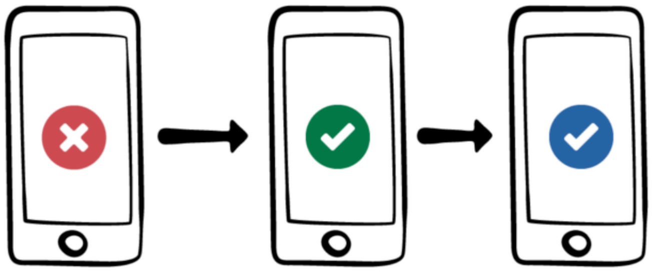
BEGINNING iOS UNIT & UI TESTING



HANDS-ON CHALLENGES

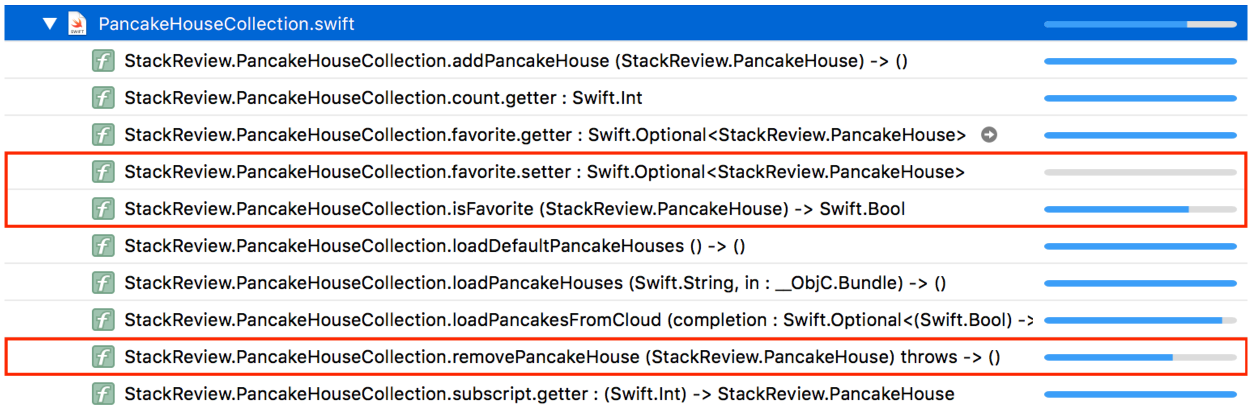
Beginning iOS Unit and UI Testing

Joshua Greene

4 Performance - Challenge

By Joshua Greene

According to the code coverage report, there are a few areas missing tests related to favorite pancake houses.



PancakeHouseCollection.swift

StackReview.PancakeHouseCollection.addPancakeHouse (StackReview.PancakeHouse) -> ()

StackReview.PancakeHouseCollection.count.getter : Swift.Int

StackReview.PancakeHouseCollection.favorite.getter : Swift.Optional<StackReview.PancakeHouse>

StackReview.PancakeHouseCollection.favorite.setter : Swift.Optional<StackReview.PancakeHouse>

StackReview.PancakeHouseCollection.isFavorite (StackReview.PancakeHouse) -> Swift.Bool

StackReview.PancakeHouseCollection.loadDefaultPancakeHouses () -> ()

StackReview.PancakeHouseCollection.loadPancakeHouses (Swift.String, in : __ObjC.Bundle) -> ()

StackReview.PancakeHouseCollection.loadPancakesFromCloud (completion : Swift.Optional<(Swift.Bool) ->

StackReview.PancakeHouseCollection.removePancakeHouse (StackReview.PancakeHouse) throws -> ()

StackReview.PancakeHouseCollection.subscript.getter : (Swift.Int) -> StackReview.PancakeHouse

It's time to close the code coverage gap!

Challenge

Based on the documentation comments, `PancakeHouseCollection` should behave as follows:

- Once you set a favorite pancake house via the setter, you can check if any given pancake house is the favorite using `isFavorite(_:)`.
- You cannot set a pancake house as the favorite if it isn't already in the collection.

Your challenge is to write two new unit tests, one for each of these behaviors.

Hints

For the first test, grab the first pancake house using the subscript method; set this as the `favorite`; and assert `isFavorite(_:)` returns `true` when passed this pancake house.

To ensure `isFavorite(_:)` doesn't *always* return `true`, assert `isFavorite(_:)` returns `false` when passed a different pancake house.

For the second test, you can create a new pancake house using a dictionary like this one:

```
  let dictionary: [String : Any] = ["name": "Test Pancake House",
                                    "priceGuide": 1,
                                    "rating": 1,
                                    "details": "Test"]
```

Try setting this as the `favorite`, which isn't allowed, and then assert the `favorite` is `nil`.

These tests are similar to ones you've wrote before. Try to write these yourself first. If you get stuck, check out the next page for sample solutions.

Solution

Setting and checking a favorite

Open `PancakeHouseCollectionTests.swift` and add the following test method to the class:

```
func testSetFavoritePancake() {

  // given
  let pancakeHouse = collection[0]
  let otherPancakeHouse = collection[1]

  // when
  collection.favorite = pancakeHouse

  // then
  XCTAssertTrue(collection.isFavorite(pancakeHouse))
  XCTAssertFalse(collection.isFavorite(otherPancakeHouse))
}
```

You know the test data set has three pancake houses, so it's safe to use the subscript to get the first two pancake houses.

Once you set the favorite with the setter, all you need is two assertions to check both cases of `isFavorite(_:)`: one for when it matches the favorite and returns `true`, and another for when it doesn't match and returns `false`.

Setting an unknown favorite

Add the following test method to `PancakeHouseCollectionTests`:

```
func testGivenFavoriteNotInCollectionDoesntSetFavorite() {

  // given
  let dictionary: [String : Any] = ["name": "Test Pancake House",
                                    "priceGuide": 1,
                                    "rating": 1,
                                    "details": "Test"]
  let pancakeHouse = PancakeHouse(dictionary: dictionary)!

  // when
  collection.favorite = pancakeHouse

  // then
  XCTAssertNil(collection.favorite)
}
```

The first two lines of the method set up a new pancake house.

You then try to set it as the favorite without adding it to the collection first.

Remember, each test method is proceeded by a call to `setUp()`, which means there will be a fresh set of test data. So, you don't have to worry about test data from a previous test interfering with this one.

Lastly, you assert the `collection.favorite` wasn't set by asserting it's `nil`.

Über challenge

In a previous challenge, you wrote tests to cover `addPancakeHouse(_:)` and `removePancakeHouse(_:)`. However, if you read the comment for `removePancakeHouse(_:)`, you'll see the following note:

> @discussion If the pancake house isn't part of the collection, throws `PancakseHouseError.triedToRemoveUnknownPancakeHouse` error. If the pancake house is the current favorite, throws `PancakseHouseError.triedToRemoveFavoritePancakeHouse` error.

You don't have any tests that verify this behavior happens as expected! So, you need to add them!

Remember to refactor your tests as you go along!

You should be able to refactor out a new helper method that creates a new `PancakeHouse`. The method signature should look like this: `func givenNewPancakeHouse() -> PancakeHouse`.

If you get stuck, check out the completed challenge project in the resources for this video for a sample solution. Good luck!