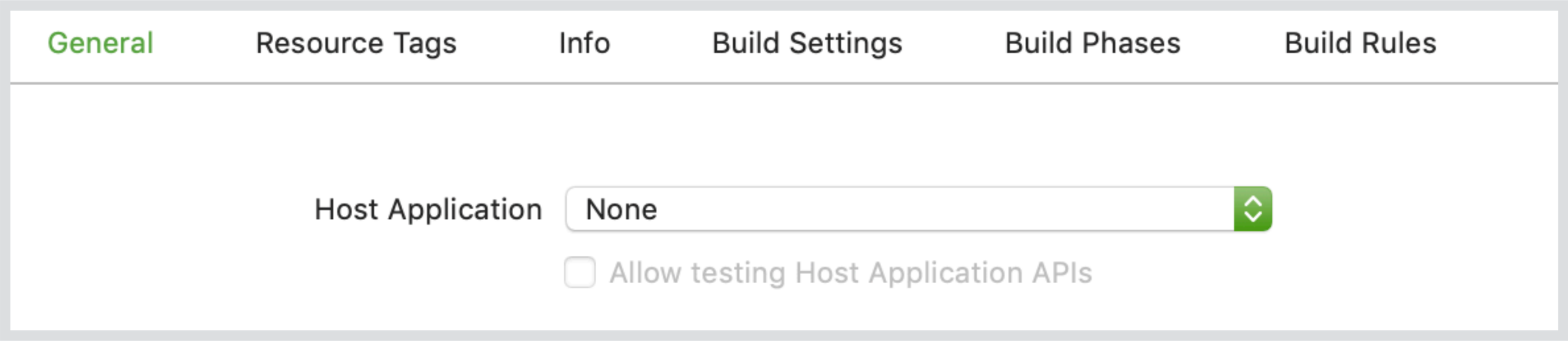
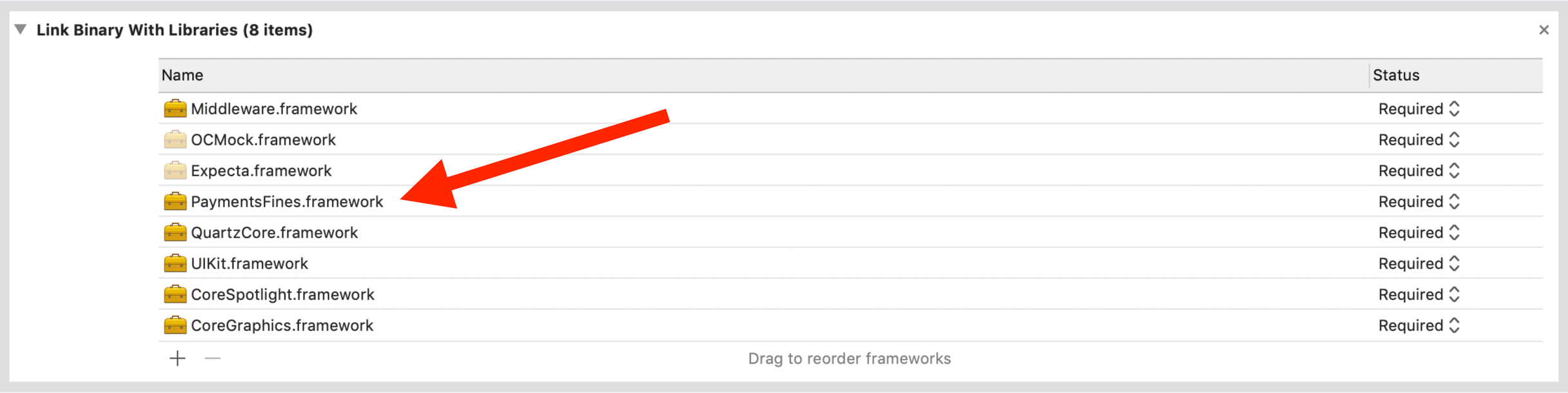
Tests

Host application



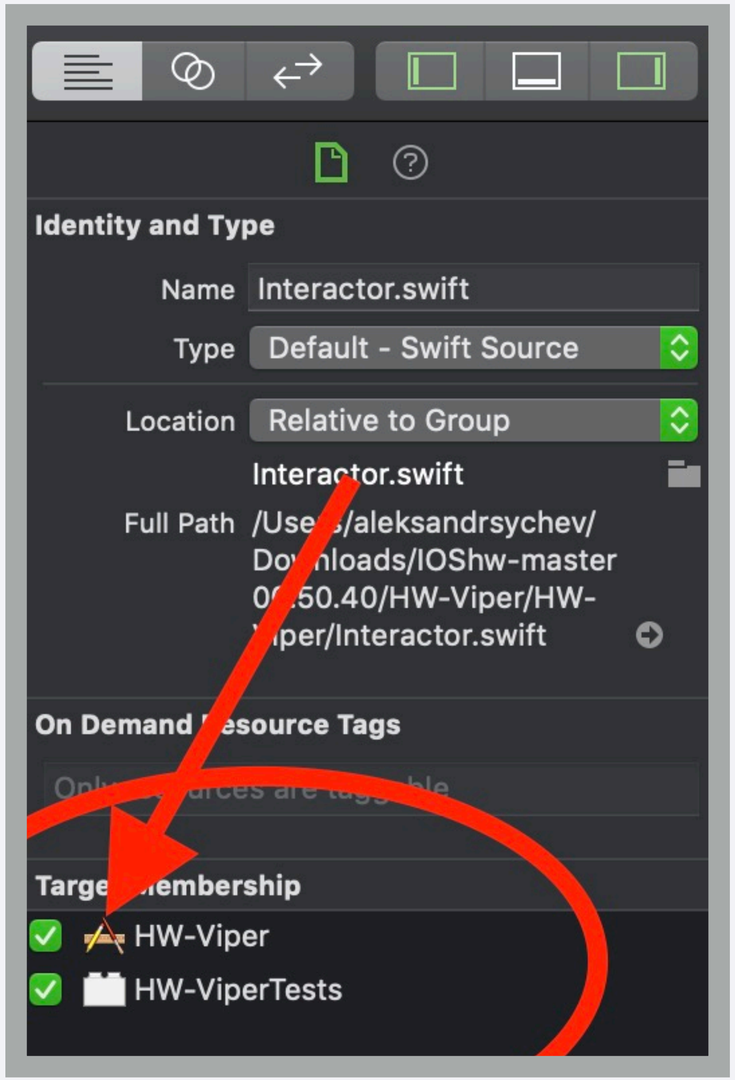
Host application



Сбербанк Онлайн

Devteam

Host application



Сбербанк Онлайн

Devteam

Examples

Characteristics of Testable Code

Control over inputs

Visibility into outputs

No hidden state

https://developer.apple.com/videos/play/wwdc2017/414/

Testable vs Untestable

Dependency Injection/Transparency	Implicit Dependencies
Interfaces/Factories/IoC	New / alloc-init
Law Of Demeter	Service Locators
Local	Static
Single Responsibility/Layered Architecture	Mixed Concerns
Composition	Deep Inheritance
Simple Initializers	Complex Initializers
Idempotence	Singletons, Random

Сбербанк Онлайн

Devteam

Assertions

Assertions

```
public func XCTFail(_ message: String = default,
                    file: StaticString = #file,
                    line: UInt = #line)
```

```
func testExample() {
        // arrange

        // act

        // assert
        XCTFail("message")
}
```

Сбербанк Онлайн

Devteam

```swift
func testExample() {
    // arrange
    let pair = (20, 20)

    // act

    // assert
    XCTAssertEqual(pair, (20, 21))
}
```

Сбербанк Онлайн

Devteam

```
func testExample() {
    // arrange
    let pair = (20, 20)

    // act

    // assert
    XCTAssertEqual(pair, (20, 21))
}
```

Сбербанк Онлайн

Devteam

```
func testExample() {
    // arrange
    let pair = (20, 20)

    // act

    // assert
    XCTAssert(pair == (20, 21))
}
```

Сбербанк Онлайн

Devteam

```
    func testExample() {
        // arrange
        let pair = (20, 20)

        // act

        // assert
        XCTAssert(pair == (20, 21))      XCTAssertTrue failed
    }
```

```
    func testExample() {
        // arrange
        let pair = (20, 20)

        // act

        // assert
        XCTAssert(pair == (20, 21), "was \(pair)")
    }
```

XCTAssertTrue failed - was (20, 20)

Сбербанк Онлайн

Devteam

Write a custom test assertion

```
func assertIntPairsEqual(actual: (_: Int, _: Int),
                         expected: (_: Int, _: Int),
                         file: StaticString = #file,
                         line: UInt = #line) {
    if actual != expected {
        XCTFail("Expected \(expected) but was \(actual)",
                file: file, line: line)
    }
}
```

```
func assertIntPairsEqual(actual: (_: Int, _: Int),
                         expected: (_: Int, _: Int),
                         file: StaticString = #file,
                         line: UInt = #line) {
    if actual != expected {
        XCTFail("Expected \(expected) but was \(actual)",
                file: file, line: line)
    }
}
```

Сбербанк Онлайн

Devteam

```
func assertIntPairsEqual(actual: (_: Int, _: Int),
                         expected: (_: Int, _: Int),
                         file: StaticString = #file,
                         line: UInt = #line) {
    if actual != expected {
        XCTFail("Expected \(expected) but was \(actual)",
                file: file, line: line)
    }
}
```

Сбербанк Онлайн

Devteam

```
    func testExample() {
        // arrange
        let pair = (20, 20)

        // act

        // assert
        assertIntPairsEqual(actual: pair, expected: (20, 21))
    }

```

failed - Expected (20, 21) but was (20, 20)

Сбербанк Онлайн

Devteam

```
func assertPairsEqual<T: Equatable, U: Equatable>(actual: (_: T, _: U),
                                                  expected: (_: T, _: U),
                                                  file: StaticString = #file,
                                                  line: UInt = #line)
```

Mocks.
Protocols and Extensions

```
struct ToDoService {

    init(session: URLSession) {
        // …
    }
}

let service = ToDoService(session: URLSession.shared)
```

```
struct ToDoService {

    init(session: URLSession) {
        // …
    }
}

let service = ToDoService(session: URLSession.shared)
```

```
protocol URLSessionProtocol {}
extension URLSession: URLSessionProtocol {}

struct ToDoService {

    init(session: URLSessionProtocol) {
        // …
    }
}
```

```
protocol URLSessionProtocol {}
extension URLSession: URLSessionProtocol {}

struct ToDoService {

    init(session: URLSessionProtocol) {
        // …
    }
}
```

Сбербанк Онлайн

Devteam

```swift
protocol URLSessionProtocol {}
extension URLSession: URLSessionProtocol {}

struct ToDoService {

    init(session: URLSessionProtocol) {
        // …
    }
}
```

Сбербанк Онлайн

Devteam

```
protocol URLSessionProtocol {}
extension URLSession: URLSessionProtocol {}

struct ToDoService {

    init(session: URLSessionProtocol) {
        // …
    }
}
```

```
let service = ToDoService(session: URLSession.shared)
```

```
class MockURLSession: URLSessionProtocol {}
```

```
protocol URLSessionProtocol {

    func dataTask(request: URLRequest,
                  completion:(Data?, Response?, Error?) -> Void) -> DataTask

}
```

```
class MockURLSession: URLSessionProtocol {

    func dataTask(with …) -> URLSessionDataTask {
        return URLSessionDataTask()
    }
}
```

Mocks

1. Число вызовов

2. Переданные аргументы

```
class MockURLSession: URLSessionProtocol {
    var dataTaskCallCount = 0

    func dataTask(with …) -> URLSessionDataTask {
        dataTaskCallCount += 1
        return URLSessionDataTask()
    }
}
```

```
func testExample() {
    // arrange
    ...

    // act
    ...

    // assert
    XCTAssertEqual(mockURLSession.dataTaskCallCount, 1)
}
```

```
class MockURLSession: URLSessionProtocol {
    var dataTaskLastURL: URL?

    func dataTask(with …) -> URLSessionDataTask {
        dataTaskLastURL = url
        return URLSessionDataTask()
    }
}
```

```
func testExample() {
    // arrange
    ...

    // act
    ...

    // assert
    XCTAssertEqual(mockURLSession.dataTaskLastURL?.host,
                   "http://expected.com")
}
```

Сбербанк Онлайн

Devteam

Demo

Mocks

1. Используйте вспомогательные методы для одной проверки
2. Накапливайте аргументы в коллекциях

Зачем нужны тесты

Виды тестов

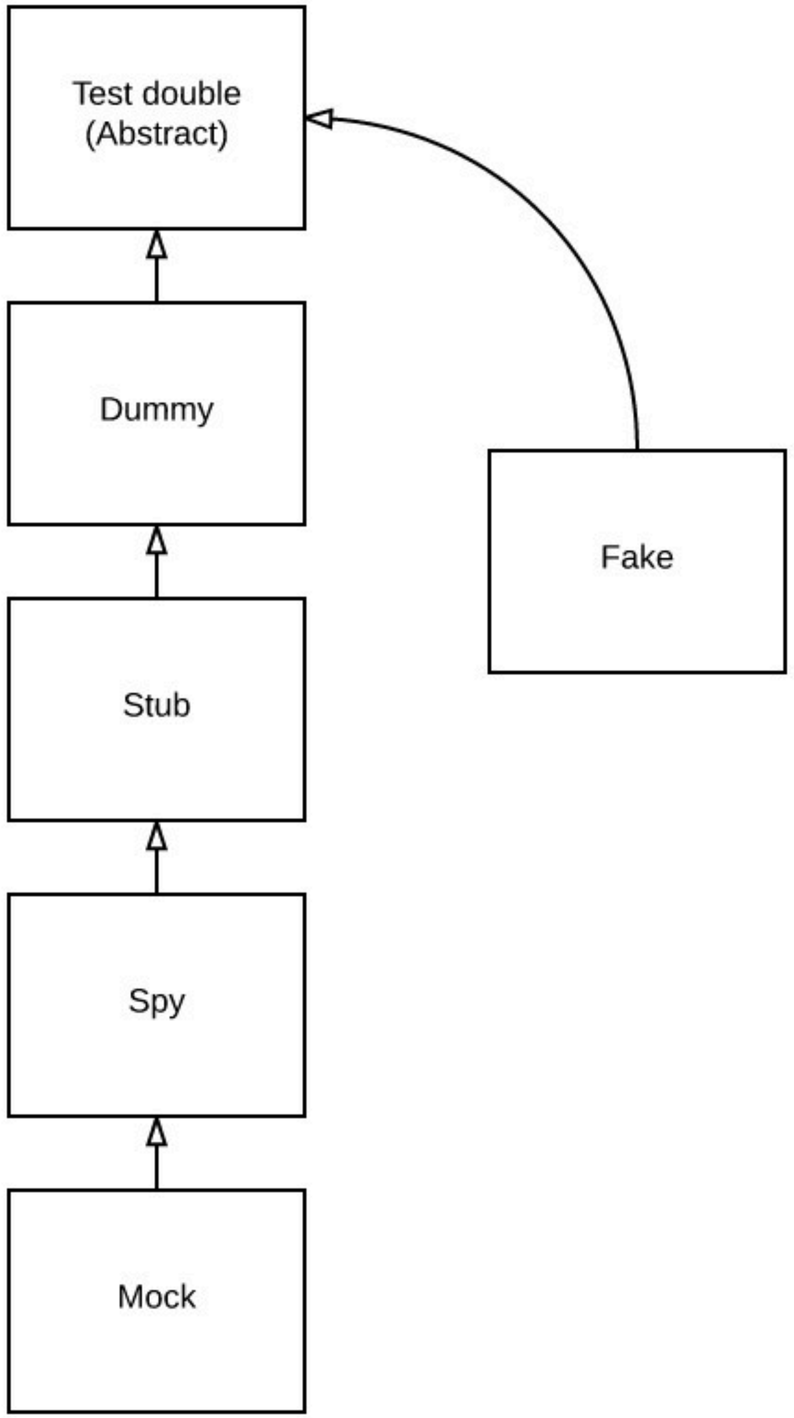
“Чистые” тесты

Test doubles

Tips and tricks

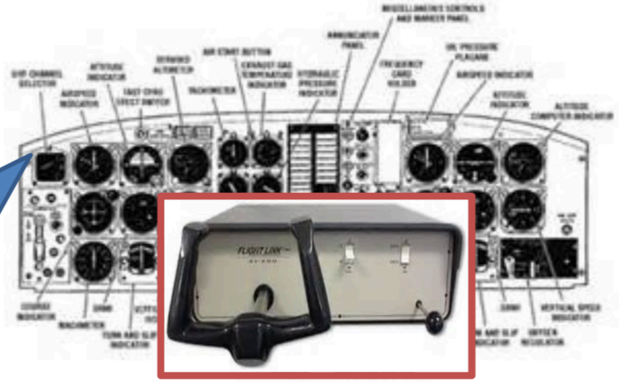
Сбербанк Онлайн

Devteam



Dummy

Stub/Spy



Fake

Mock

Devteam

```
protocol UserServiceProtocol {

    func save(username: String, password: String) -> Bool
}
```

Dummy

```
class UserServiceDummy: UserServiceProtocol {

    func save(username: String, password: String) -> Bool {
        fatalError()
    }
}
```

Stub

```
class UserServiceStub: UserServiceProtocol {

    func save(username: String, password: String) -> Bool {
        return true
    }
}
```

Spy

```
class UserServiceSpy: UserServiceProtocol {

    var saveCallsCount = 0
    func save(username: String, password: String) -> Bool {
        saveCallsCount += 1
        return true
    }
}
```

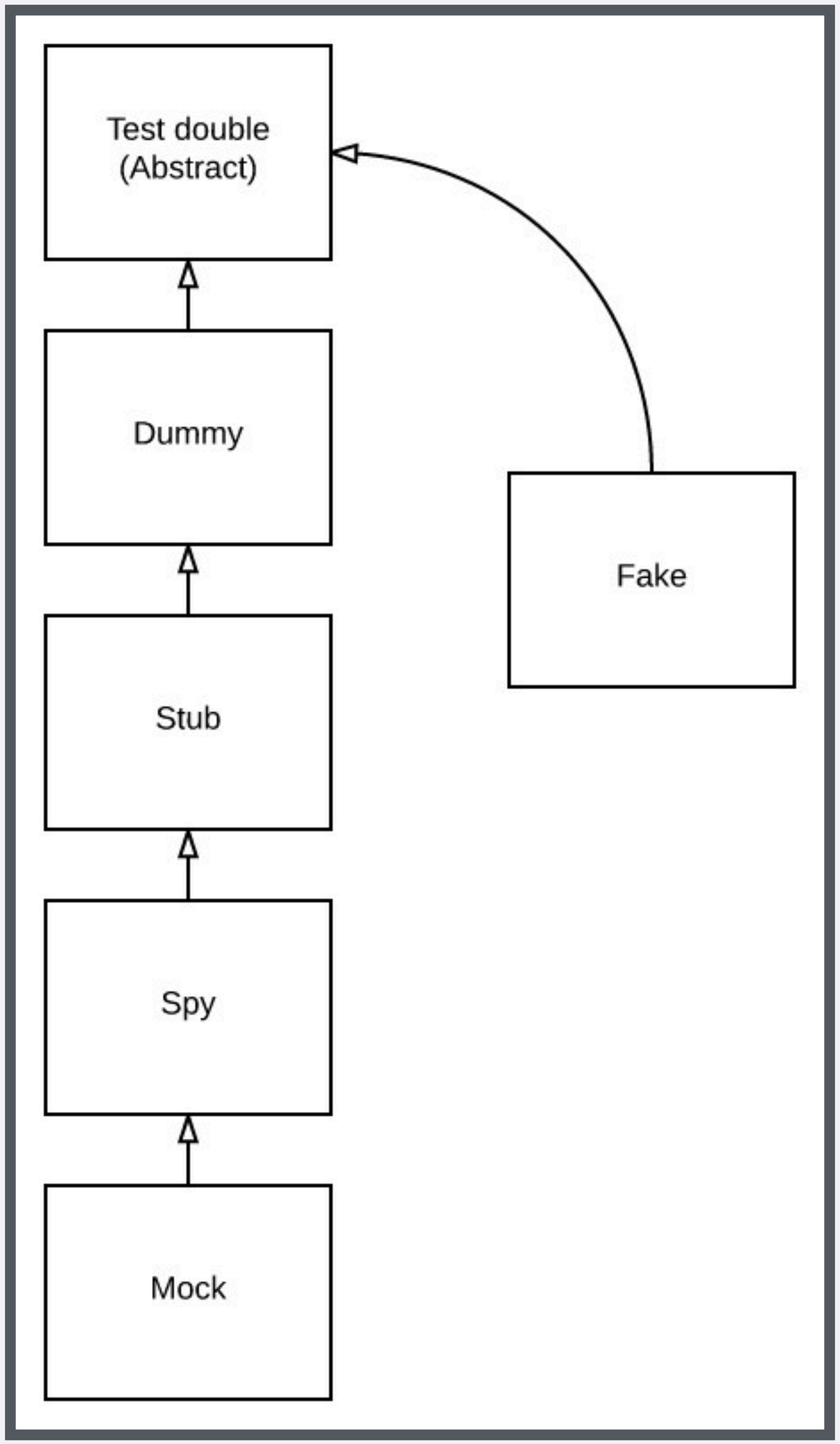
True mock

```
class UserServiceMock: UserServiceProtocol {

    var saveCallsCount = 0
    func save(username: String, password: String) -> Bool {
        saveCallsCount += 1
        return true
    }

    func verifySave(expectedCallCount: Int) {
        XCTAssertEqual(expectedCallCount, saveCallsCount)
    }
}
```

Fake

```
class UserServiceFake: UserServiceProtocol {

    func save(username: String, password: String) -> Bool {
        return username == "СберКот"
    }
}
```

Test double



Сбербанк Онлайн

Devteam

РАБОТА В ГРУППАХ

How to Create a Mock?

InstantMock enables to create a single mock that can be used in many tests, for a protocol or a class.

For a Protocol

The easiest way to create a mock for a protocol is to inherit from the `Mock` class.

```
// MARK: Protocol to be mocked
protocol Foo {
    func bar(arg1: String, arg2: Int) -> Bool
}

// MARK: Mock class inherits from `Mock` and adopts the `Foo` protocol
class FooMock: Mock, Foo {

    // implement `bar` of the `Foo` protocol
    func bar(arg1: String, arg2: Int) -> Bool {
        return super.call(arg1, arg2)! // provide values to parent class
    }

}
```

Сбербанк Онлайн

https://github.com/pirishd/InstantMock

Devteam

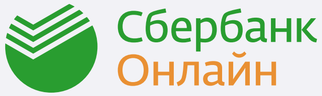
Зачем нужны тесты

Виды тестов

“Чистые” тесты

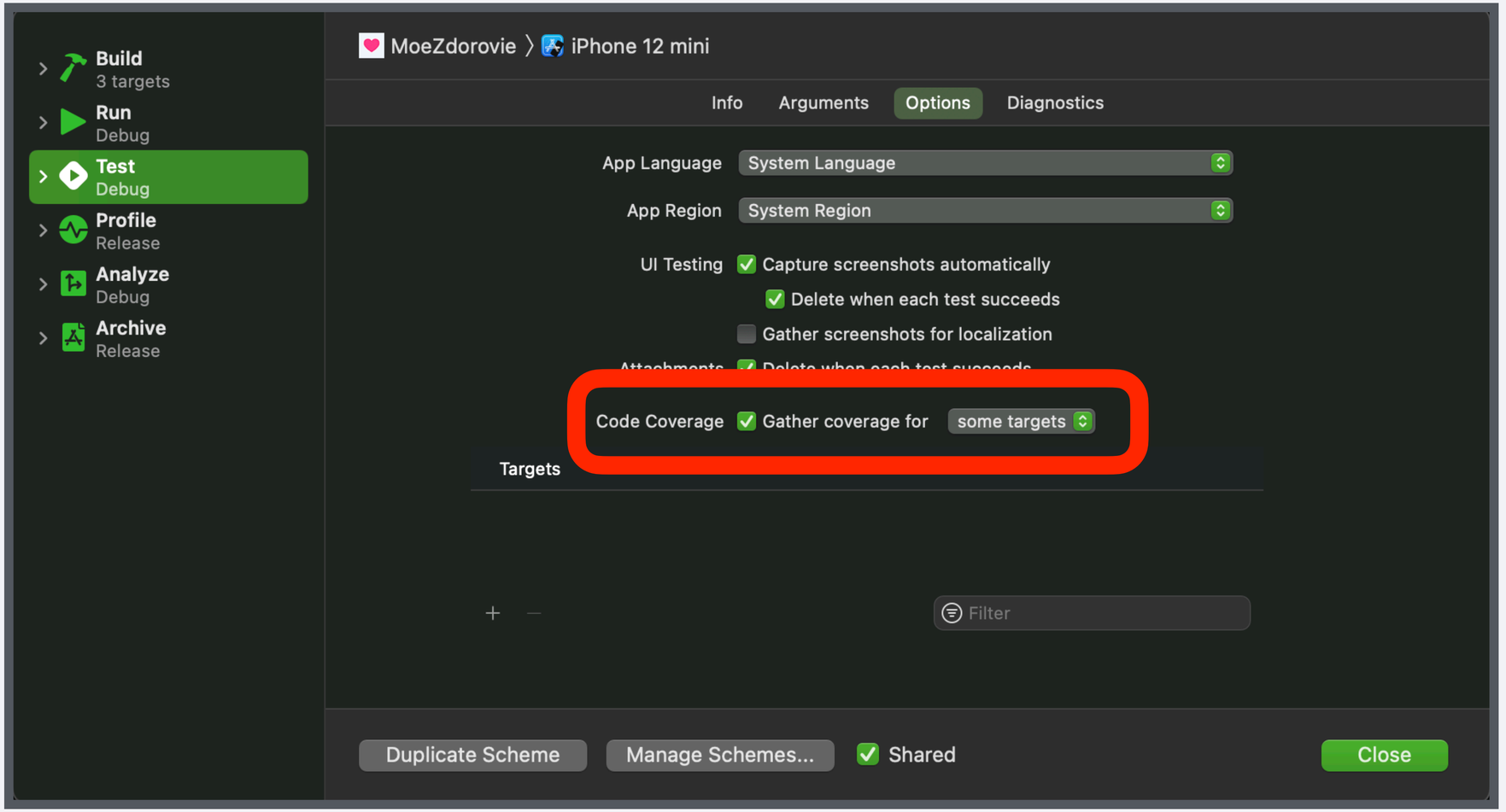
Test doubles

Tips and tricks



Devteam

Code coverage

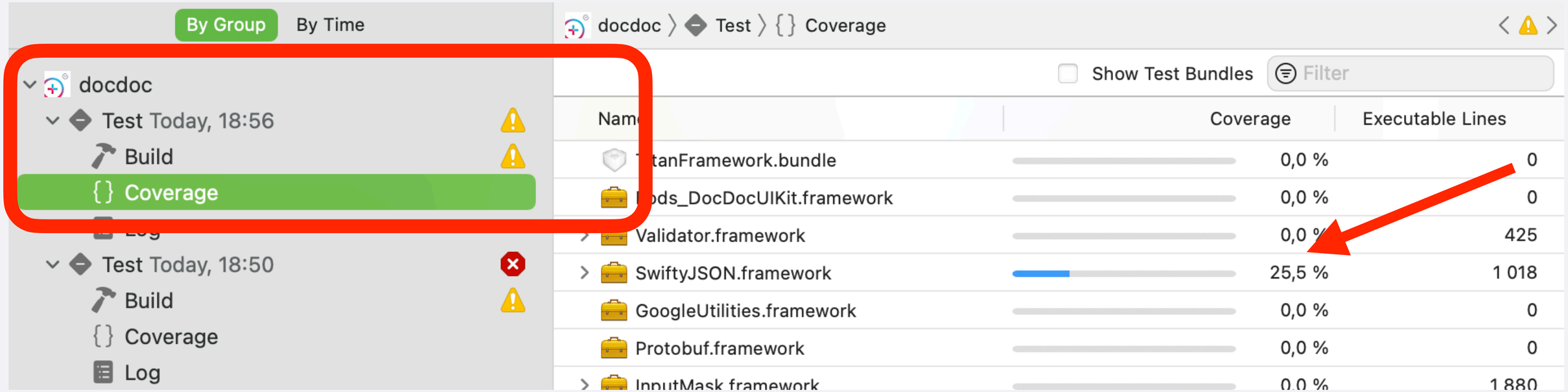


- **Function** coverage
- **Statement** coverage
- **Edge** coverage
- **Branch** coverage
- **Condition** coverage
- **input/output** coverage

Code coverage

- What code is actually being run when you run your tests?
- How many tests are enough tests?
- What parts of your code are not being tested?

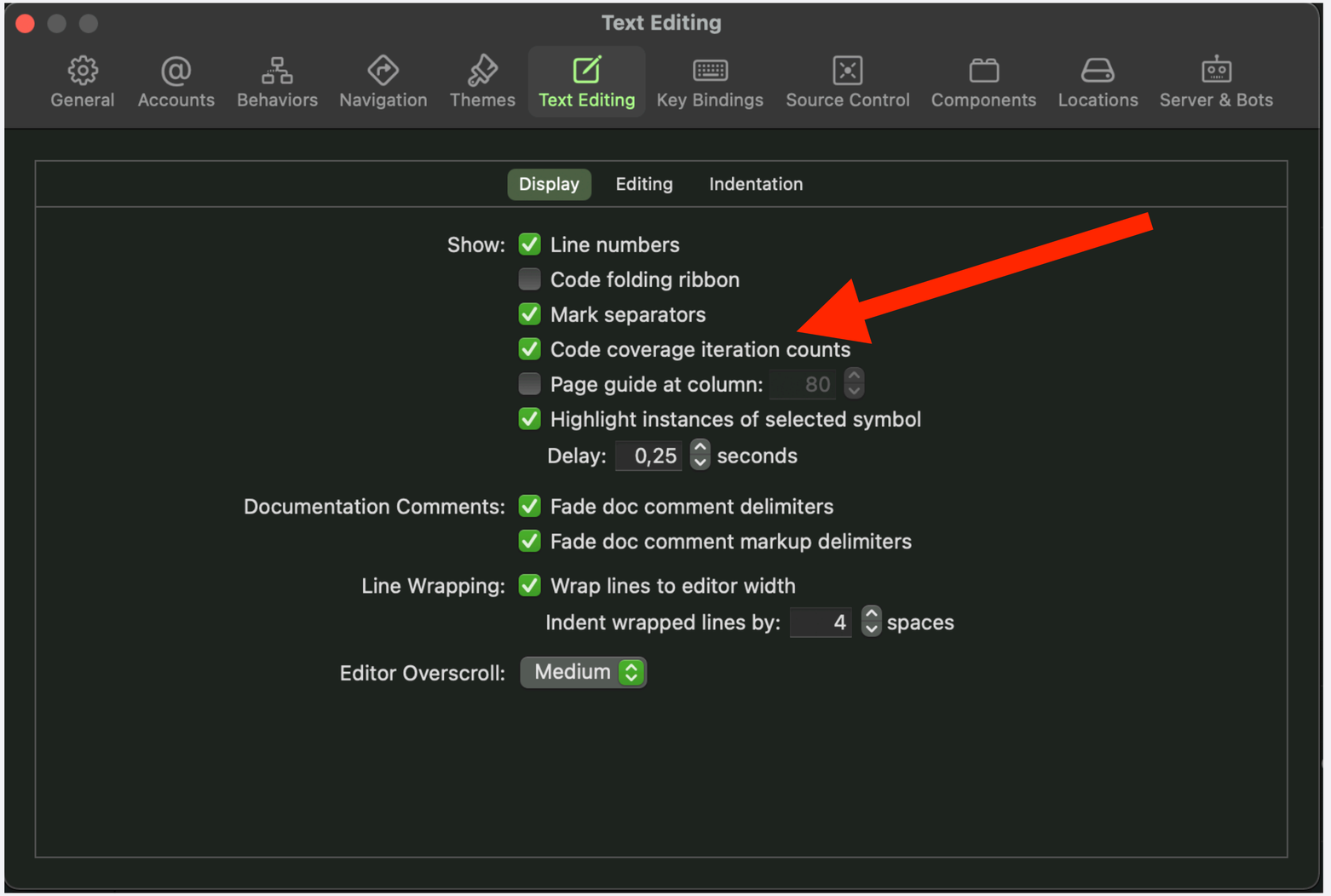
Code coverage



Сбербанк Онлайн

Devteam

Code coverage



Сбербанк Онлайн

Devteam

Code coverage

```
public class ActionListView<T: ActionListItem>: CustomView, InteractiveDismissableView,
    UITableViewDataSource, UITableViewDelegate {

    public var title: String? {
        didSet {
            configureTitleLabel(with: title)
        }
    }

    public var items: [T] = [] {
        didSet {
            tableView.reloadData()
        }
```

Code coverage

100% 

DEMO

Code coverage. Demo

```swift
/// A simple struct containing a list of users.
struct UsersViewModel {

    let users: [String]

    var hasUsers: Bool {
        return !users.isEmpty
    }
}

/// A test case to validate our logic inside the `UsersViewModel`.
final class UsersViewModelTests: XCTestCase {

    func testThatViewModelKnowsThatItHasUsers() {
        // arrange
        let viewModel = UsersViewModel(users: ["Antoine", "Jaap", "Lady"])

        // act
        let result = viewModel.hasUsers

        // assert
        XCTAssertTrue(result)
    }
}
```

Сбербанк Онлайн

Devteam

NAME	DESCRIPTION	IDEAL VALUE	COMMON VALUE	PROBLEM VALUE
PDWT	Процент разработчиков, пишущих тесты	100 %	20%-70%	Любое меньше 100%
PBCNT	Процент багов, приводящих к созданию новых тестов	100 %	0%-5%	Любое меньше 100%
PTVB	Процент тестов, проверящих поведение	100 %	10 %	Любое меньше 100%
PTD	Процент детерминированных тестов	100 %	50%-80%	Любое меньше 100%

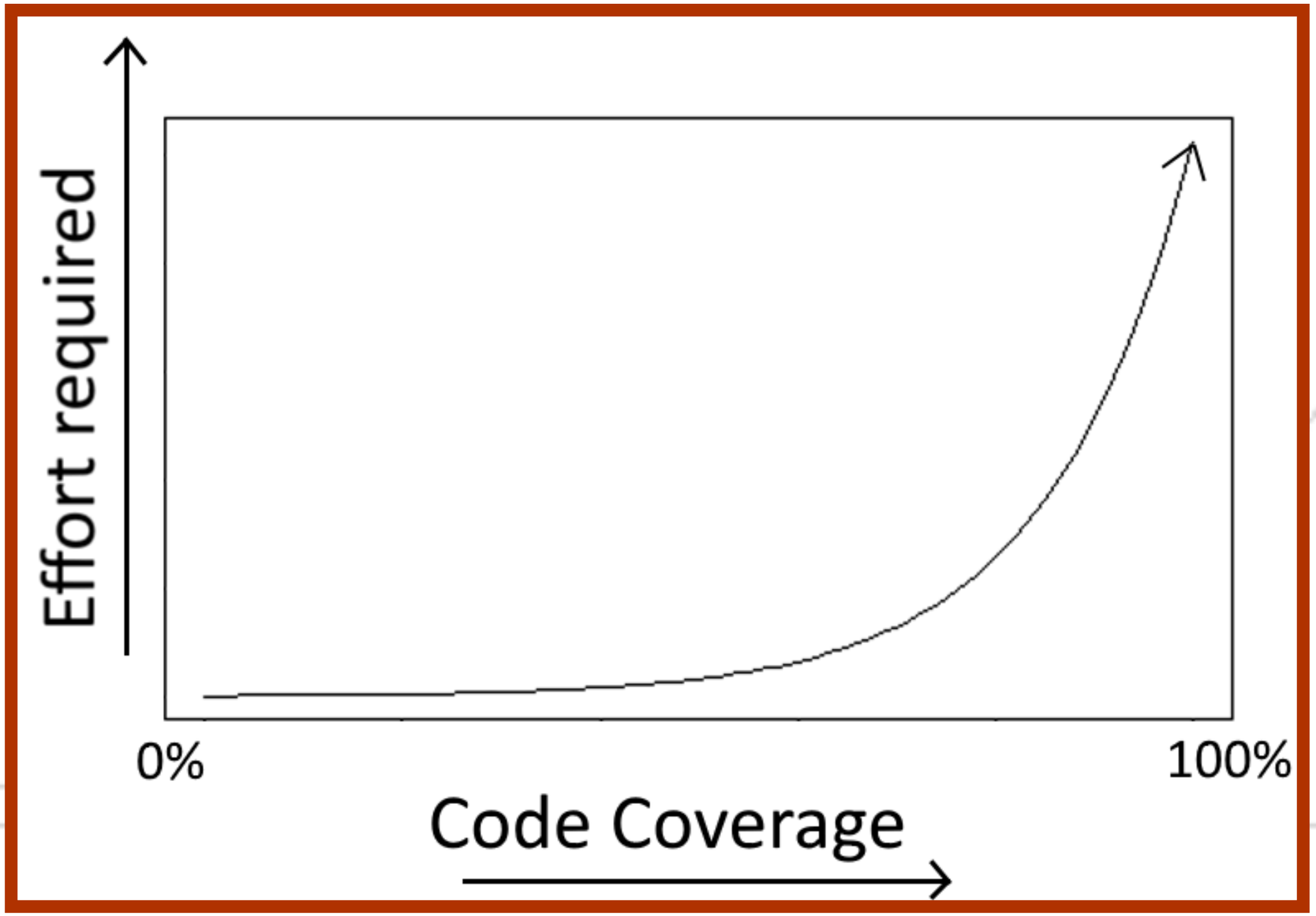
NAME	DESCRIPTION	IDEAL VALUE	COMMON VALUE	PROBLEM VALUE
PDWT	Процент разработчиков, пишущих тесты	100 %	20%-70%	Любое меньше 100%
PBCNT	Процент багов, приводящих к созданию новых тестов	100 %	0%-5%	Любое меньше 100%
PTVB	Процент тестов, проверящих поведение	100 %	10 %	Любое меньше 100%
PTD	Процент детерминированных тестов	100 %	50%-80%	Любое меньше 100%

NAME	DESCRIPTION	IDEAL VALUE	COMMON VALUE	PROBLEM VALUE
PDWT	Процент разработчиков, пишущих тесты	100 %	20%-70%	Любое меньше 100%
PBCNT	Процент багов, приводящих к созданию новых тестов	100 %	0%-5%	Любое меньше 100%
PTVB	Процент тестов, проверящих поведение	100 %	10 %	Любое меньше 100%
PTD	Процент детерминированных тестов	100 %	50%-80%	Любое меньше 100%

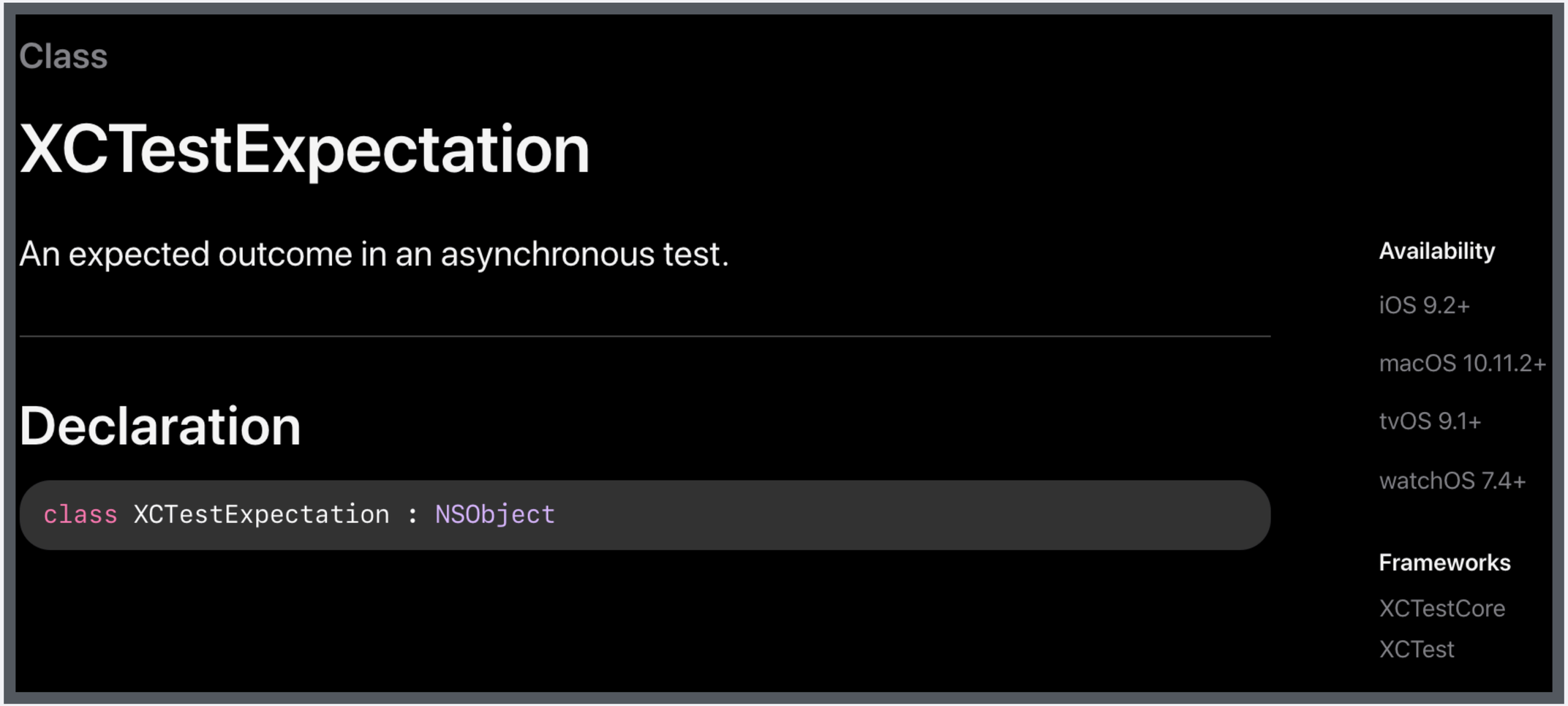
NAME	DESCRIPTION	IDEAL VALUE	COMMON VALUE	PROBLEM VALUE
PDWT	Процент разработчиков, пишущих тесты	100 %	20%-70%	Любое меньше 100%
PBCNT	Процент багов, приводящих к созданию новых тестов	100 %	0%-5%	Любое меньше 100%
PTVB	Процент тестов, проверящих поведение	100 %	10 %	Любое меньше 100%
PTD	Процент детерминированных тестов	100 %	50%-80%	Любое меньше 100%

NAME	DESCRIPTION	IDEAL VALUE	COMMON VALUE	PROBLEM VALUE
PDWT	Процент разработчиков, пишущих тесты	100 %	20%-70%	Любое меньше 100%
PBCNT	Процент багов, приводящих к созданию новых тестов	100 %	0%-5%	Любое меньше 100%
PTVB	Процент тестов, проверящих поведение	100 %	10 %	Любое меньше 100%
PTD	Процент детерминированных тестов	100 %	50%-80%	Любое меньше 100%

20%


Effort required
0%
100%
Code Coverage

Unit testing asynchronous Swift code



Сбербанк Онлайн

Devteam

Unit testing asynchronous. Expectations

- Create an instance of **XCTestExpectation**
- Fulfill the expectation when async operation has finished
- Wait for the expectation to be **fulfilled**
- Assert the expected result

Unit testing asynchronous. Expectations

```
protocol HTTPClient {
    func execute(request: URLRequest, completion: @escaping (Result<Data, Error>) -> Void)
}

struct MusicService {
    let httpClient: HTTPClient

    func search(_ term: String, completion: @escaping (Result<[Track], Error>) -> Void) {
        httpClient.execute(request: .search(term: term)) { result in
            completion(self.parse(result))
        }
    }

    private func parse(_ result: Result<Data, Error>) -> Result<[Track], Error> { ... }
}
```

Unit testing asynchronous. Expectations

```
func testThatMusicServiceReturnsAnyExistedOptionOnResearch() {
    // 1.
    let didReceiveResponse = expectation(description: #function)

    // 2.
    let sut = MusicService(httpClient: RealHTTPClient())

    // 3.
    var result: Result<[Track], Error>?

    // 4.
    sut.search("ANYTHING") {
        result = $0
        didReceiveResponse.fulfill()
    }

    // 5.
    wait(for: [didReceiveResponse], timeout: 5)

    // 6.
    XCTAssertNotNil(result?.value)
}
```

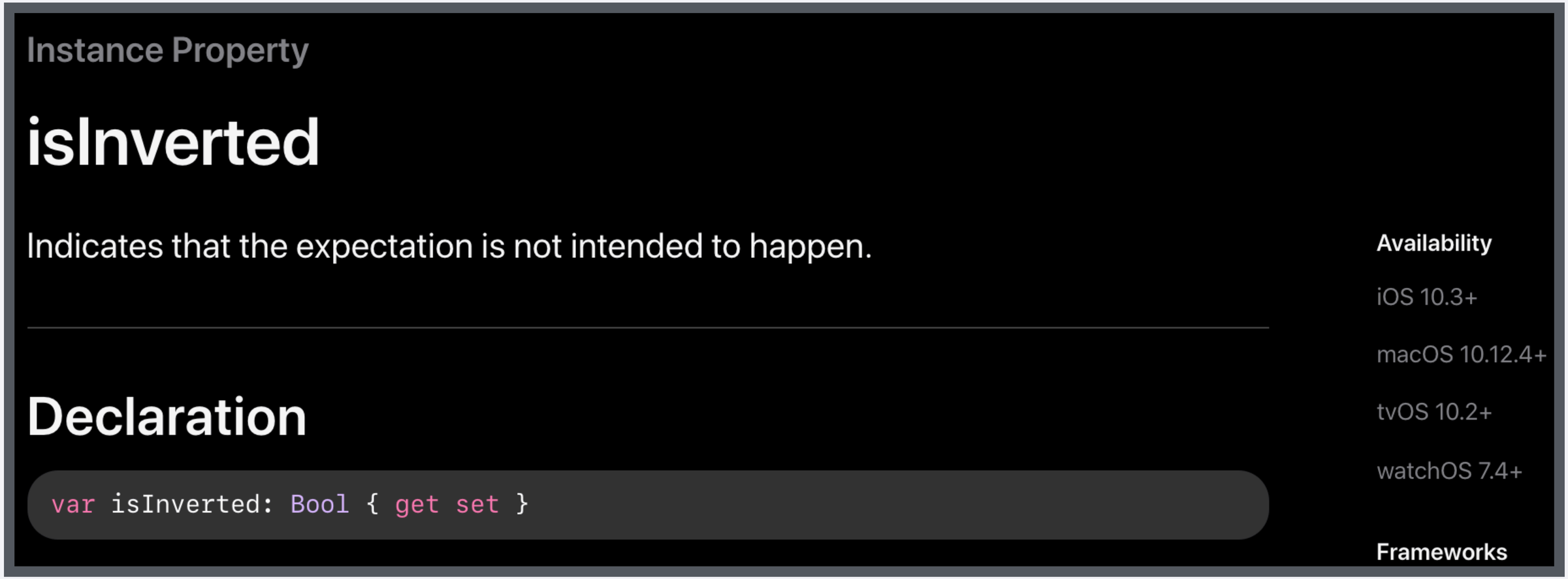
Сбербанк Онлайн

Devteam

Unit testing asynchronous. Expectations

```
func testExpectationFulfillmentCount() {

    let exp = expectation(description: #function)
    exp.expectedFulfillmentCount = 3
    exp.assertForOverFulfill = true


    ...
    sut.doSomethingThreeTimes()

    wait(for: [exp], timeout: 1)
}
```

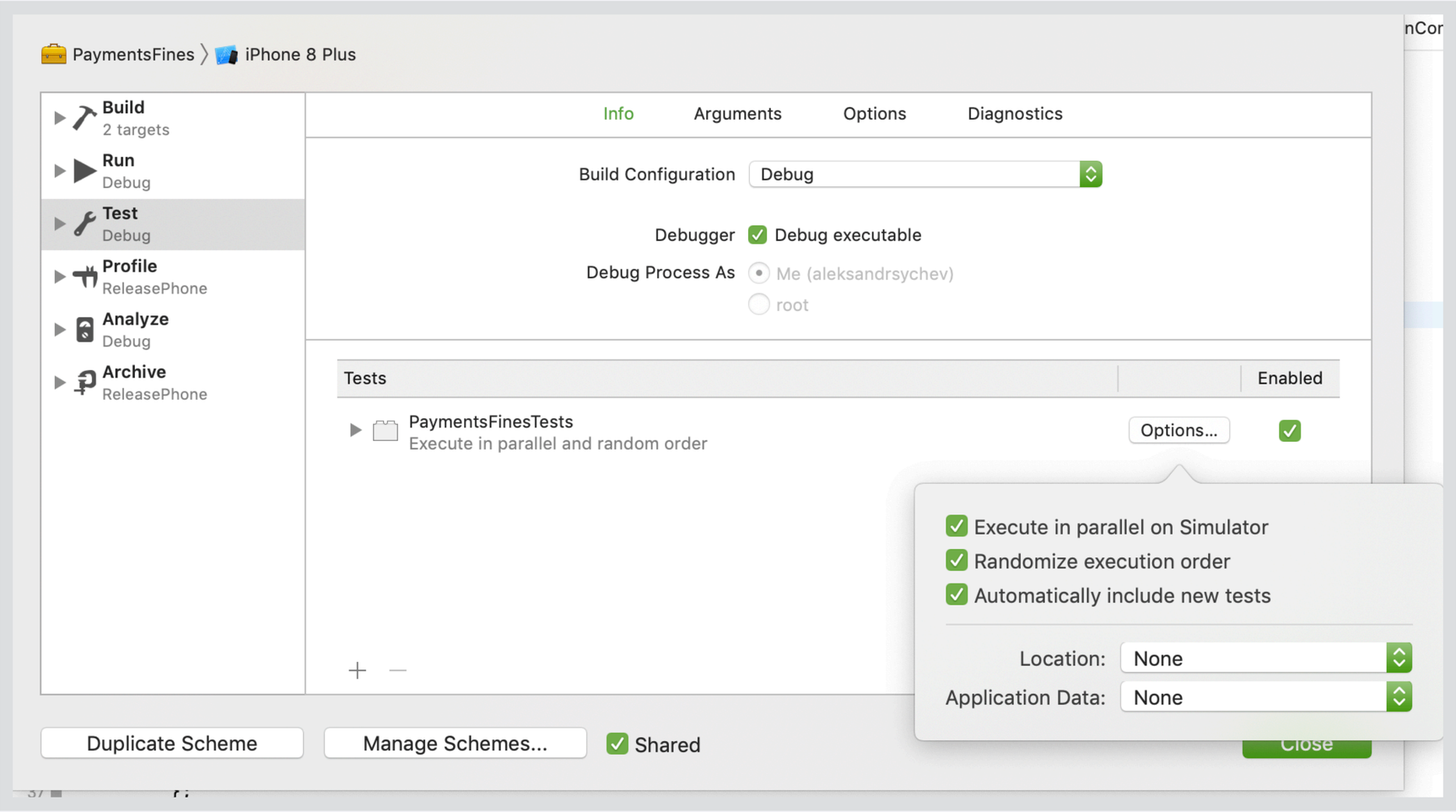
Unit testing asynchronous. Expectations



Сбербанк Онлайн

Devteam

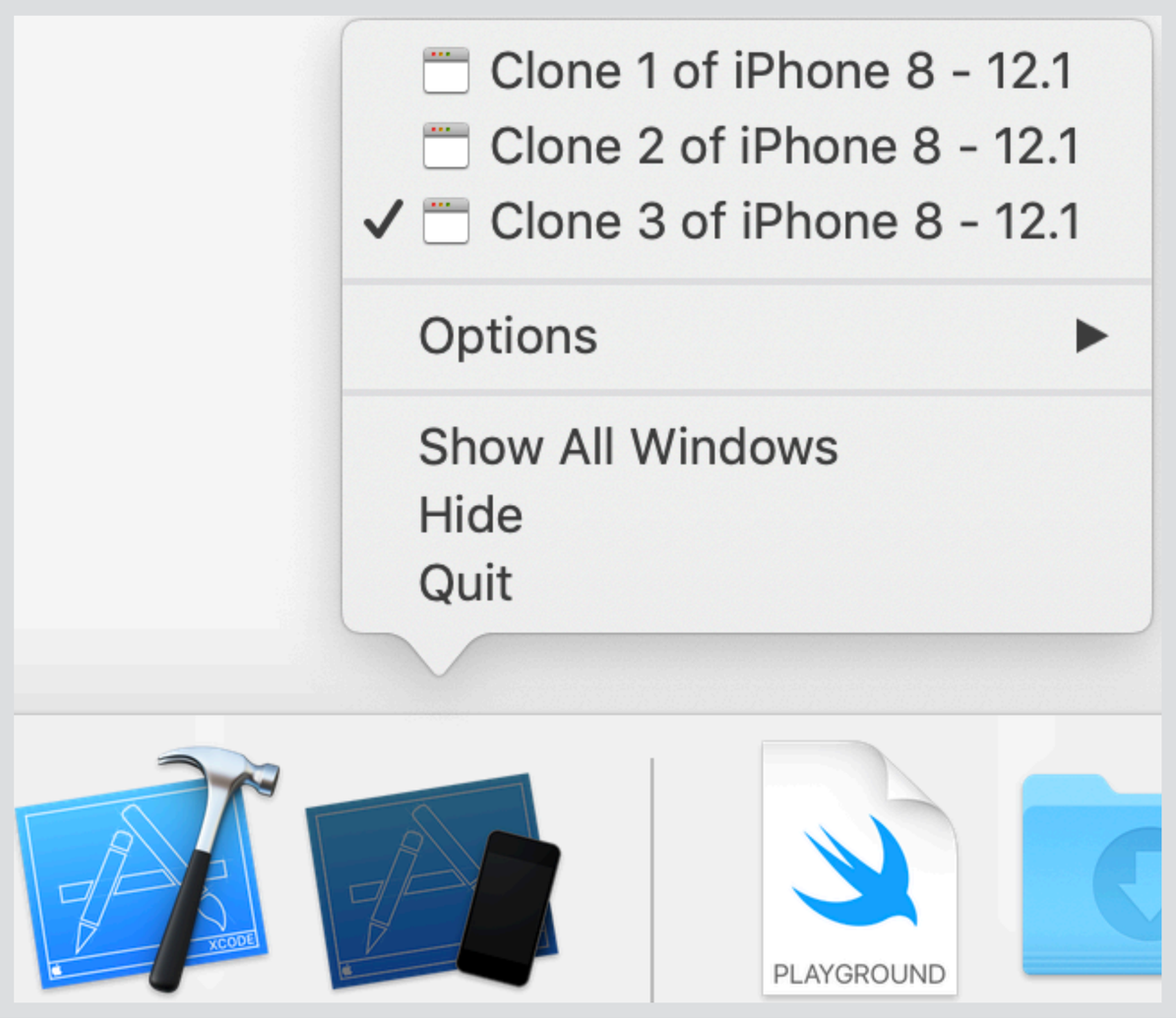
Random and parallel tests



Сбербанк Онлайн

Devteam

Random and parallel tests



Сбербанк Онлайн

Devteam

Random and parallel tests

```
$ xcodebuild test -scheme MyApp \
  -destination "plaform=iOS,name=iPhone 8" \
  -destination "platform=iOS,name=iPhone 8 Plus"
```

Rules

- Avoid coding a tautology
- Don’t Test Private Methods
- Don’t Stub Private Methods
- Don’t Stub External Libraries
- Don't mock everything
- Don’t Test Constructors & Don't mock value objects

Avoid coding a tautology

```swift
struct Summator<T:AdditiveArithmetic> {

    func sum(_ a:T, _ b:T) -> T {
        return a + b
    }
}

final class SummatorTests: XCTestCase {

    func testThatSummatorWorksCorrectly() {
        // arrange
        let summator = Summator<Int>()
        let a:Int = 5
        let b:Int = 3

        // act
        let result = summator.sum(a, b)

        // assert
        XCTAssertEqual(a + b, result)
    }
}
```

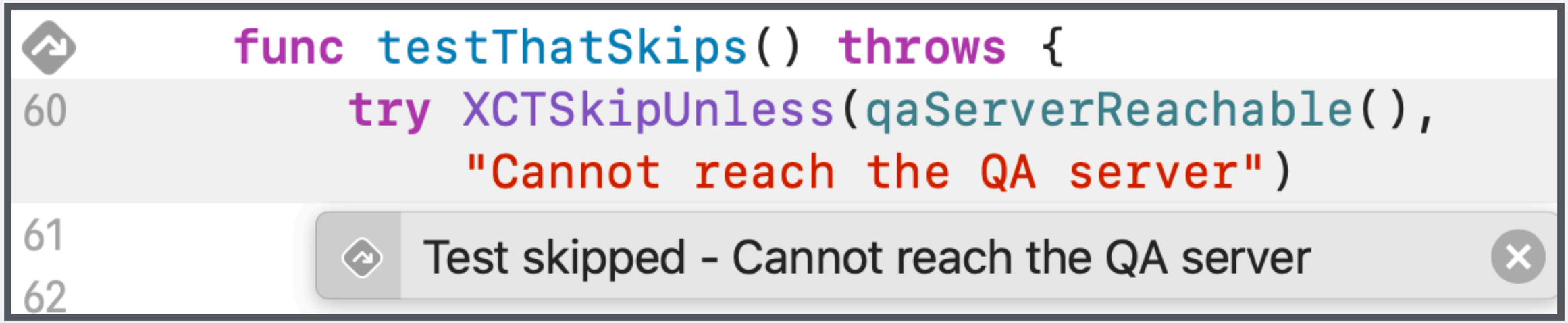
Rules

- Avoid coding a tautology
- Don’t Test Private Methods
- Don’t Stub Private Methods
- Don’t Stub External Libraries
- Don't mock everything
- Don’t Test Constructors & Don't mock value objects

Skipping tests

```
func testThatSkips() throws {
  try XCTSkipUnless(qaServerReachable(), "QA server not reachable")
  // Retrieve test data
  // Perform test
}
```

Skipping tests



Skipping tests

Function

XCTSkipIf(_:_:file:line:)

Skips remaining tests in a test method if the specified condition is met.

Declaration

```
func XCTSkipIf(_ expression: @autoclosure () throws -> Bool, _ message:
    @autoclosure () -> String? = nil, file: StaticString = #filePath, line: UInt
    = #line) throws
```

Сбербанк Онлайн

Devteam

Почитать

1. https://qualitycoding.org/
2. https://hackernoon.com/test-f-i-r-s-t-65e42f3adc17
3. https://github.com/ghsukumar/SFDC_Best_Practices/wiki/F.I.R.S.T-Principles-of-Unit-Testing
4. https://github.com/Quick/Quick/blob/d30f9e93402b6fcc7013c86afeb77da4c38e9f27/Documentation/en-us/ArrangeActAssert.md
5. https://developer.apple.com/videos/play/wwdc2017/414/
6. https://developer.apple.com/videos/play/wwdc2018/403/
7. https://www.objc.io/issues/15-testing/

Почитать

8. https://blog.cleancoder.com/uncle-bob/2014/05/14/TheLittleMocker.html

9. https://marcosantadev.com/test-doubles-swift/

10. https://www.swiftbysundell.com/articles/simple-swift-dependency-injection-with-functions/

11. https://www.swiftbysundell.com/articles/unit-testing-asynchronous-swift-code/

12. https://useyourloaf.com/blog/xcode-10-random-and-parallel-tests/

13. https://www.fbidb.io/docs/test_execution

14. http://randycoulman.com/blog/2016/12/20/tautological-tests/

Домашнее задание

Добавить тестовый таргет (Unit test bundle) и написать модульные тесты в выпускном проекте. Обеспечить покрытие **10%**

СРОК - ЗАЩИТА ПРОЕКТА