

<https://github.com/GroundSpeed/BreakingGround>

# UNIT TESTING FOR THE EFFICIENT XCODE EXPERIENCE

Don Miller  
Founder of GroundSpeed™  
Co-founder of Seed Coworking  
@donmiller

# OBJECTIVES

- What is Unit Testing?
- Why Unit Testing?
- Xcode - Setting Up Unit Tests
- Xcode - Creating Performance Tests When Using A 3rd Party API
- Xcode - Testing parse.com Connection
- Xcode - Code Coverage



# WHAT IS UNIT TESTING?

- Test small pieces of your code
- Xcode provides a separate testing environment that allows you to bolt on tests to your application
- Unit testing is a great way to test methods that you are planning to create
- You can start by calling the method as you have planned
- Once test fails, start writing the method to make it pass

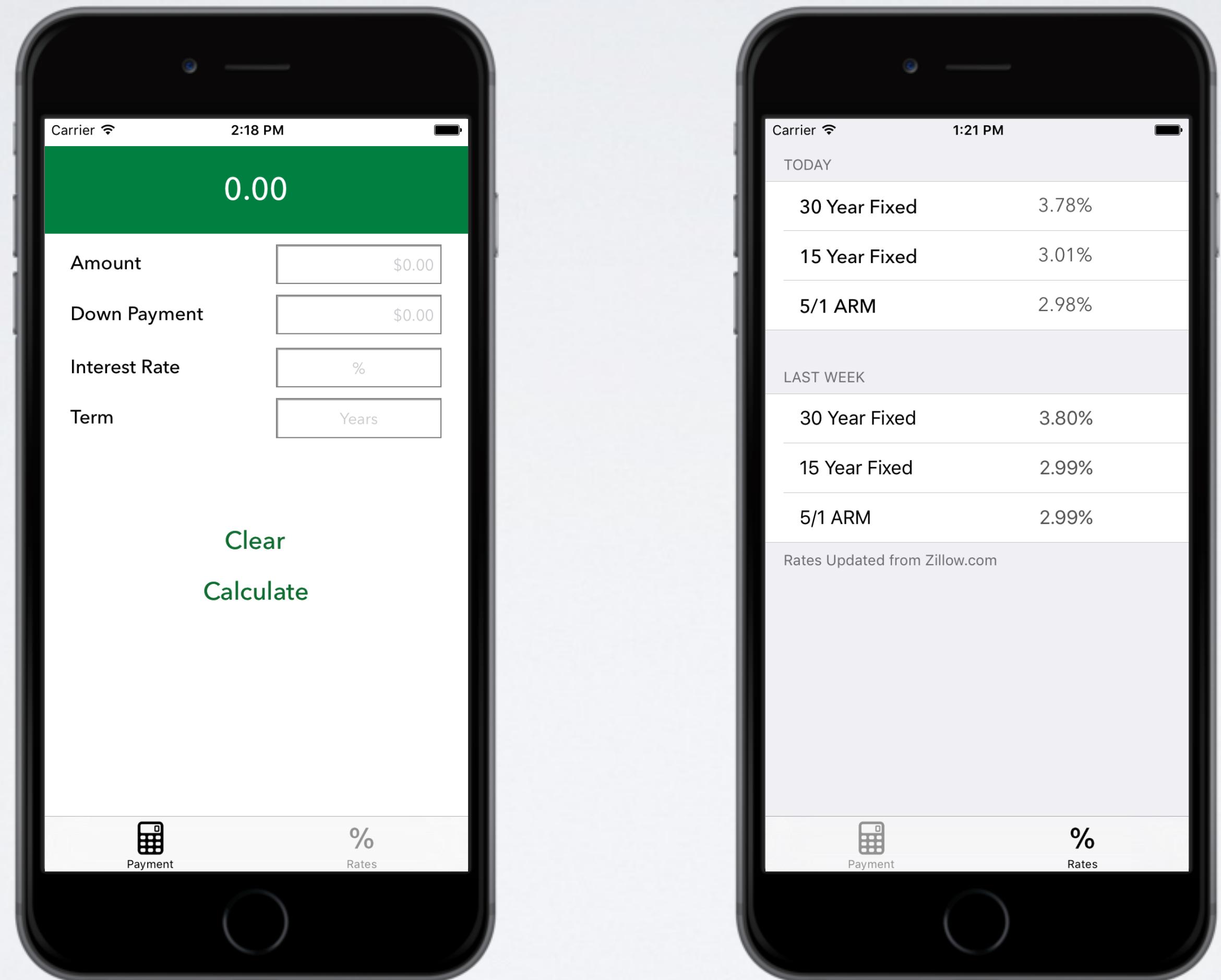


# WHY UNIT TEST?

- Verify your code does what you expect
- Makes refactoring much less painful
- Creates a discipline of creating smaller more concise methods



# APP WE WILL BE TESTING



GroundSpeed™  
rapid web + mobile software

# HOW TO ADD UNIT TESTS?

Choose options for your new project:

Product Name:   

Organization Name:   

Organization Identifier:   

Bundle Identifier: com.goundspeedhq.NewAppToTest

Language:   

Devices:   

Use Core Data

Include Unit Tests

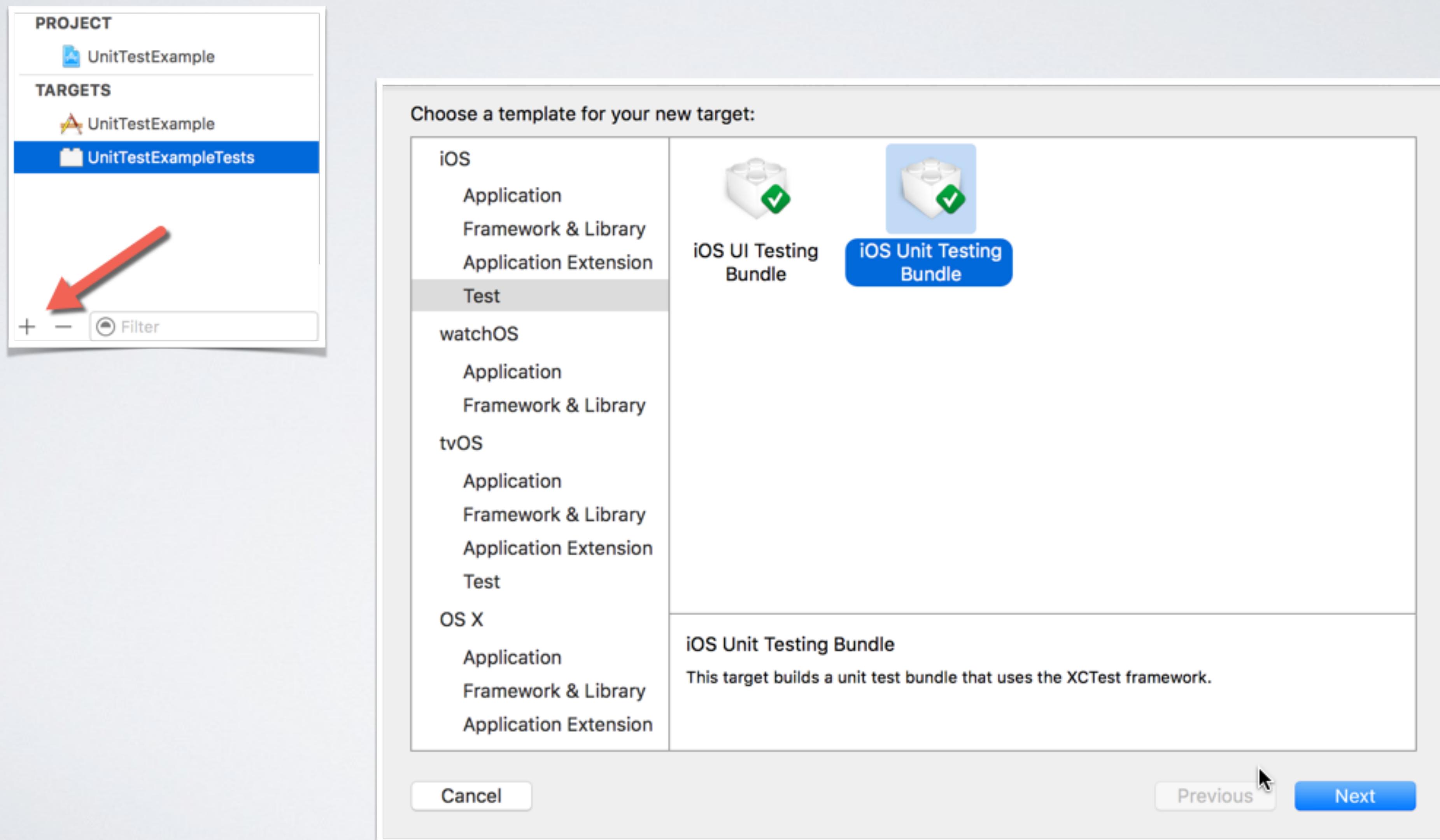
Include UI Tests

→ Make sure to check box

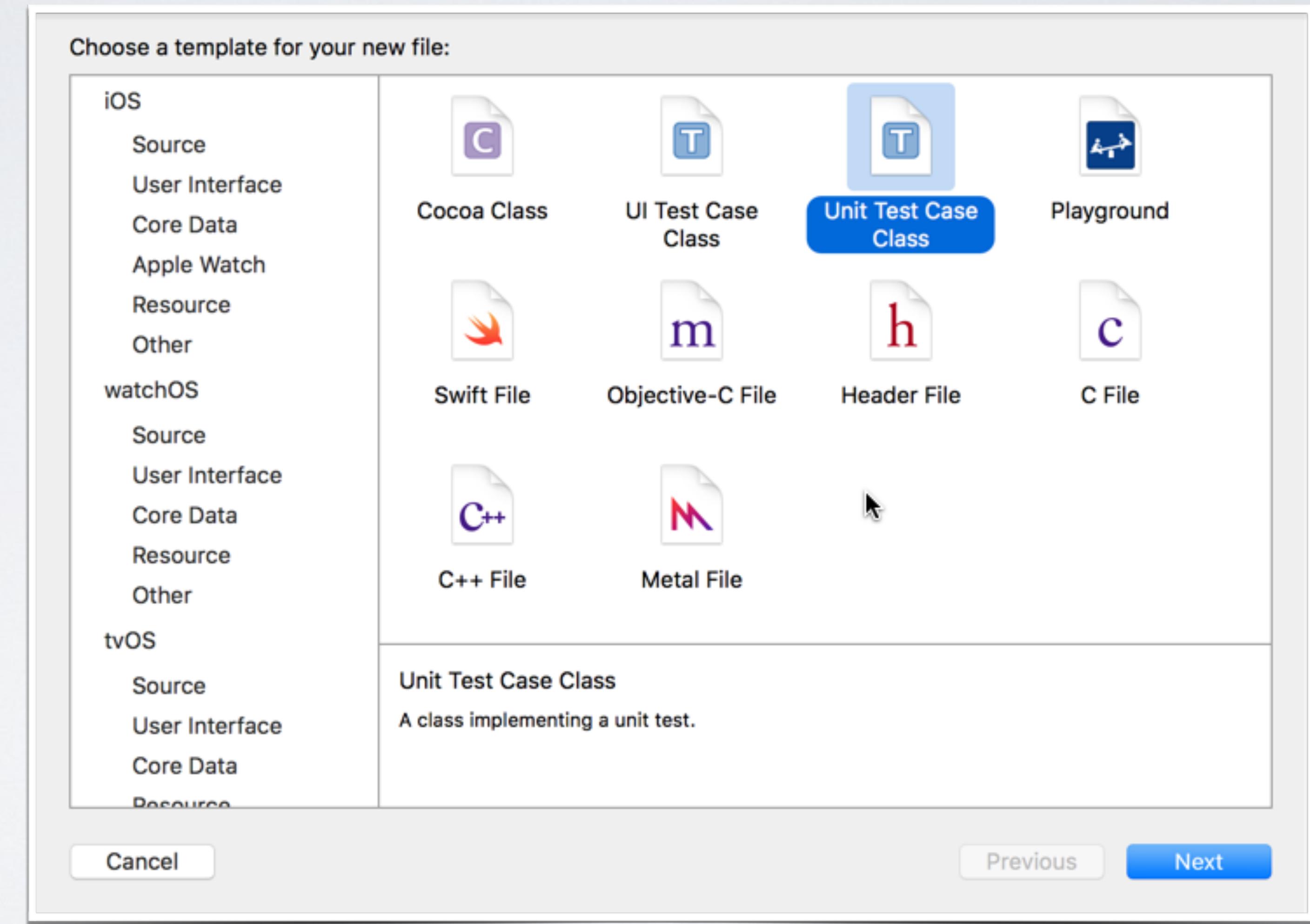
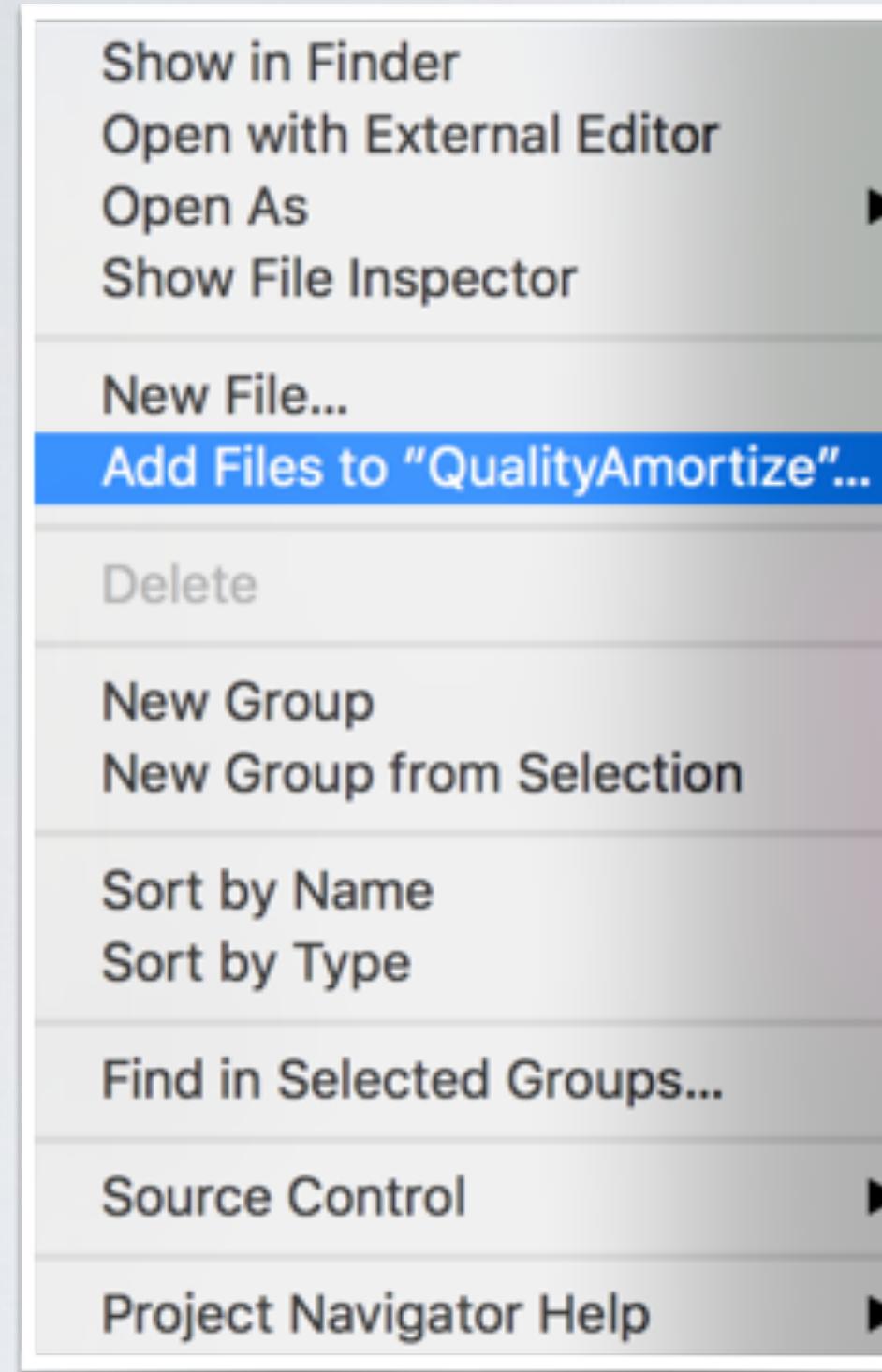
Cancel Previous Next



# HOW TO ADD TESTS TO AN EXISTING PROJECT



# ADDING NEW TEST CLASSES



# UNIT TEST CONVENTIONS

- All Unit Test methods should begin with “test”
- Unit Test methods cannot have parameters
- Unit Test methods are subclasses of XCTestCase
- There are no restrictions on the amount of test classes or tests
- To run code prior to each test, override the setup method
- To run code after each test, override the teardown method



# FUNDAMENTAL TEST

- All of the XCTest assertions come down to a single, base assertion

```
XCTAssert(expression, format...)
```

- If the expression evaluates to true, the test passes. Otherwise, the test fails, printing the formatted message.



# BOOLEAN TESTS

- For Bool values, or simple boolean expressions, use `XCTAssertTrue` & `XCTAssertFalse`

```
[XCTAssertTrue(expression, format...)  
XCTAssertFalse(expression, format...)]
```



# EQUALITY TESTS

- When testing whether two values are equal, use `XCTAssert[Not]Equal` for scalar values and `XCTAssert[Not]EqualObjects` for objects

```
XCTAssertEqual(expression1, expression2, format...)
XCTAssertNotEqual(expression1, expression2, format...)
```

```
func testOnePlusOneEqualsTwo() {
    XCTAssertEqual(1 + 1, 2, "one plus one should equal two")
}
```



# NIL TESTS

- Use `XCTAssert[Not]Nil` to assert the existence (or non-existence) of a given value

```
XCTAssertNil(expression, format...)
XCTAssertNotNil(expression, format...)
```



# UNCONDITIONAL FAILURE

- The `XCTFail` assertion will always fail

```
XCTFail(format...)
```



# PERFORMANCE TESTING

```
func testDateFormatterPerformance() {  
    let dateFormatter = NSDateFormatter()  
    dateFormatter.dateStyle = .LongStyle  
    dateFormatter.timeStyle = .ShortStyle  
  
    let date = NSDate()  
  
    measureBlock() {  
        let string = dateFormatter.stringFromDate(date)  
    }  
}
```



# XCTEST EXPECTATION

```
func testPerformanceZillowConnection() {
    self.measureBlock {
        self.callAsynchronousZillowConnection()
    }
}

func callAsynchronousZillowConnection() {

    let URL = NSURL(string: GlobalHelper().kPostEndpoint)!
    let expectation = expectationWithDescription("GET \(URL)")

    let session = NSURLSession.sharedSession()
    let task = session.dataTaskWithURL(URL) { data, response, error in
        XCTAssertNotNil(data, "data should not be nil")
        XCTAssertNil(error, "error should be nil")

        if let HTTPResponse = response as? NSHTTPURLResponse,
            responseURL = HTTPResponse.URL,
            MIMEType = HTTPResponse.MIMEType
        {
            XCTAssertEqual(responseURL.absoluteString, URL.absoluteString, "HTTP response URL should be equal to original URL")
            XCTAssertEqual(HTTPResponse.statusCode, 200, "HTTP response status code should be 200")
            XCTAssertEqual(MIMEType, "application/json", "HTTP response content type should be application/json")
        } else {
            XCTFail("Response was not NSHTTPURLResponse")
        }
        expectation.fulfill()
    }
    task.resume()
    waitForExpectationsWithTimeout(task.originalRequest!.timeoutInterval) { error in
        if let error = error {
            print("Error: \(error.localizedDescription)")
        }
        task.cancel()
    }
}
```



```

import UIKit

class GlobalHelper: NSObject {

    static let kApiKey = "X1-ZWz1f3t3bvl4i3_1brbp"
    let kPostEndpoint = "https://www.zillow.com/webservice/GetRateSummary.htm?zws-id=\(kApiKey)&output=json"

    func getMonthlyPayment(amount: Float, downPayment: Float, term: Float, interestRate: Float, lblPayment: UILabel) -> UILabel {
        // A = payment Amount per period
        // P = initial Principal (loan amount)
        // r = interest rate per period
        // n = total number of payments or periods

        let principal : Float = amount - downPayment
        let payments = term*12
        let rate = interestRate/12/100
        let amount = calculatPMTWithRatePerPeriod(rate, numberOfPayments: payments, loanAmount: principal, futureValue: 0, type: 0)

        if (isnan(amount) || isinf(amount))
        {
            lblPayment.font = UIFont.boldSystemFontOfSize(18)
            lblPayment.textColor = UIColor.redColor()
            lblPayment.text = "You must enter all required fields.";
        }
        else
        {
            lblPayment.font = UIFont(name: "Avenir Next", size: 28)
            lblPayment.textColor = UIColor.whiteColor()
            lblPayment.text = String(format: "%.02f", amount)
        }

        return lblPayment
    }

    func calculatPMTWithRatePerPeriod (ratePerPeriod: Float, numberOfPayments: Float, loanAmount: Float, futureValue: Float, type: Float) -> Float {
        var q : Float
        q = pow(1 + ratePerPeriod, numberOfPayments)

        let returnValue = (ratePerPeriod * (futureValue + (q * loanAmount))) / ((-1 + q) * (1 + ratePerPeriod * (type)))

        return returnValue
    }
}

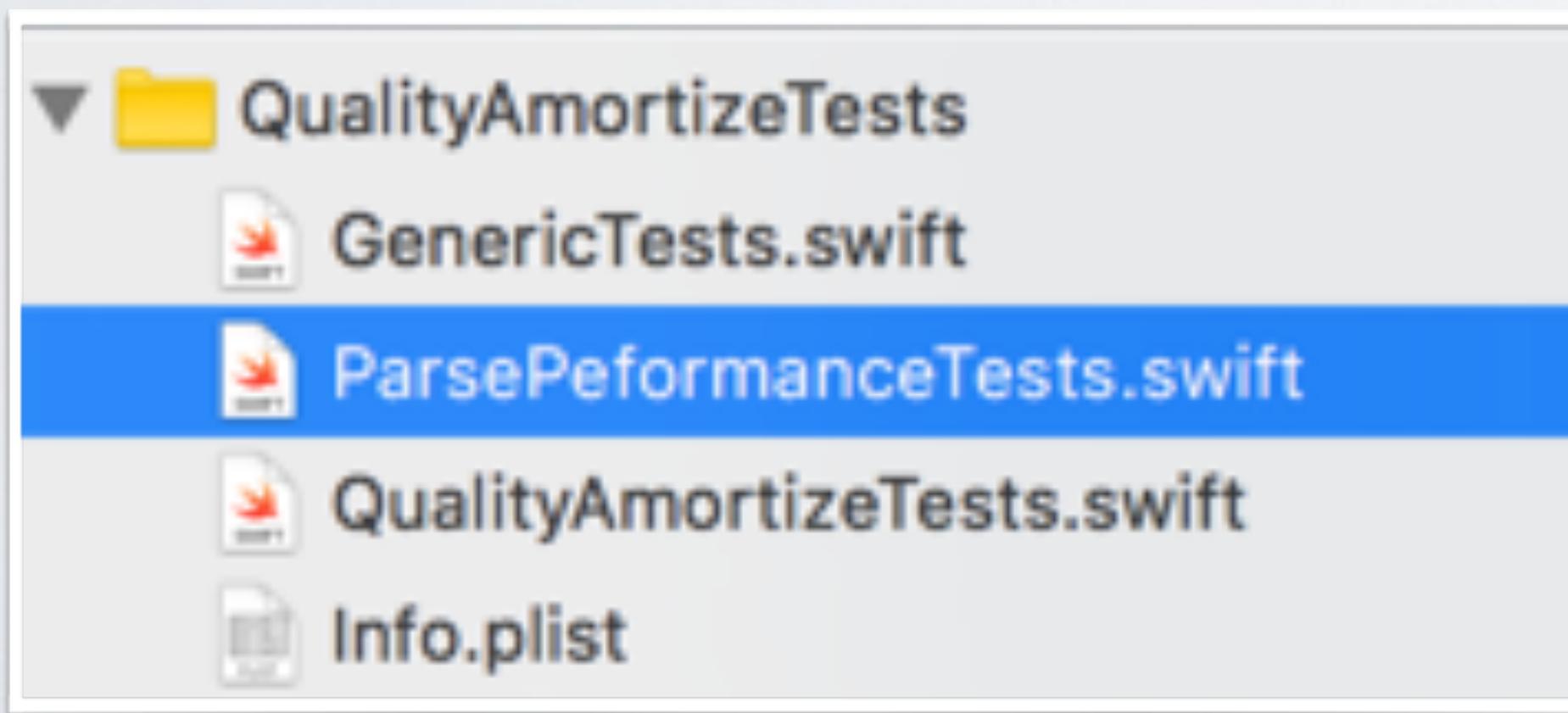
extension String {
    func toPercent() -> String {
        return String.localizedStringWithFormat("%.2f%%", Float(self)!)
    }
}

```



# @TESTABLE

- @testable solves the problem of previous Swift versions by giving you access to everything that is internal
- Access levels were different in Objective-C and this problem was not noticed until Swift 1.0



```
import XCTest
@testable import QualityAmortize
```



# TEST OUR GETPAYMENT() METHOD

```
24 func testGetPayment() {
25     let amount: Float = 100000.00
26     let downPayment: Float = 1000.00
27     let interestRate: Float = 5.0
28     let term: Float = 30
29     let label : UILabel = UILabel()
30
31     let result = GlobalHelper().getMonthlyPayment(amount, downPayment: downPayment, term: term, interestRate:
32         interestRate, lblPayment: label)
33
34     XCTAssert(result.text == "531.45", "\u{result.text} should equal 531.45")
35     XCTAssert(result.textColor == UIColor.whiteColor(), "\u{result.textColor} should be white")
36 }
37
38 func testGetPaymentBadNumber() {
39     let amount: Float = 0
40     let downPayment: Float = 0
41     let interestRate: Float = 0
42     let term: Float = 0
43     let label : UILabel = UILabel()
44
45     let result = GlobalHelper().getMonthlyPayment(amount, downPayment: downPayment, term: term, interestRate:
46         interestRate, lblPayment: label)
47
48     XCTAssert(result.text == "You must enter all required fields.", "\u{result.text} should read You must enter all
        required fields.")
49     XCTAssert(result.textColor == UIColor.redColor(), "\u{result.textColor} should be red")
50 }
```

# TEST CALCULATEPAYMENT() METHOD

```
51 func testCalculatePayment() {  
52     let amount: Float = 100000.00  
53     let downPayment: Float = 1000.00  
54     let interestRate: Float = 5.0  
55     let term: Float = 30  
56  
57     let principal : Float = amount - downPayment  
58     let payments = term*12  
59     let rate = interestRate/12/100  
60  
61     var result = GlobalHelper().calculatPMTWithRatePerPeriod(rate, numberOfPayments: payments, loanAmount: principal,  
62         futureValue: 0.0, type: 0.0)  
63     result = round(100*result)/100 //Round to two decimal places  
64     XCTAssert(result == 531.45, "\u005C(result) should equal 531.45")  
65 }
```



# TEST THE ZILLOW API JSON CALL

```
67 func testJsonCall() {  
68     let zillowRates : Dictionary<String, String> = ApiHelper().getRatesFromZillow()  
69  
70     XCTAssert(zillowRates.count == 6, "\u{zillowRates.count} should include 6 records")  
71 }
```



# TEST THE RATES MODEL

```
73 func testRatesModel() {  
74     let rates = Rates()  
75  
76     rates.thirtyYearFixed = "2.00"  
77     rates.fifteenYearFixed = "1.00"  
78     rates.fiveOneARM = "1.50"  
79  
80     XCTAssertNotNil(rates.thirtyYearFixed)  
81     XCTAssertNotNil(rates.fifteenYearFixed)  
82     XCTAssertNotNil(rates.fiveOneARM)  
83 }  
84  
85 func testRatesModelForPercentSign() {  
86     let rates = Rates()  
87  
88     rates.thirtyYearFixed = "2.00".toPercent()  
89     rates.fifteenYearFixed = "1.00".toPercent()  
90     rates.fiveOneARM = "1.50".toPercent()  
91  
92     XCTAssertTrue(rates.thirtyYearFixed.characters.last! == "%", "\u{rates.thirtyYearFixed} should end with a percent")  
93     XCTAssertTrue(rates.fifteenYearFixed.characters.last! == "%", "\u{rates.thirtyYearFixed} should end with a percent")  
94     XCTAssertTrue(rates.fiveOneARM.characters.last! == "%", "\u{rates.thirtyYearFixed} should end with a percent")  
95 }
```



# CALL ZILLOW TO VERIFY A 200 JSON RESPONSE

```
◇97 func testPerformanceZillowConnection() {
98     self.measureBlock {
99         self.callAsynchronousZillowConnection()
100    }
101 }
102
103 func callAsynchronousZillowConnection() {
104
105     let URL = NSURL(string: GlobalHelper().kPostEndpoint)!
106     let expectation = expectationWithDescription("GET \(URL)")
107
108     let session = NSURLSession.sharedSession()
109     let task = session.dataTaskWithURL(URL) { data, response, error in
110         XCTAssertNotNil(data, "data should not be nil")
111         XCTAssertNil(error, "error should be nil")
112
113         if let HTTPResponse = response as? NSHTTPURLResponse,
114             responseURL = HTTPResponse.URL,
115             MIMEType = HTTPResponse.MIMEType
116         {
117             XCTAssertEqual(responseURL.absoluteString, URL.absoluteString, "HTTP response URL should be equal
118             to original URL")
119             XCTAssertEqual(HTTPResponse.statusCode, 200, "HTTP response status code should be 200")
120             XCTAssertEqual(MIMEType, "application/json", "HTTP response content type should be application/
121                 json")
122         } else {
123             XCTFail("Response was not NSHTTPURLResponse")
124         }
125
126         expectation.fulfill()
127     }
128     task.resume()
129     waitForExpectationsWithTimeout(task.originalRequest!.timeoutInterval) { error in
130         if let error = error {
131             print("Error: \(error.localizedDescription)")
132         }
133         task.cancel()
134     }
}
```



# WORKING WITH PARSE

The screenshot shows the Parse.com dashboard interface. At the top, there's a navigation bar with tabs for Core, Analytics, Push, Settings, and Docs. A user profile for "Don Miller" is also visible. Below the navigation is a table titled "Data" with columns for Role, objectId, foo, createdAt, updatedAt, and ACL. There are two entries: one for "Role" (objectId 0) and one for "User" (objectId 1). The User entry has values: foo = "bar", createdAt = "Jan 08, 2016, 06:28", updatedAt = "Jan 08, 2016, 06:28", and ACL = "Public Read and Write". Below the table is a code editor window displaying Swift code for an AppDelegate:

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    // Override point for customization after application launch.

    Parse.enableLocalDatastore()

    // Initialize Parse.
    Parse.setApplicationId("yDEI...Pc",
        clientKey: "GUT...IJo")

    // [Optional] Track statistics around application opens.
    PFAnalytics.trackAppOpenedWithLaunchOptions(launchOptions)

    return true
}
```

A callout box with the text "setup in AppDelegate" points to the line "Parse.setApplicationId". At the bottom of the code editor, there's a note: "setup in AppDelegate". The sidebar on the left contains links for Config and API Console. At the very bottom, there are links for Docs, Billing, Downloads, Help, Status, Blog, and Parse.com.



# CREATE TEST TO CHECK FOR PARSE CORRECT SETUP

```
9 import XCTest
10 import Parse
11 import Bolts
12
13 class ParsePerformanceTests: XCTestCase {
14
15     func testParsePerformanceOnMainThread() {
16         // This is an example of a performance test case.
17         self.measureBlock {
18             self.checkParseIsSetup()
19         }
20     }
21
22     func checkParseIsSetup() {
23         let testObject = PFObject(className: "TestObject")
24         testObject["foo"] = "bar"
25
26         do { try testObject.save()
27             print("Object has been saved.")
28         }
29         catch let e as NSError { print("Parse save error: \(e)") }
30     }
31 }
```



# TEST PARSE PERFORMANCE

```
44 func testParsePerformanceGettingProducts() {
45     // This is an example of a performance test case.
46     self.measureBlock {
47         self.checkParsePullingData()
48     }
49 }
50
51 func checkParsePullingData() {
52     let query = PFQuery(className:"Products")
53
54     do { let objects = try query.findObjects()
55         for object in objects {
56             print(object.objectId)
57         }
58     }
59     catch let e as NSError { print("Parse load error: \(e)") }
60 }
```



# TEST LOADING OF PRODUCTS OBJECT FROM PARSE

```
12 class Products {
13
14     var objectId : String?
15     var productId : Int?
16     var photo : String?
17     var photoFile : PFFile = PFFile()
18     var photoImage : UIImage?
19     var title : String?
20     var flyer : String?
21     var productDesc : String?
22     var itemCode : String?
23     var packagingOptions : String?
24     var freeShipping : Bool?
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
61
62 func testGettingRatesCollection() {
63     let arrayProducts : Array<Products> = Products().getAllProductsFromParse()
64
65     XCTAssert(arrayProducts.count > 0, "\u{arrayProducts.count} should be greater than zero")
66 }
```

```
30     let query = PFQuery(className: "Products",
31
32     var arrayProducts : Array<Products> = []
33
34     do { let objects = try query.findObjects()
35         for object in objects {
36             self.objectId = object["objectId"] as? String
37             self.productId = object["Id"] as? Int
38             self.photoImage = object["PhotoImage"] as? UIImage
39             self.title = object["Title"] as? String
40             self.productDesc = object["Description"] as? String
41
42                 arrayProducts.append(self)
43
44             }
45         }
46     } catch let e as NSError { print("Parse load error: \(e)") }
47
48
49     return arrayProducts
50 }
```



# CODE COVERAGE

Screenshot of a code coverage tool interface showing test results for the project "QualityAmortize".

The interface includes a toolbar with icons for file, search, and refresh, followed by a breadcrumb trail: QualityAmortize > Test QualityAmortize : 11:15:46 PM.

Filtering options include "By Group" (selected) and "By Time".

Test summary for "Test Today, 11:15 PM":

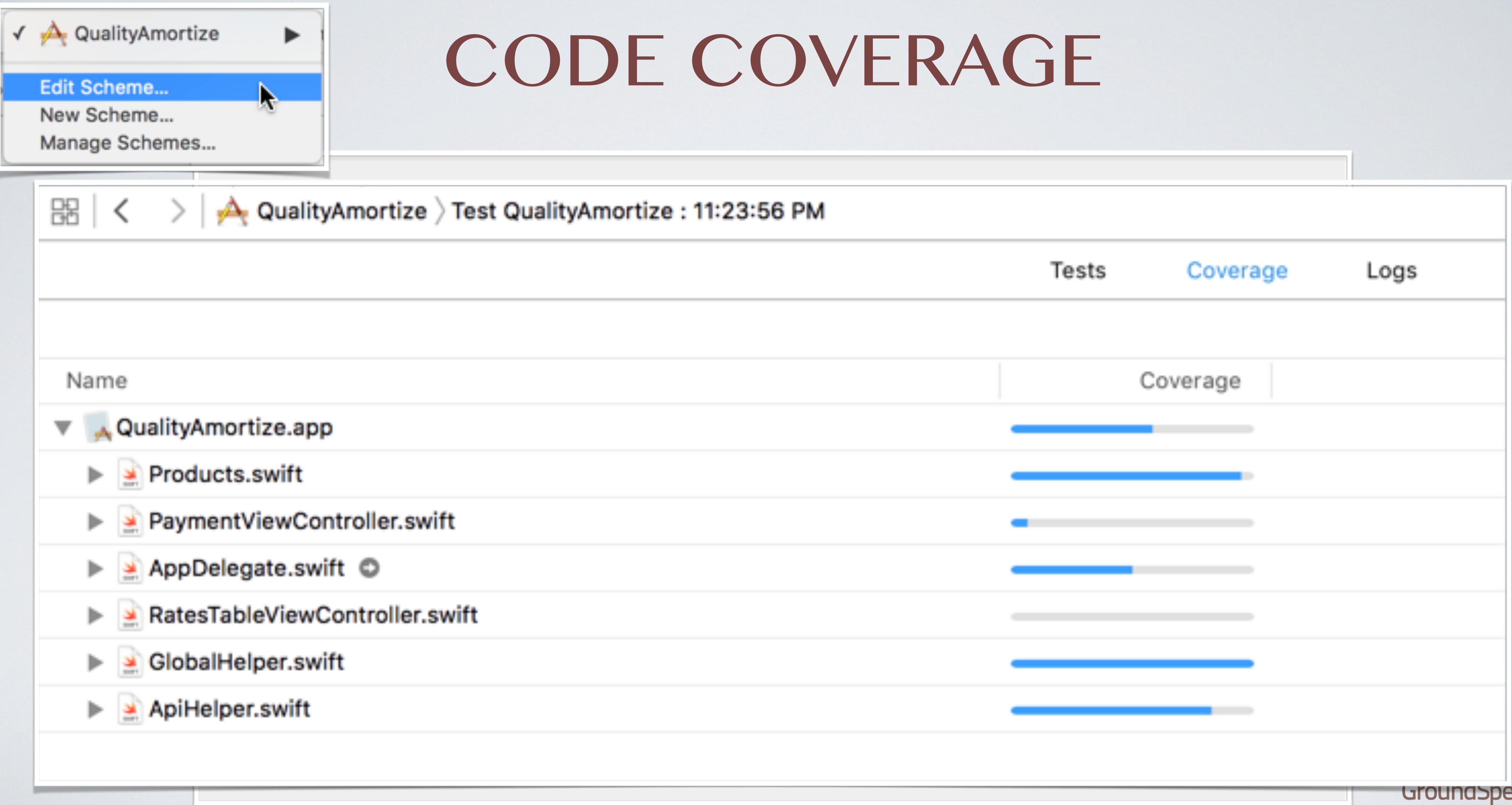
- All tests passed.
- Performance metrics: All tests passed, 0.00 s total time.

Test details:

- GenericTests > QualityAmortizeTests**
  - testDateFormatterPerformance() - Passed, 0.00 s
  - testOnePlusOneEqualsTwo() - Passed
- ParsePerformanceTests > QualityAmortizeTests**
  - testGettingRatesCollection() - Passed
  - testParsePerformanceGettingProducts() - Passed, 0.45 s
  - testParsePerformanceOnMainThread() - Passed, 0.33 s
- QualityAmortizeTests > QualityAmortizeTests**
  - testCalculatePayment() - Passed
  - testGetPayment() - Passed
  - testGetPaymentBadNumber() - Passed
  - testJsonCall() - Passed
  - testPerformanceZillowConnection() - Passed, 0.23 s
  - testRatesModel() - Passed
  - testRatesModelForPercentSign() - Passed



# CODE COVERAGE



# QUESTIONS? THANK YOU!

**Don Miller**

**don@GroundSpeedHQ.com**

**@donmiller**

**http://github.com/donmiller**

**http://github.com/groundspeed**



**GroundSpeed™**  
rapid web + mobile software