**Лабораторная работа №10**

*«Тестирование приложений для ios»*

**Цель работы:** научиться проводить тестирование мобильных приложений под IOS методом белого ящика в среде разработки XCode и методом черного ящика с использованием симуляторов или реальных устройств. Научиться снимать логи с IOS устройств.



Рисунок 1 – Запуск WB Mac OS



Рисунок 2 – Открытие Xcode

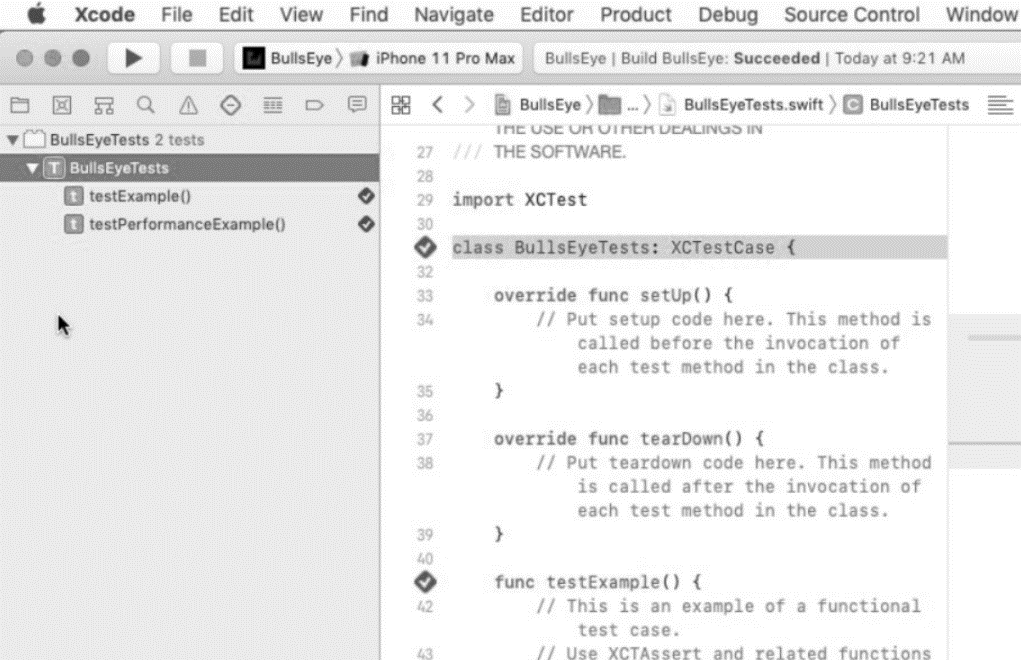


Рисунок 3 – BullsEyeTests выполнен успешно тоьб

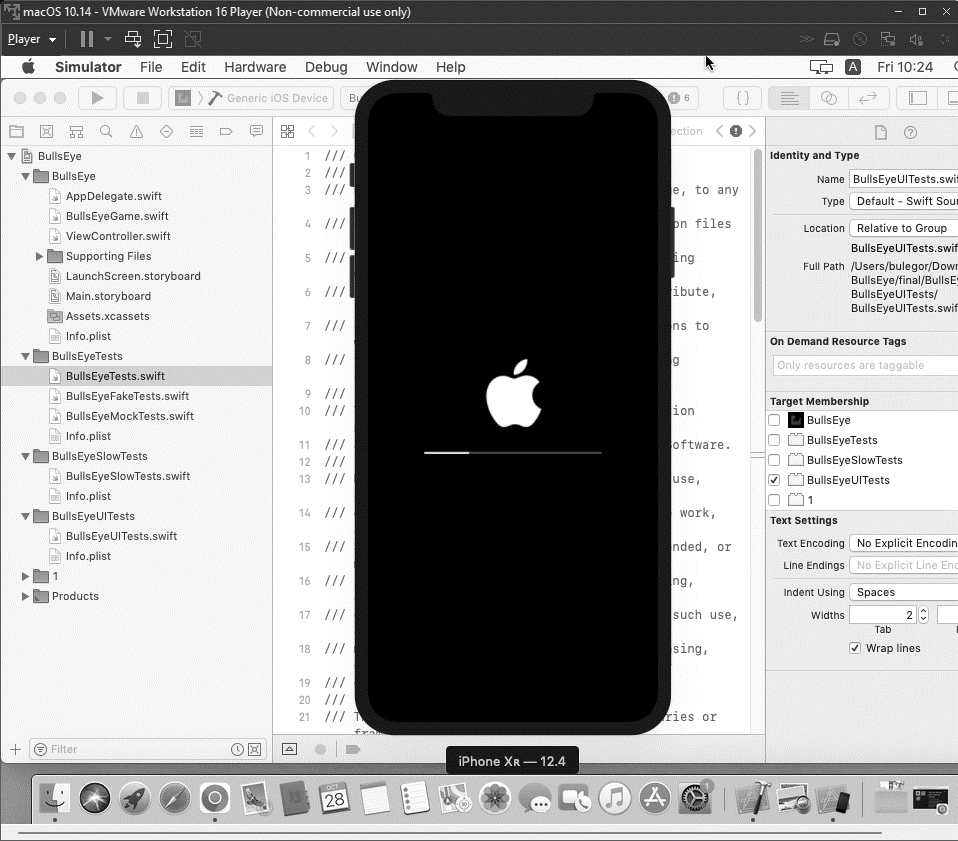


Рисунок 4 – Создание эмулятора

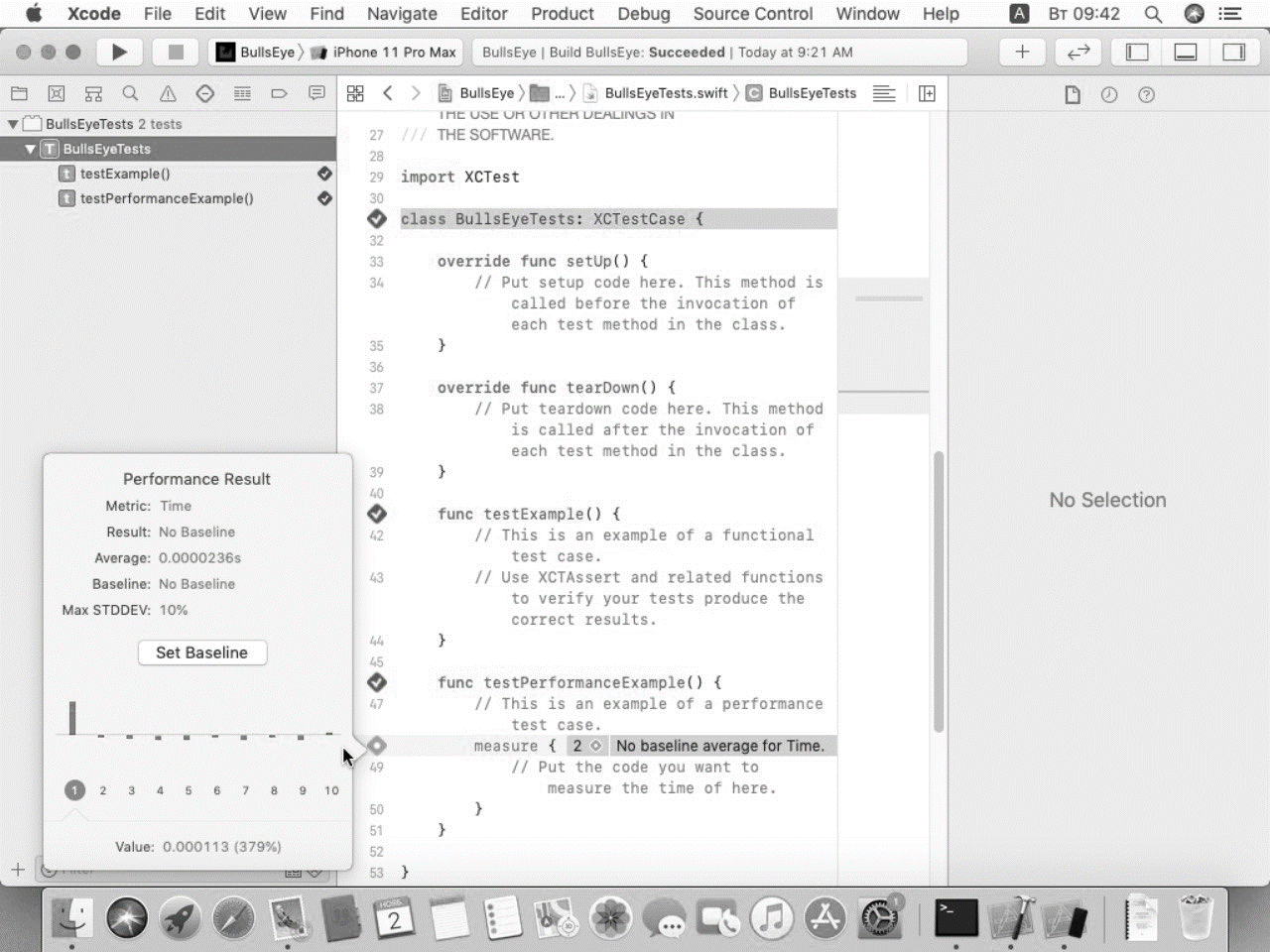


Рисунок 5 – Тест о производительности

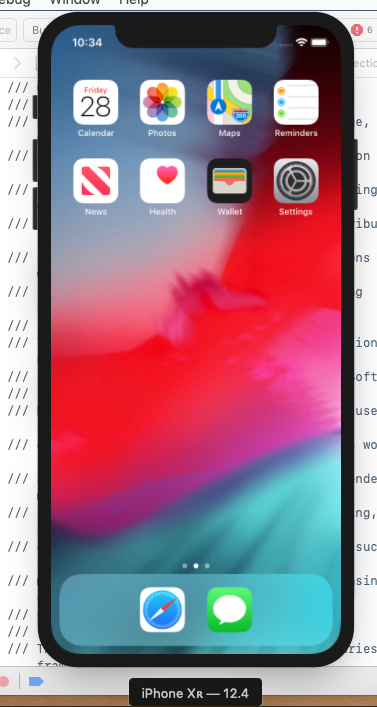


Рисунок 6 – Запуск телефона iPhone Xr

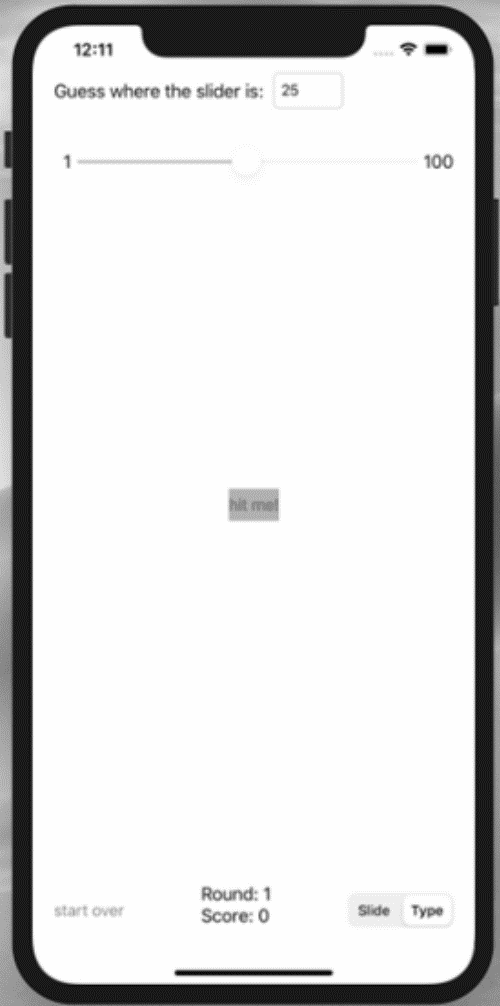


Рисунок 7 – Главный экран

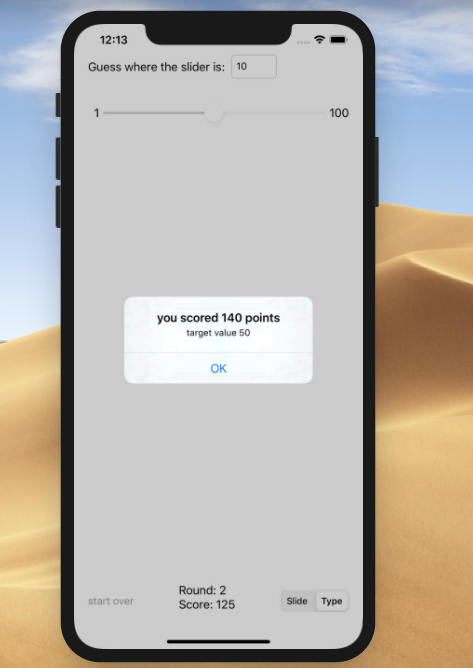


Рисунок 8 – Пункт “Type”

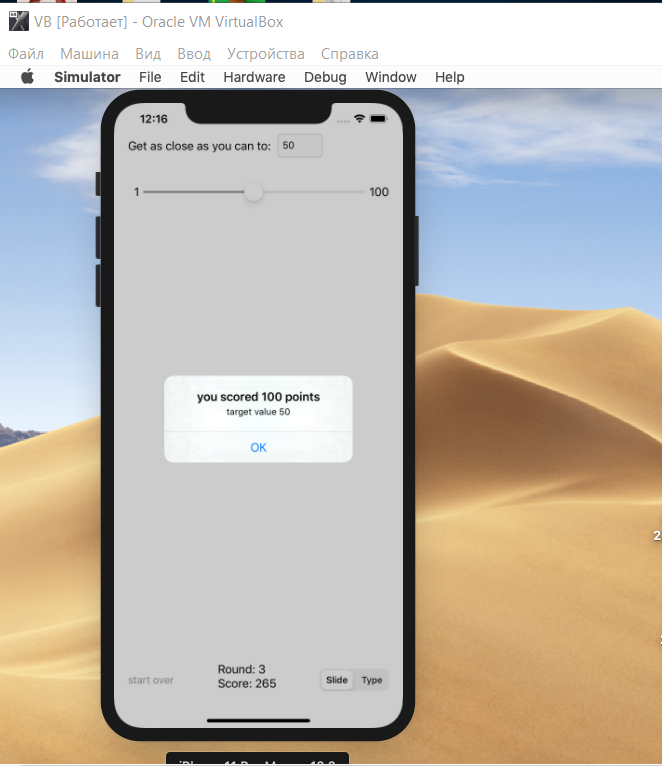


Рисунок 10 – выбор пункта Slide

**Запуск первого теста в коде BullsEyeTests.Swift**

import XCTest

@testable import BullsEye

var gameUnderTest: BullsEyeGame!

class BullsEyeTests: XCTestCase {

override func setUp() {

gameUnderTest = BullsEyeGame ()

gameUnderTest.startNewGame ( )

}

override func tearDown() {

gameUnderTest = nil

}

func testScoreIsComputed () {

let guess = gameUnderTest.targetValue +

5

\_ = gameUnderTest. cneck(guess: guess)

XCTAssertEqual(gameUnderTest.scoreRound, 95, "Score computed from guess is wrong")

}

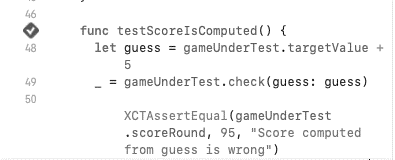


Рисунок 12 – Тест пройден успешно

**Поиск намеренно созданной ошибки:**

При запуске теста, он должен остановиться на строке XCTAssertEqual

func testScoreIsComputedWhenGuessGTTarget()

{

let guess = gameUnderTest.targetValue - 5

\_= gameUnderTest.check(guess: guess)

XCTAssertEqual(gameUnderTest.scoreRound, 95, "Score computed from guess is wrong")

}

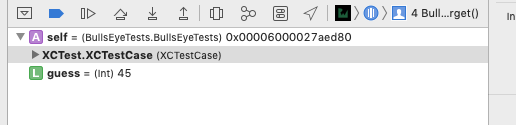


Рисунок 13 – Консоль отладки

**Использование XCTestExpection для тестирования асинхронных операций**

import XCTest

@testable import HalfTunes

var sessionUnderTest: URLSession!

class HalfTunesTests: XCTestCase {

override func setUp() {

super.setUp()

sessionUnderTest = URLSession(configuration: URLSessionConfiguration.default)

}

override func tearDown() {

sessionUnderTest = nil

super.tearDown()

}

func testValidCallToiTunesGetsHTTPStatusCode200() {

let url = URL(string: "https://itunes.apple.com/search?media=music&entity=song&term=abba")

let promise = expectation(description: "Status code: 200")

let dataTask = sessionUnderTest.dataTask(with: url!) { data, response, error in

if let error = error {

XCTFail("Error: \(error.localizedDescription)")

return

} else if let statusCode = (response as? HTTPURLResponse)?.statusCode {

if statusCode == 200 {

promise.fulfill()

} else {

XCTFail("Status code: \(statusCode)")

}

}

}

dataTask.resume()

waitForExpectations(timeout: 5, handler: nil)

}

func testPerformanceExample() {

measure {

}

}

}

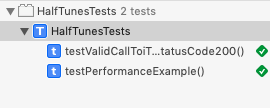


Рисунок 14 – Тест пройден успешно

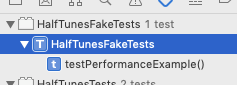


Рисунок 15 – HalfItunesFakeTests

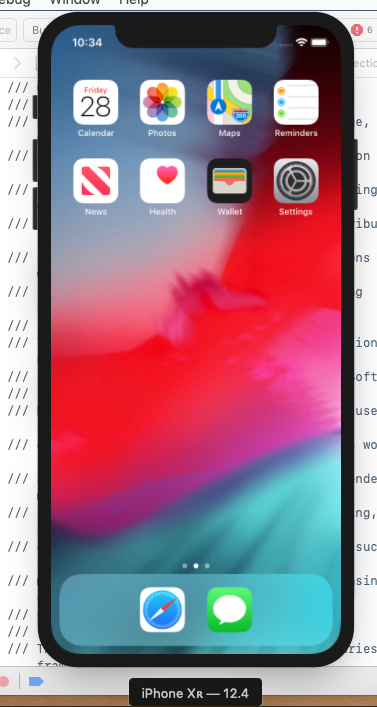


Рисунок 16 – HalfItines



Рисунок 17 – Основное меню

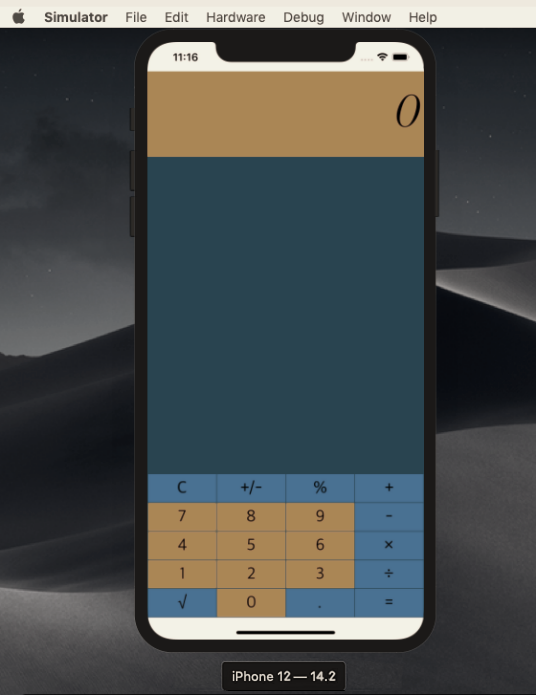


Рисунок 18 – Калькулятор

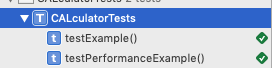


Рисунок 19 – Выполнение тестов

import XCTest

@testable import CALculator

var CALculator:XCTest

class CALculatorTests: XCTestCase{

override func setUP() {

}

override func tearDown() {

}

func testExample() {

}

func testPerfomanceExample(){

self.measure { }

}

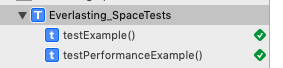


Рисунок 20 – Выполнение тестов



Рисунок 21 – Выполнение программы

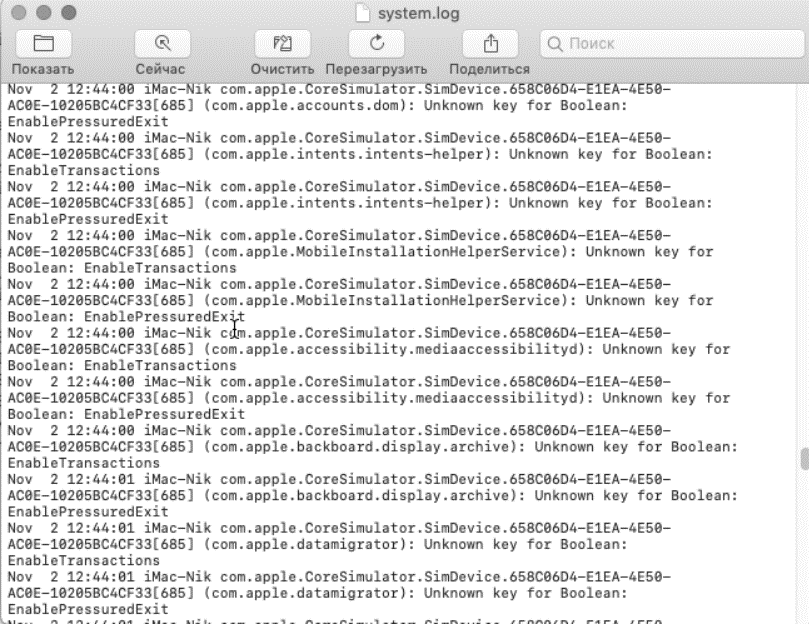


Рисунок 22 – Снятие логов

**Контрольные вопросы:**

**1. Перечислите и кратко опишите основные инструменты тестирования в среде разработки XCode.**

– Fast (Скорость): Тесты должны выполняться быстро, чтобы людям хотелось их выполнять;

– Independent / Isolated (Независимость /

Изолированность): Тесты должны полностью игнорировать друг друга. Результаты выполнения одного теста не должны быть входными данными для другого;

– Repetable (Повторяемость): Вы должны получать одинаковые результаты каждый раз при запуске теста, не зависимо от среды выполнения. Провайдеры/источники внешних данных и вопросы параллелизма могут вызвать перемежающиеся сбои;

– Self-validating (Самодостоверность): Результатом выполнения теста должно быть булево значение. Тест либо пройден, либо нет, и это должно быть легко понятно любому

разработчику;

– Timely(Своевременность): В идеале тесты должны быть написаны непосредственно перед тем, как вы напишете код программного продукта, который они должны проверить.

Следуя этим главным принципам, вы сможете создать простые и полезные тесты, и у вашего приложения не будет проблем с этими тестами.

**2. Назовите несколько тестов которые можно провести только на реальных устройствах.**

BullsEye и HalfTunes.

**3. Опишите основные требования к приложению при загрузке в AppStore.**

Вам нужно войти под своим Apple ID в AppStore и скачать XCode.

**4. Какие вы знаете способы снять краш-логи приложения на IOS.**

Варианты снятия крэш-логов с устройства:

1. Снятие через синхронизацию устройства в iTunes.

2. Снятие логов через программу iTools.

3. Снятие логов в Mac OS.