubMoviesRouter(limit: 6)
        let completionExpectation = expectation(description: "Fetch movies should call completion")
        client.fetchMovies { movies in
            completionExpectation.fulfill()
        }
        waitForExpectations(timeout: 5)
    }

}
```

```
class StubMoviesRouter: MoviesRouter {
    override func urlRequest() -> URLRequest {
        let url = URL(string: "http://10.105.147.41:8882/us/rss/topmovies/limit=6/json")!
        return URLRequest(url: url)
    }
}
```



# Waiting for result (Asynchronous)

```
class MoviesClientTests: XCTestCase {

    func test_fetch_calls_completion() {
        let client = MoviesClient()
        client.router = StubMoviesRouter(limit: 6)
        let completionExpectation = expectation(description: "Fetch movies should call completion")
        client.fetchMovies { movies in
            completionExpectation.fulfill()
        }
        waitForExpectations(timeout: 5)
    }

}

class StubMoviesRouter: MoviesRouter {
    override func urlRequest() -> URLRequest {
        let url = URL(string: "http://10.105.147.41:8882/us/rss/topmovies/limit=6/json")!
        return URLRequest(url: url)
    }
}
```



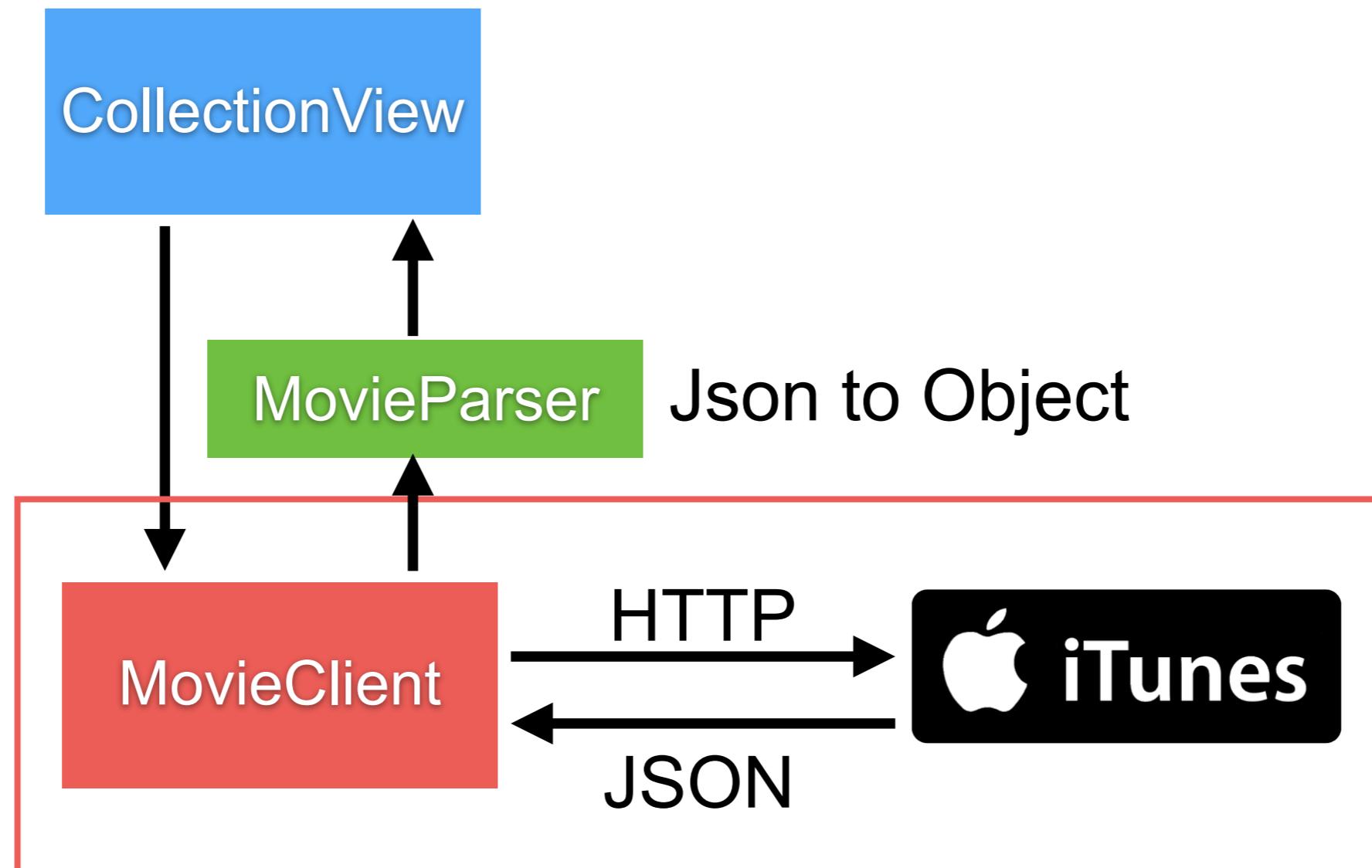
# Good or Bad ?



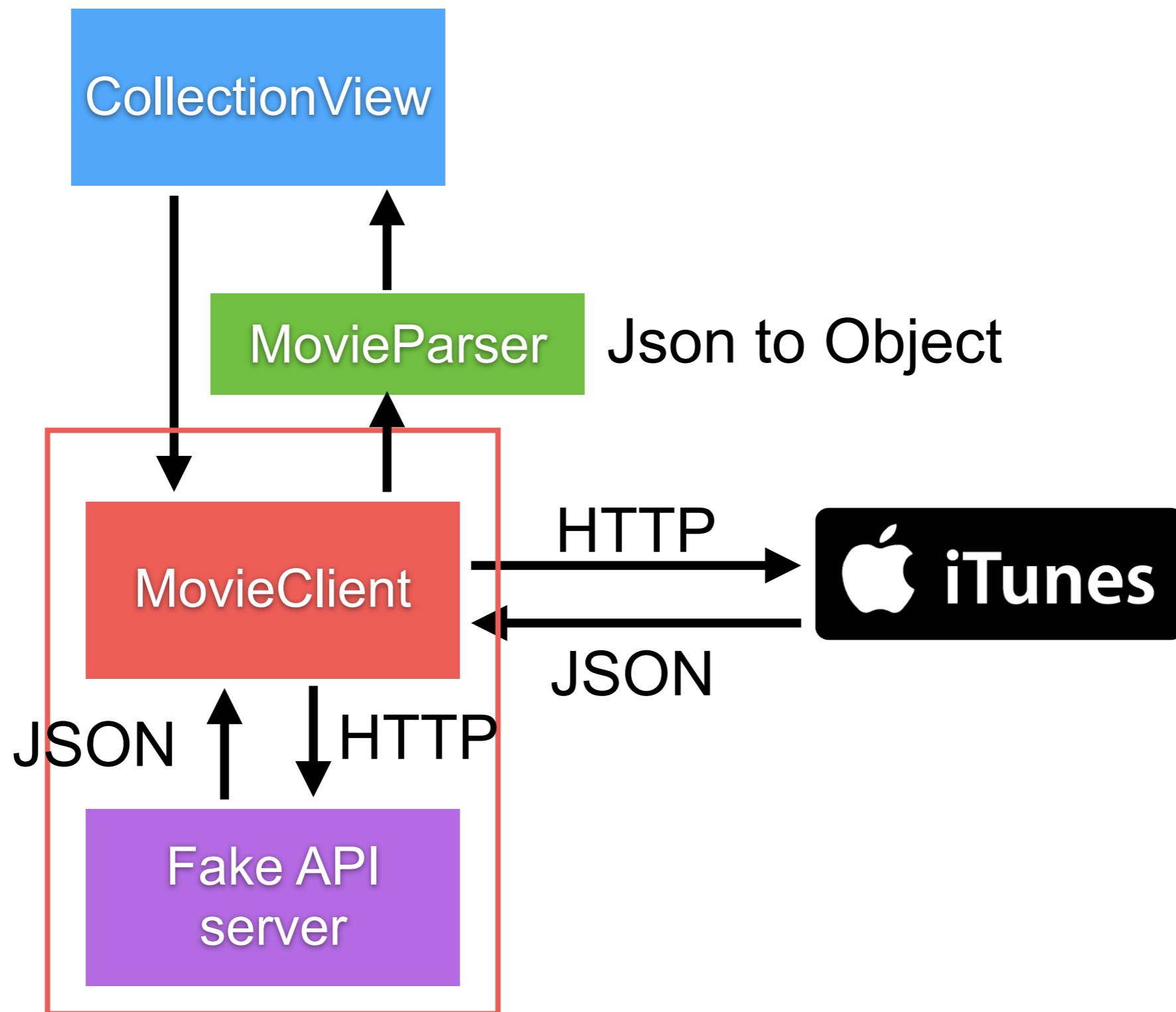
# UI Testing



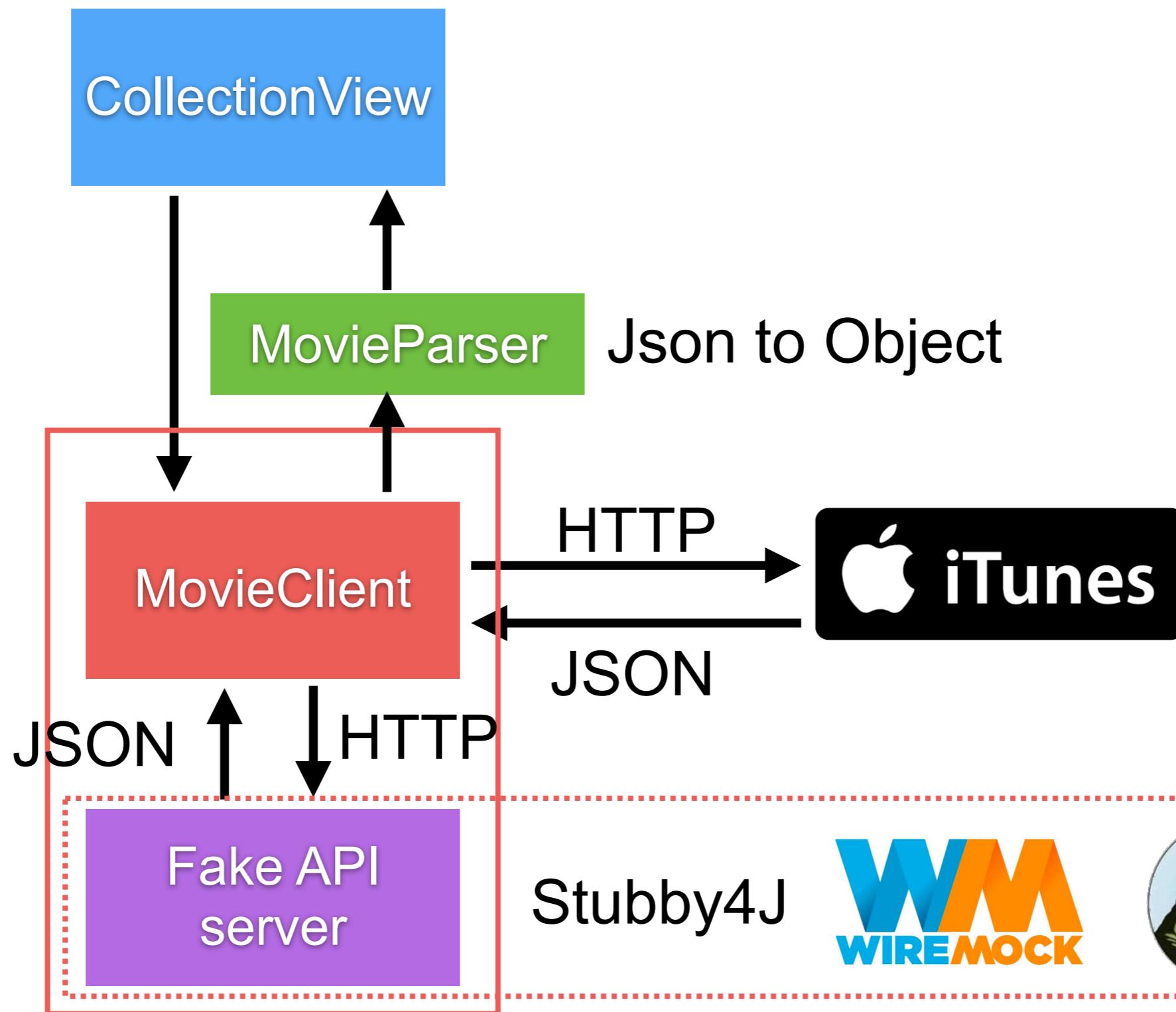
# Working with APIs



# Working with APIs



# Working with APIs



# **Problem**

# **How to change URL of APIs from tests ?**



# For iOS app

Active compilation + #if DEBUG

**Use Launch Environment from tests**

Custom page/ app for tests



# Start your Mock API Server



# Using Launch Environments from Tests

Easy to use but need more code !!

```
class TopMoviesUITests: XCTestCase {

    override func setUp() {
        continueAfterFailure = false
        let app = XCUIApplication()
        app.launchEnvironment["url"] =
            "http://192.168.1.33:8882/us/rss/topmovies/limit=6/json"
        app.launch()
    }

    func test_show_6_items() {
        let app = XCUIApplication()
        let collectionViewsQuery = app.collectionViews
        XCTAssertEqual(6, collectionViewsQuery.cells.count)
    }
}
```



# Write more production code

To check data from tests

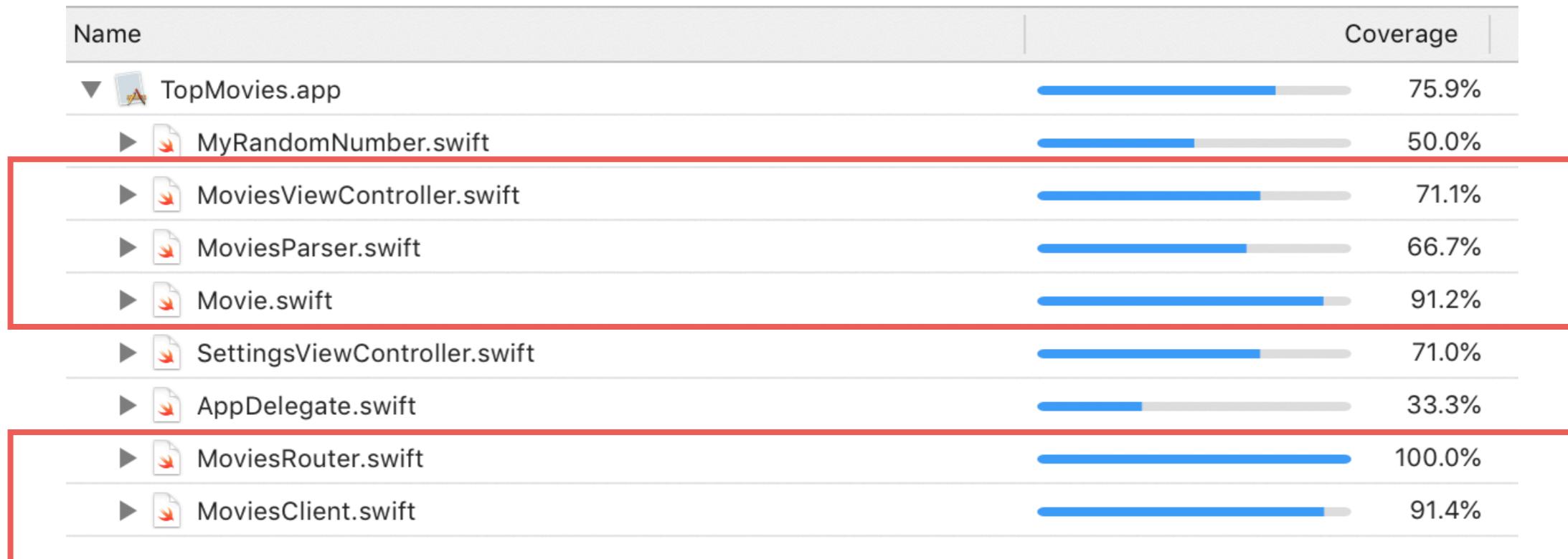
```
class MoviesRouter {
    var limit: Int

    init(limit: Int) {
        self.limit = limit
    }

    func urlRequest() -> URLRequest {
        if let fromTest = ProcessInfo.processInfo.environment["url"].self {
            let url = URL(string: fromTest)!
            return URLRequest(url: url)
        }
        let url = URL(string:
            "https://itunes.apple.com/us/rss/topmovies/limit=\(limit)/json")!
        return URLRequest(url: url)
    }
}
```



# Code coverage



# Add more tests by yourself



# Working with fastlane



# Fastlane

Scan  
Snapshot

Scan Snapshot



# Install Fastlane

```
$gem install fastlane
```



# Using scan

\$fastlane scan init



# Scanfile

\$fastlane scan

```
scheme("TopMovies")
open_report(true)
clean(true)
skip_build(true)
```



# Scan result

\$fastlane scan

Test Results		
Number of tests	12	
Number of failures	0	



# Scan result in ./test\_output

## \$fastlane scan

Test Results

12 tests

All

Failing

Passing

### TopMoviesTests.MyRandomNumberTest

```
0.001s test_random_number_5
0.000s test_random_number_more_than_10
0.000s test_random_should_called_get_1_time
```

### TopMoviesTests.SettingsTests

```
0.006s test_setting_error_message_should_not_display
0.004s test_setting_input_1_should_error_display
0.003s test_setting_input_201_should_error_display
0.007s test_setting_input_2_should_error_not_display
0.003s test_setting_input_2x_should_error_display
0.003s test_setting_input_2x_should_error_message
0.003s test_setting_label_should_display
0.003s test_setting_number_should_placeholder_50
0.003s test_setting_title_page_should_Settings
```

Report generated with [xcpretty](#)



# Using snapshot

## \$fastlane snapshot init

```
[✓] 🚀
✓ Successfully created SnapshotHelper.swift './SnapshotHelper.swift'
✓ Successfully created new Snapfile at './Snapfile'
```

Open your Xcode project and make sure to do the following:

- 1) Add a new UI Test target to your project
- 2) Add the ./fastlane/SnapshotHelper.swift to your UI Test target  
You can move the file anywhere you want
- 3) Call `setupSnapshot(app)` when launching your app

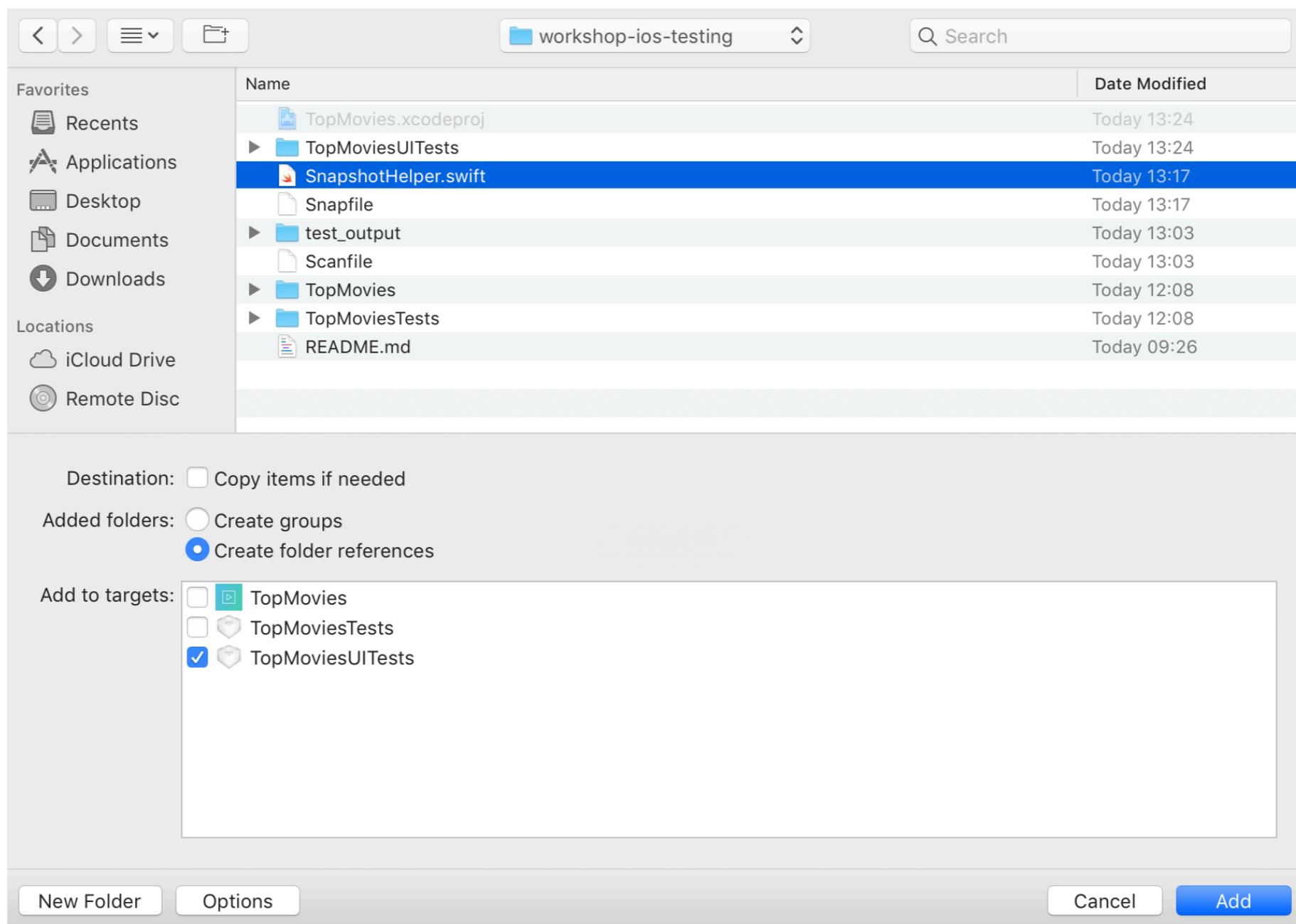
```
let app = XCUIApplication()
setupSnapshot(app)
app.launch()
```

- 4) Add `snapshot("0Launch")` to wherever you want to trigger screenshots
- 5) Add a new Xcode scheme for the newly created UITest target
- 6) Add a Check to enable the `Shared` box of the newly created scheme



# Add SnapshotHelper.swift

## To UITest Target



# Add in test cases

```
class TopMoviesUITests: XCTestCase {

    override func setUp() {
        continueAfterFailure = false
        let app = XCUIApplication()
        setupSnapshot(app)
        app.launch()
    }

    func testExample() {
        snapshot("Step 01")
        snapshot("Step 02")
        snapshot("Step 03")
    }

}
```



# Using snapshot

\$fastlane snapshot

