# Testing in iOS

Hands-on Challenges

# Testing in iOS Hands-on Challenges

# Challenge A: More unit tests!

Now that you have a test target and an initial test to study and admire, it's time to write some more tests of your own! In this short challenge, you'll do just that.

These are the two methods in `PancakeHouseCollection` to test next:

```
func addPancakeHouse(pancakeHouse: PancakeHouse)

func removePancakeHouse(pancakeHouse: PancakeHouse)
```

Seems pretty simple! These collections are powered by arrays, which means you can add a pancake house that's already in the array.

Here are some hints on how to test these methods. If you want the complete walkthrough on how to test `addPancakeHouse(_:)`, keep reading beyond the hints!

## Hints

To test the add method, you can grab one of the elements in the collection and add it. For the assertion, you can store the count of the collection before and then after you add something. The post-add count should be one greater than before.

To test the remove method, should be pretty similar. Grab the first pancake house in the collection and then try removing it. Again, you could compare the two counts of the collection.

## Add testing

Here's a sample solution on testing the add method:

```
func testAdd() {
  let pancakeData = ["name": "Test Pancake House",
   "priceGuide": 1, "rating": 1, "details": "Test"]
  let samplePancakeHouse = PancakeHouse(dict: pancakeData)!
  let startCount = collection.count

  collection.addPancakeHouse(samplePancakeHouse)
  XCTAssertEqual(collection.count, startCount + 1)
}
```

Just for fun, you can see the code to create a new pancake house from scratch rather than use one already in the array. The `PancakeHouse` initializer takes a dictionary of initial values.

> **Note:** If you're curious check out **PancakeHouse.swift** for the code, and **pancake_houses.plist** for the set of sample data.

Next, you store the starting count of the collection into `startCount`.

Finally, you add the new pancake house to the collection and then assert that the current collection's count should be the starter count + 1.

Note that the code here uses `XCTAssertEqual()`. You could also use regular `XCTAssert` with the `==` comparison; that would be equivalent and just a matter of style on which you prefer.

## Über challenge

Refer back to the hints on the previous page on how to test the remove method. You can also check out the completed challenge in the resources for this video for a sample solution.