Introduction to iOS Testing

Nashville CocoaHeads - March 2018

Addison Webb

Why is testing important?

- We want high quality applications
- We want Bug-free apps
- We want code that is easy to update and maintain

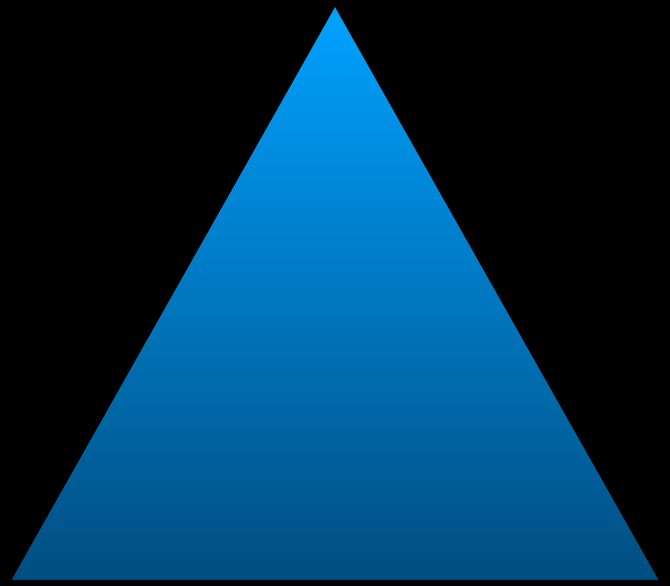
Benefits of Testing

- Helps you find regressions
- Helps document your code
- Makes you a better developer
- Gives you confidence when making changes

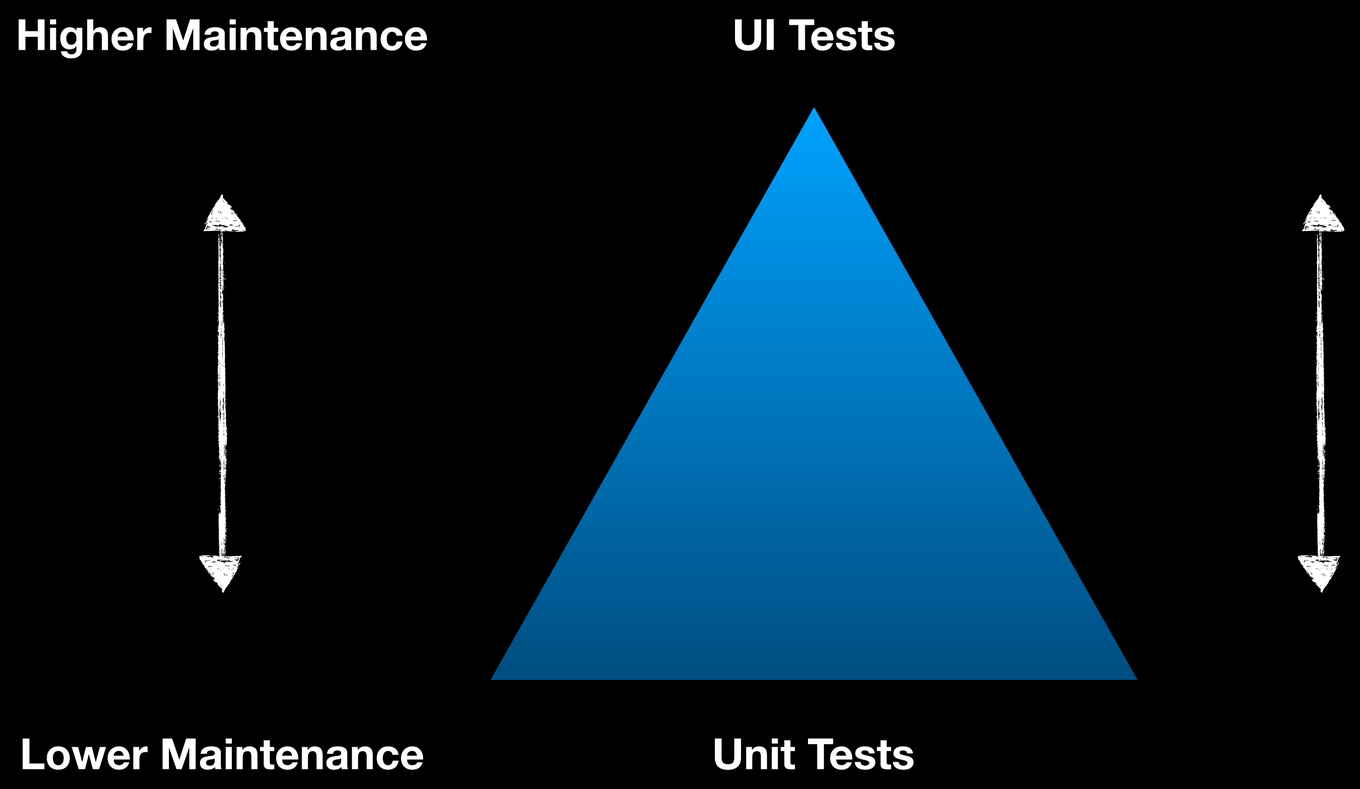
Types of Tests

- Unit Tests
- Integration Tests
- UI Tests

UI Tests



Unit Tests


Higher Maintenance
UI Tests
Lower Maintenance
Unit Tests

Qualities of Good Unit Test

- Repeatable
- Fast
- Automated

Approaches to Testing

- Never write tests
- TDD
- Somewhere in between

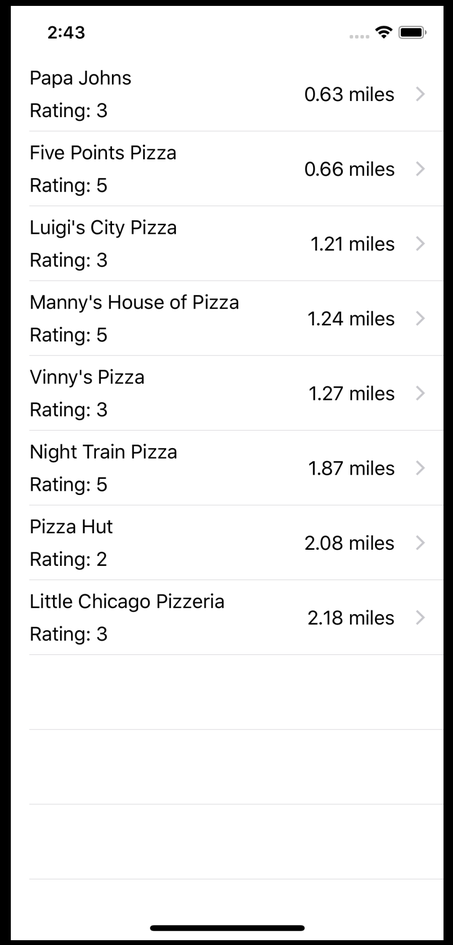
XCTest

- Testing framework that ships with Xcode
- Unit tests
- UI tests
- Performance tests

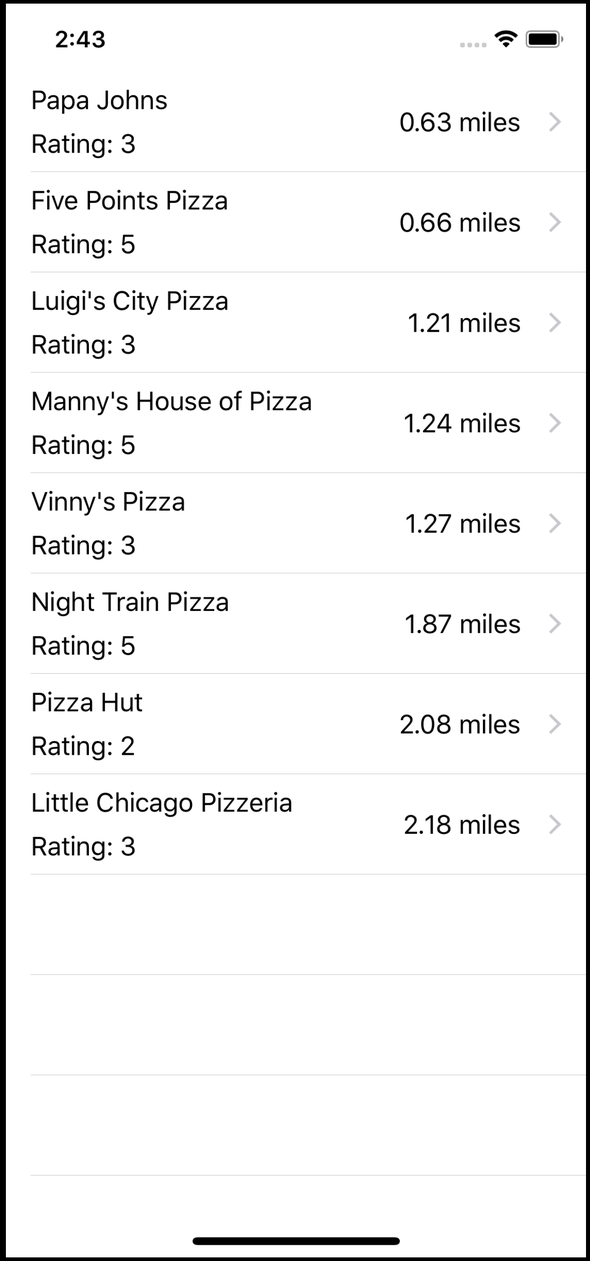
XCTest Subclass

```
import XCTest

class ExampleTests: XCTestCase {

    override func setUp() {
        super.setUp()
        // Put setup code here. This method is called before the invocation of each test method in the class.
    }

    override func tearDown() {
        // Put teardown code here. This method is called after the invocation of each test method in the class.
        super.tearDown()
    }

    func testExample() {
        // This is an example of a functional test case.
        // Use XCTAssert and related functions to verify your tests produce the correct results.
    }

    func testPerformanceExample() {
        // This is an example of a performance test case.
        self.measure {
            // Put the code you want to measure the time of here.
        }
    }

}
```

Nashville Za



```swift
import UIKit

class LocationsTableViewController: UITableViewController {

    private var locations: [Location] = []

    override func viewDidLoad() {
        super.viewDidLoad()

        // get locations
        let dataURL = Bundle.main.url(forResource: "za-data", withExtension: "json")!
        let zaData = try! Data(contentsOf: dataURL, options: [])
        let decoder = JSONDecoder()
        let unsortedLocations = try! decoder.decode([Location].self, from: zaData)
        locations = sortByDistance(unsortedLocations)
    }

    // MARK: - Table view data source

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return locations.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        return LocationTableViewCell(with: locations[indexPath.row])
    }

    private func sortByDistance(_ locations: [Location]) -> [Location] {
        return locations.sorted(by: { (location1: Location, location2: Location) -> Bool in
            return location1.distance < location2.distance
        })
    }
}
```



Location

```
import Foundation
import CoreLocation

struct Location: Codable {
    let name: String
    let rating: Int
    let latitude: Double
    let longitude: Double

    /// straight line distance in miles
    var distance: Double {
        let currentLocation: CLLocation = CLLocation(latitude: 36.174454, longitude: -86.761744)
        let zaLocation = CLLocation(latitude: latitude, longitude: longitude)

        return miles(from: zaLocation.distance(from: currentLocation))
    }

    /// formatted distance string
    var distanceString: String {
        return String(format: "%.2f miles", distance)
    }

    private func miles(from distance: CLLocationDistance) -> Double {
        return distance * 0.000621371
    }
}
```

Location Table View Cell

```
class LocationTableViewCell: UITableViewCell {
    static let cellReuseIdentifier = "LocationTableViewCell"

    private let nameLabel = UILabel()
    private let ratingLabel = UILabel()
    private let distanceLabel = UILabel()

    override init(style: UITableViewCellStyle, reuseIdentifier: String?) {
        super.init(style: style, reuseIdentifier: reuseIdentifier)

        // enable autolayout
        nameLabel.translatesAutoresizingMaskIntoConstraints = false
        ratingLabel.translatesAutoresizingMaskIntoConstraints = false
        distanceLabel.translatesAutoresizingMaskIntoConstraints = false

        // add views to view hierarchy
        contentView.addSubview(nameLabel)
        contentView.addSubview(ratingLabel)
        contentView.addSubview(distanceLabel)

        distanceLabel.setContentHuggingPriority(.defaultHigh, for: .horizontal)

        // layout
        NSLayoutConstraint.activate([
            nameLabel.leadingAnchor.constraint(equalTo: contentView.safeAreaLayoutGuide.leadingAnchor, constant: 16.0),
            nameLabel.trailingAnchor.constraint(greaterThanOrEqualTo: distanceLabel.leadingAnchor, constant: -8.0),
            nameLabel.topAnchor.constraint(equalTo: contentView.safeAreaLayoutGuide.topAnchor, constant: 8.0),
            nameLabel.bottomAnchor.constraint(equalTo: ratingLabel.topAnchor, constant: -8.0),

            ratingLabel.leadingAnchor.constraint(equalTo: nameLabel.leadingAnchor),
            ratingLabel.trailingAnchor.constraint(equalTo: nameLabel.trailingAnchor),
            ratingLabel.bottomAnchor.constraint(equalTo: contentView.safeAreaLayoutGuide.bottomAnchor, constant: -8.0),

            distanceLabel.trailingAnchor.constraint(equalTo: contentView.safeAreaLayoutGuide.trailingAnchor, constant: -8.0),
            distanceLabel.centerYAnchor.constraint(equalTo: contentView.safeAreaLayoutGuide.centerYAnchor)
            ])

        accessoryType = .disclosureIndicator
    }

    func configureCell(with location: Location) {
        nameLabel.text = location.name
        ratingLabel.text = "Rating: \(location.rating)"
        distanceLabel.text = location.distance.milesString
    }
```

```swift
import UIKit

class LocationsTableViewController: UITableViewController {

    private var locations: [Location] = []

    override func viewDidLoad() {
        super.viewDidLoad()

        // get locations
        let dataURL = Bundle.main.url(forResource: "za-data", withExtension: "json")!
        let zaData = try! Data(contentsOf: dataURL, options: [])
        let decoder = JSONDecoder()
        let unsortedLocations = try! decoder.decode([Location].self, from: zaData)
        locations = sortByDistance(unsortedLocations)
    }

    // MARK: - Table view data source

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return locations.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        return LocationTableViewCell(with: locations[indexPath.row])
    }

    private func sortByDistance(_ locations: [Location]) -> [Location] {
        return locations.sorted(by: { (location1: Location, location2: Location) -> Bool in
            return location1.distance < location2.distance
        })
    }
}
```

Testing is easier when…

- Methods have 1 job
- Take a set of inputs
- Produce 1 output

Sort Function

```
extension Array where Element == Location {

    /// Sort locations by distance. (shortest distance first)
    ///
    /// - Returns: returns sorted array of Locations with the closest locations first
    func sortByDistance() -> [Location] {
        return self.sorted(by: { (location1: Location, location2: Location) -> Bool in
            return location1.distance < location2.distance
        })
    }
}
```

Example

```
class NashvilleZaTests: XCTestCase {

    private let locations: [Location] = [
        Location(name: "Five Points Pizza", rating: 5, latitude: 36.177196, longitude: -86.750438),
        Location(name: "Luigi's City Pizza", rating: 3, latitude: 36.161238, longitude: -86.775878),
        Location(name: "Papa Johns", rating: 3, latitude: 36.166786, longitude: -86.767990),
        Location(name: "Night Train Pizza", rating: 5, latitude: 36.151734, longitude: -86.780040),
        Location(name: "Little Chicago Pizzeria", rating: 3, latitude: 36.153062, longitude: -86.790537),
        Location(name: "Vinny's Pizza", rating: 3, latitude: 36.163647, longitude: -86.780069),
        Location(name: "Manny's House of Pizza", rating: 5, latitude: 36.164354, longitude: -86.780004),
        Location(name: "Pizza Hut", rating: 2, latitude: 36.199149, longitude: -86.740257)
    ]

    func testLocationsSortDistance() {
        let sortedLocations = locations.sortByDistance()

        XCTAssert(sortedLocations[0].name == "Papa Johns")
        XCTAssert(sortedLocations[1].name == "Five Points Pizza")
        XCTAssert(sortedLocations[2].name == "Luigi's City Pizza")
        XCTAssert(sortedLocations[3].name == "Manny's House of Pizza")
        XCTAssert(sortedLocations[4].name == "Vinny's Pizza")
        XCTAssert(sortedLocations[5].name == "Night Train Pizza")
        XCTAssert(sortedLocations[6].name == "Pizza Hut")
        XCTAssert(sortedLocations[7].name == "Little Chicago Pizzeria")
    }
```

Common Pattern

- Arrange
- Act
- Assert

```swift
import Foundation
import CoreLocation

struct Location: Codable {
    let name: String
    let rating: Int
    let latitude: Double
    let longitude: Double

    /// straight line distance in miles
    var distance: Double {
        let currentLocation: CLLocation = CLLocation(latitude: 36.174454, longitude: -86.761744)
        let zaLocation = CLLocation(latitude: latitude, longitude: longitude)

        return miles(from: zaLocation.distance(from: currentLocation))
    }

    /// formatted distance string
    var distanceString: String {
        return String(format: "%.2f miles", distance)
    }

    private func miles(from distance: CLLocationDistance) -> Double {
        return distance * 0.000621371
    }
}
```

New Location

```
struct Location: Codable {
    let name: String
    let rating: Int
    let latitude: Double
    let longitude: Double

    /// straight line distance in miles
    var distance: CLLocationDistance {
        let currentLocation: CLLocation = CLLocation(latitude: 36.174454, longitude: -86.761744) // faked location
        let zaLocation = CLLocation(latitude: latitude, longitude: longitude)

        return zaLocation.distance(from: currentLocation)
    }
}
```

```swift
extension CLLocationDistance {
    var miles: Double {
        return self * 0.000621371
    }

    /// formatted distance string
    var milesString: String {
        return String(format: "%.2f miles", miles)
    }
}
```

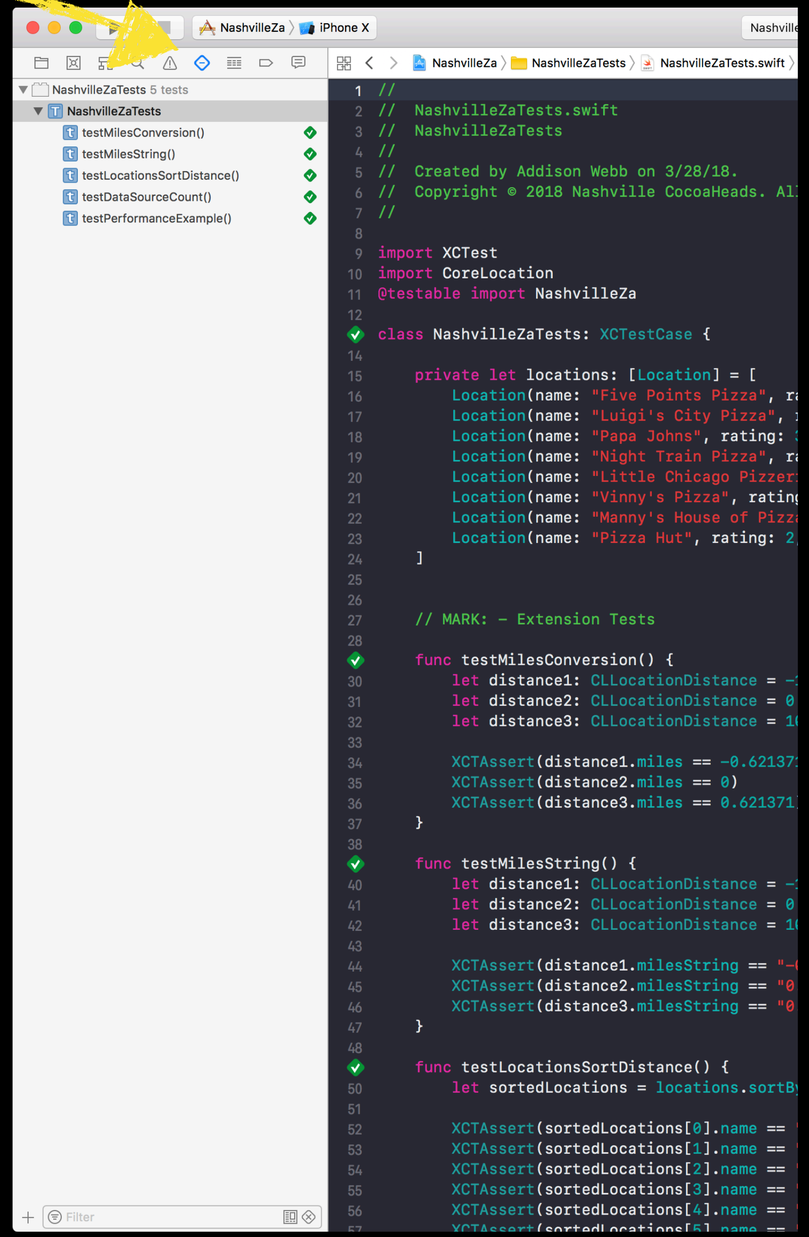
Example 2

```
func testMilesConversion() {
    let distance1: CLLocationDistance = -1000
    let distance2: CLLocationDistance = 0
    let distance3: CLLocationDistance = 1000

    XCTAssert(distance1.miles == -0.621371)
    XCTAssert(distance2.miles == 0)
    XCTAssert(distance3.miles == 0.621371)
}

func testMilesString() {
    let distance1: CLLocationDistance = -1000
    let distance2: CLLocationDistance = 0
    let distance3: CLLocationDistance = 1000

    XCTAssert(distance1.milesString == "-0.62 miles")
    XCTAssert(distance2.milesString == "0.00 miles")
    XCTAssert(distance3.milesString == "0.62 miles")
}
```

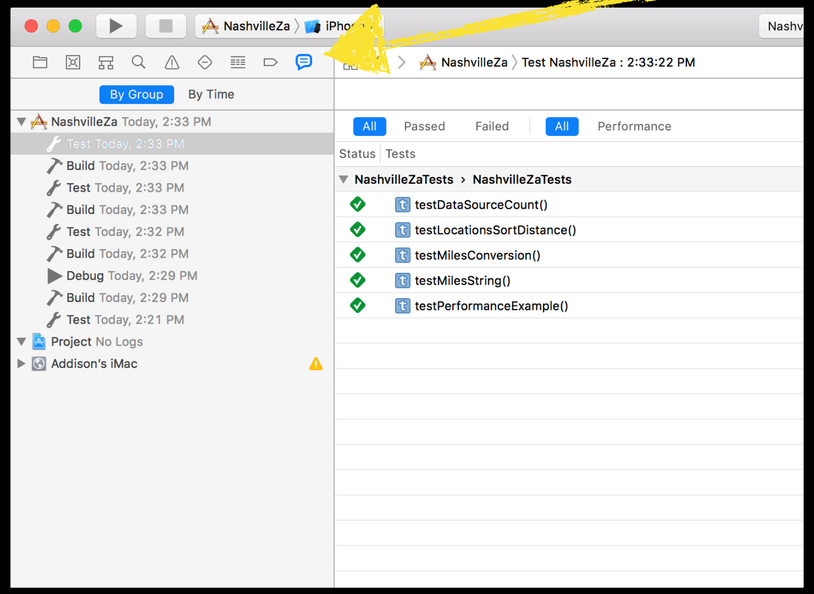
Xcode

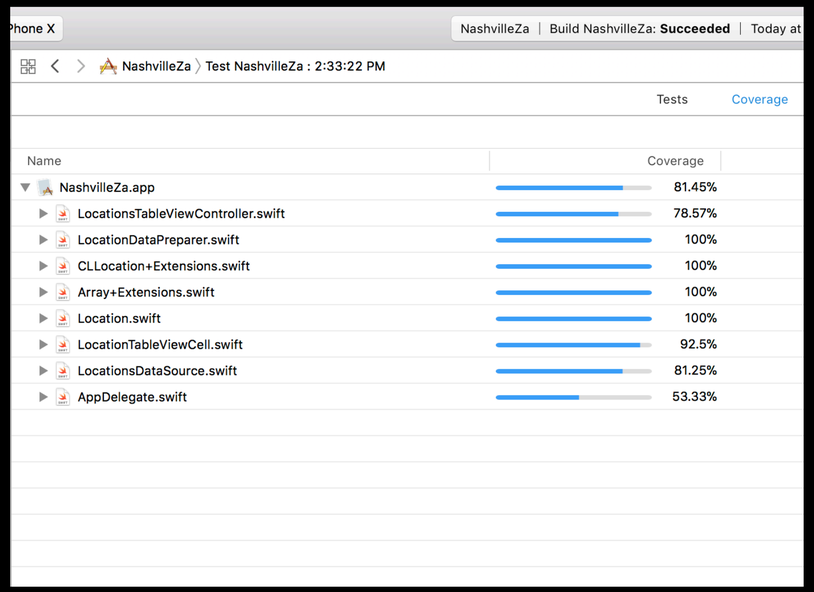
- ⌘ + U - Will run your tests
- Test Navigator
- Report Navigator

Test Navigator



Report Navigator





Test Coverage

In Conclusion…

- Look for code you can separate into logical units
- Write code in a way that reduces side-effects (i.e. SRP)
- Use tests as a indicator of code quality

Questions