Behavioral
Driven Development

“

More than the act of testing, the act of designing tests is one of the best bug preventers known.

Boris Beizer

”

BDD

an idea about how software development should be managed by business interests

Given [initial context], when [event occurs], then [ensure some outcomes]

Instead of writing tests you should think of specifying behavior. *Behavior* is how the user wants the application to behave.

Behavior of the system, looking from the outside

TDD

1. Write a test
2. Run the test, it should fail
3. Write the minimum code required to pass the test
4. Run the test to check if it passes
5. Optionally refactor your code
6. Repeat

Objective C

Kiwi: Simple BDD for iOS

build passing

Kiwi is a Behavior Driven Development library for iOS development. The goal is to provide a BDD library that is exquisitely simple to setup and use.

Why?

The idea behind Kiwi is to have tests that are more readable than what is possible with the bundled test framework.

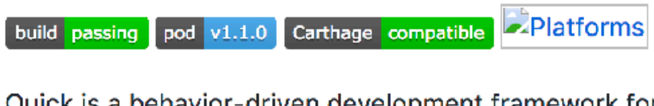
Tests (or rather specs) are written in Objective-C and run within the comfort of Xcode to provide a test environment that is as unobtrusive and seamless as possible in terms of running tests and error reporting.

Specs look like this:

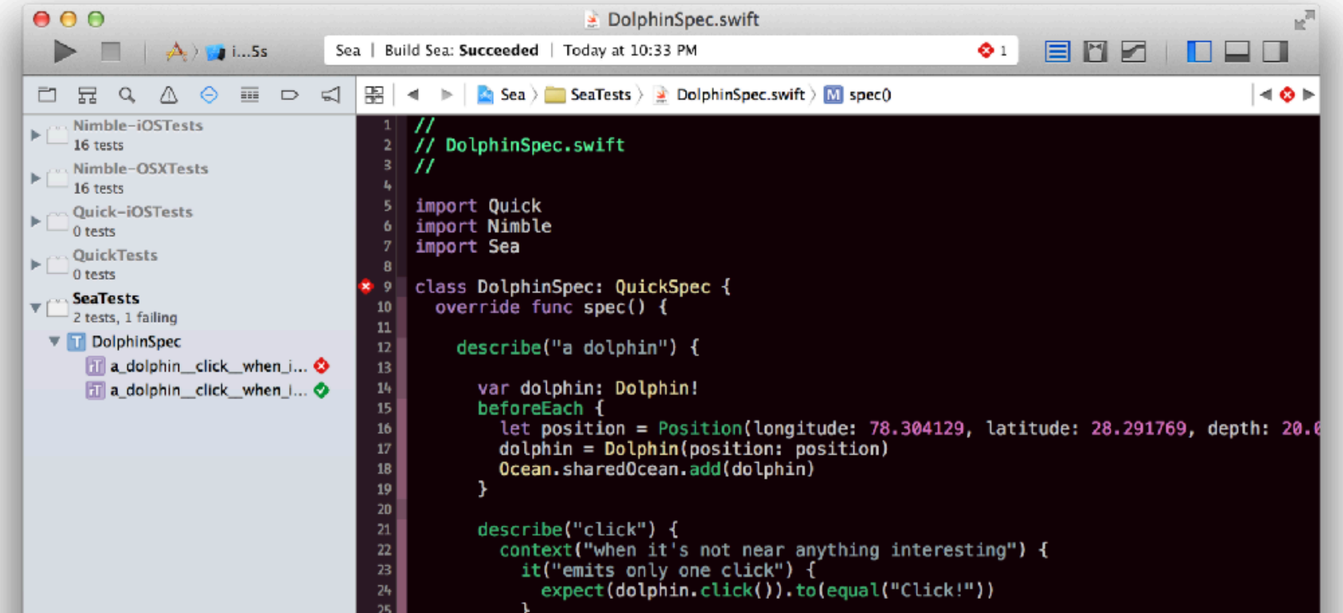
```
describe(@"Team", ^{
```

Swift

README.md

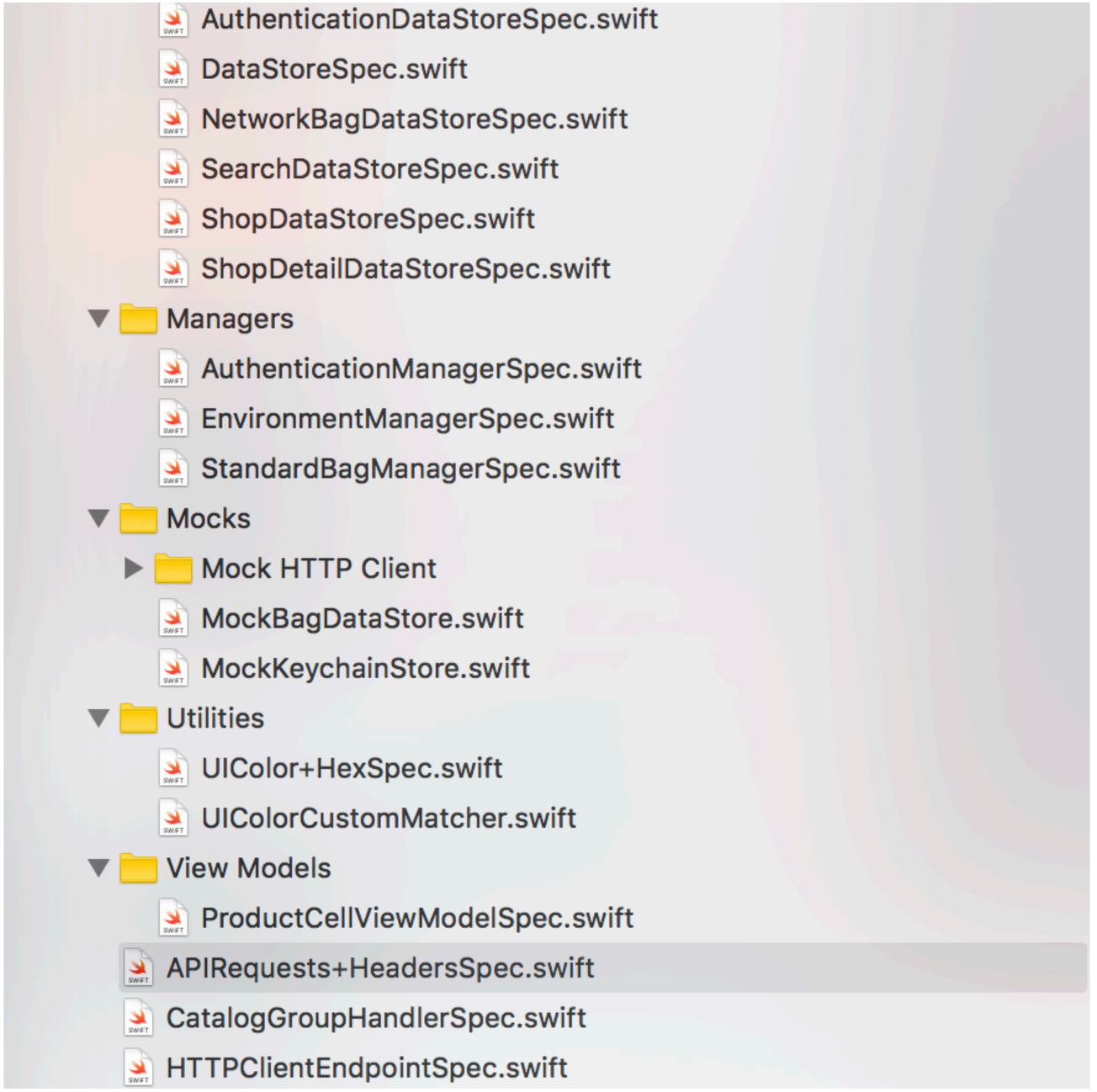


Quick is a behavior-driven development framework for Swift and Objective-C. Inspired by RSpec, Specta, and Ginkgo.



What does a BDD test look like?

<type-name>Spec.swift


AuthenticationDataStoreSpec.swift
DataStoreSpec.swift
NetworkBagDataStoreSpec.swift
SearchDataStoreSpec.swift
ShopDataStoreSpec.swift
ShopDetailDataStoreSpec.swift
Managers
AuthenticationManagerSpec.swift
EnvironmentManagerSpec.swift
StandardBagManagerSpec.swift
Mocks
Mock HTTP Client
MockBagDataStore.swift
MockKeychainStore.swift
Utilities
UIColor+HexSpec.swift
UIColorCustomMatcher.swift
View Models
ProductCellViewModelSpec.swift
APIRequests+HeadersSpec.swift
CatalogGroupHandlerSpec.swift
HTTPClientEndpointSpec.swift

```swift
import Nimble
import enum Result.Result
@testable import BeautyCounterCore

// swiftlint:disable function_body_length force_cast

class AuthenticationDataStoreSpec: QuickSpec {
    override func spec() {

        describe("when authenticating a user") {
            var username: String!
            var password: String!
            var responseStub: Any!

            context("and a valid username, valid password is provided", {
                beforeEach {
                    username = "fakeUser"
                    password = "fakePass"
                    responseStub = MockHTTPClient.stub(forFilename: "post_auth")
                    MockHTTPClient.mockResult = .success(responseStub)
                    AuthenticationDataStore.httpClient = MockHTTPClient.self
                }

                it("it should call completion with a session token and a consultant tuple", closure: {
                    waitUntil(action: { (done) in
                        AuthenticationDataStore.authenticateUser(username: username,
                                                                 password: password,
                                                                 completion: { (result) in
                            expect(result.error).to(beNil())
                            expect(result.value?.authToken).to(equal("7a1dc3c2eb21808a044ffddf352d4575|10211"))
                            expect(result.value?.consultant).toNot(beNil())
                            done()
                        })
                    })
                })
            })
```

Test method names should be sentences

An expressive test name is more useful when a test fails

BeautyCounter 〉 Test BeautyCounter : 10:44:42 AM

Tests Coverage Logs

All Passed Failed All Performance Filter

Tests	Status
APIRequests_HeadersSpec › BeautyCounterCoreTests	
when_getting_the_default_headers__and_an_auth_token_exists_in_the_keychain__should_add_the_auth_token_as_a_string_to_the_headers()	✓
when_getting_the_default_headers__and_no_auth_token_exists_in_the_keychain__should_not_add_the_auth_token_to_the_headers()	✓
when_getting_the_default_headers__and_the_country_code_exists_in_the_user_defaults__should_add_the_country_code_as_a_string_to_the_headers()	✓
when_getting_the_default_headers__and_the_country_code_does_not_exist_in_the_keychain__should_not_add_the_country_code_to_the_headers()	✓
combining_custom_headers_with_default_headers__and_unique_headers_are_provided__should_return_a_dictionary_containing_both_the_custom_and_default_header_key_values()	✓
combining_custom_headers_with_default_headers__and_default_header_override_values_are_provided__should_override_the_default_header_values_in_the_returned_dictionary()	✓
AuthenticationDataStoreSpec › BeautyCounterCoreTests	
when_authenticating_a_user__and_a_valid_username__valid_password_is_provided__it_should_call_completion_with_a_session_token_and_a_consultant_tuple()	✓
when_authenticating_a_user__and_an_invalid_username_or_invalid_password_is_provided__should_call_completion_with_an_unauthorized_error()	✓
when_authenticating_a_user__and_an_invalid_response_is_provided__should_call_completion_with_an_jsonDeserialization_error()	✓
when_deserializing_an_Auth_token__and_valid_JSON_is_provided__should_return_a_valid_auth_token()	✓
when_deserializing_an_Auth_token__and_invalid_JSON_is_provided__should_return_a_jsonDeserialization_error()	✓
when_deserializing_a_consultant__and_valid_JSON_is_provided__should_return_a_valid_consultant()	✓
when_deserializing_a_consultant__and_invalid_JSON_is_provided__should_return_a_jsonDeserialization_error()	✓
AuthenticationManagerSpec › BeautyCounterCoreTests	
when_checking_if_a_user_is_authenticated__and_an_auth_token_exists__should_return_true()	✓
when_checking_if_a_user_is_authenticated__and_an_auth_token_does_not_exist__should_return_false()	✓
when_authenticating_a_user__and_authentication_succeeds__should_save_the_auth_token_to_the_keychain_and_consultant_to_user_defaults()	✓
when_authenticating_a_user__and_authentication_request_returns_unauthorized__should_call_completion_with_the_client_unauthorized_error()	✓
when_signing_out_a_user__and_the_auth_token_and_country_code_are_saved__should_clear_the_keychain_and_user_defaults()	✓
DataStoreSpec › BeautyCounterCoreTests	
when_deserializing_data_to_JSON__and_valid_JSON_data_is_provided__should_return_a_valid_JSON_object()	✓
when_deserializing_data_to_JSON__and_invalid_JSON_data_is_provided__should_return_json_deserialization_error()	✓
EnvironmentManagerSpec › BeautyCounterCoreTests	
when_getting_the_current_environment__and_a_valid_override_is_specified_in_user_defaults__should_return_the_override_value()	✓
when_getting_the_current_environment__and_an_invalid_override_is_specified_in_user_defaults__and_a_valid_default_environment_is_specified_in_build_settings__should_return_the_default_environment()	✓
when_getting_the_current_environment__and_an_invalid_override_is_specified_in_user_defaults__and_an_invalid_default_environment_is_specified_in_build_settings__should_return_the_fallback_default_environm	✓
when_getting_the_current_environment__and_an_invalid_override_is_specified_in_user_defaults__and_no_default_environment_is_specified_in_build_settings__should_return_the_fallback_default_environment__re	✓
when_getting_the_current_environment__and_no_override_is_specified_in_user_defaults__and_a_valid_default_environment_is_specified_in_build_settings__should_return_the_default_environment()	✓
when_getting_the_current_environment__and_no_override_is_specified_in_user_defaults__and_an_invalid_default_environment_is_specified_in_build_settings__should_return_the_fallback_default_environment__re	✓
when_getting_the_current_environment__and_no_override_is_specified_in_user_defaults__and_no_default_environment_is_specified_in_build_settings__should_return_the_fallback_default_environment()	✓

```swift
//
//  OnePasswordTests.swift
//  PCFSwift
//
//  Created by Thibault Klein on 4/10/17.
//  Copyright © 2017 CocoaPods. All rights reserved.
//

import XCTest
@testable import PCFSwift

class OnePasswordTests: XCTestCase {

    func testOnePasswordUser() {
        let user = OnePasswordUser(email: "Bruce Wayne", password: "Gotham")
        XCTAssertEqual(user.email, "Bruce Wayne")
        XCTAssertEqual(user.password, "Gotham")
    }

    func testOnePasswordIsAvailable() {
        let manager = OnePasswordManager()

        XCTAssertFalse(manager.available)
    }

    func testOnePasswordButtonValid() {
        let manager = OnePasswordManager()

        do {
            let button = try manager.button()
            XCTAssertNotNil(button.imageView?.image)
        } catch _ {
            XCTFail()
        }
    }
```

Matchers

Expect(X)toEqual(Y)

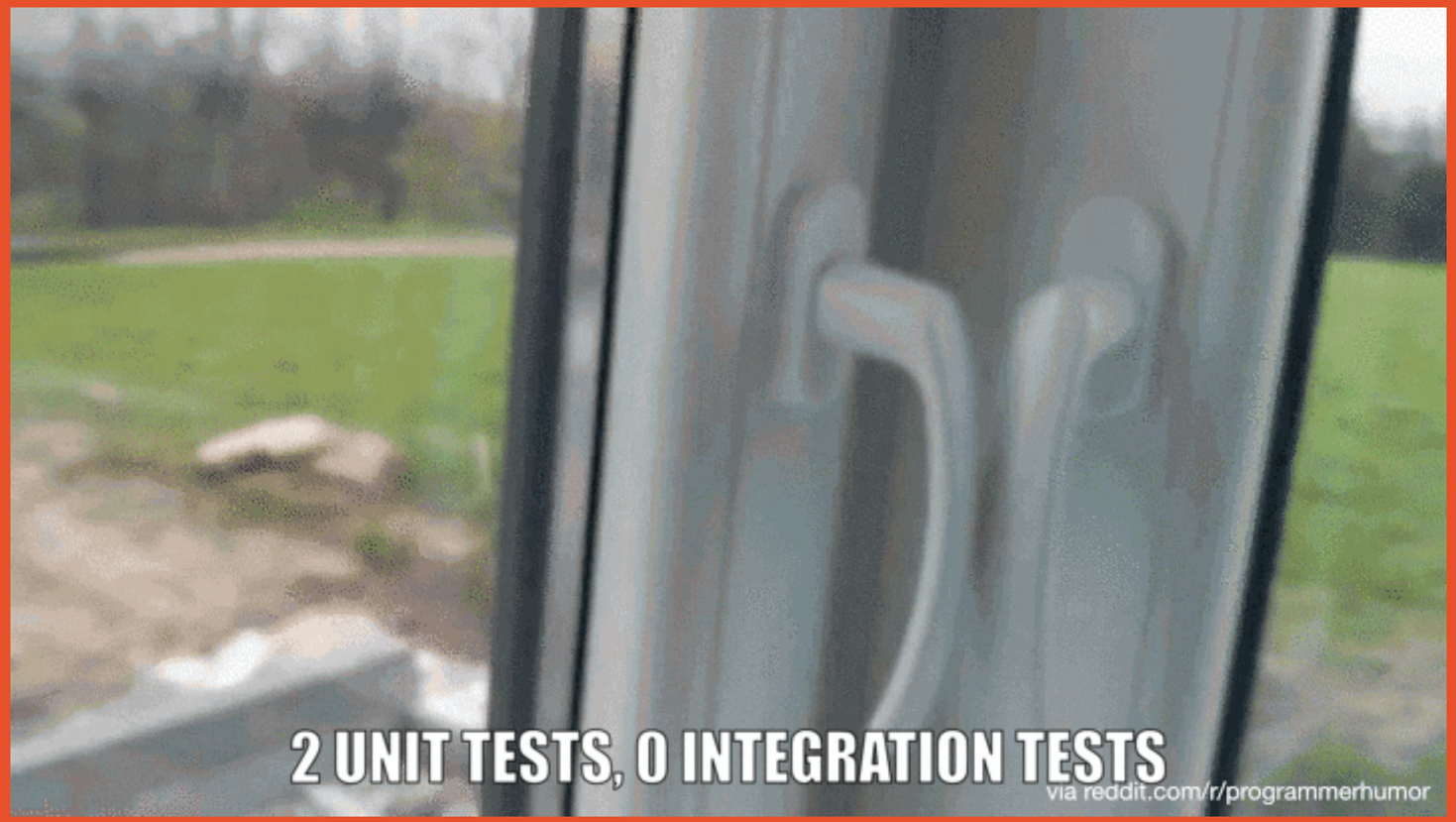
```
expect(someObject).toBe(Nil)
```

Quick + Nimble

Use Nimble to express the expected outcomes of Swift or Objective-C expressions. Inspired by Cedar.

```
// Swift
expect(1 + 1).to(equal(2))
expect(1.2).to(beCloseTo(1.1, within: 0.1))
expect(3) > 2
expect("seahorse").to(contain("sea"))
expect(["Atlantic", "Pacific"]).toNot(contain("Mississippi"))
expect(ocean.isClean).toEventually(beTruthy())
```

Downsides


2 UNIT TESTS, 0 INTEGRATION TESTS
via reddit.com/r/programmerhumor

Writing tests before writing code seems counterintuitive

But it will mean moving faster in the long term

Quick and Nimble integration with Xcode is *very* bad

Xcode’s Syntax Highlighting Crashes almost every single time I touch a test

Onboarding new team members onto a project who don’t know BDD can be difficult

Thank you