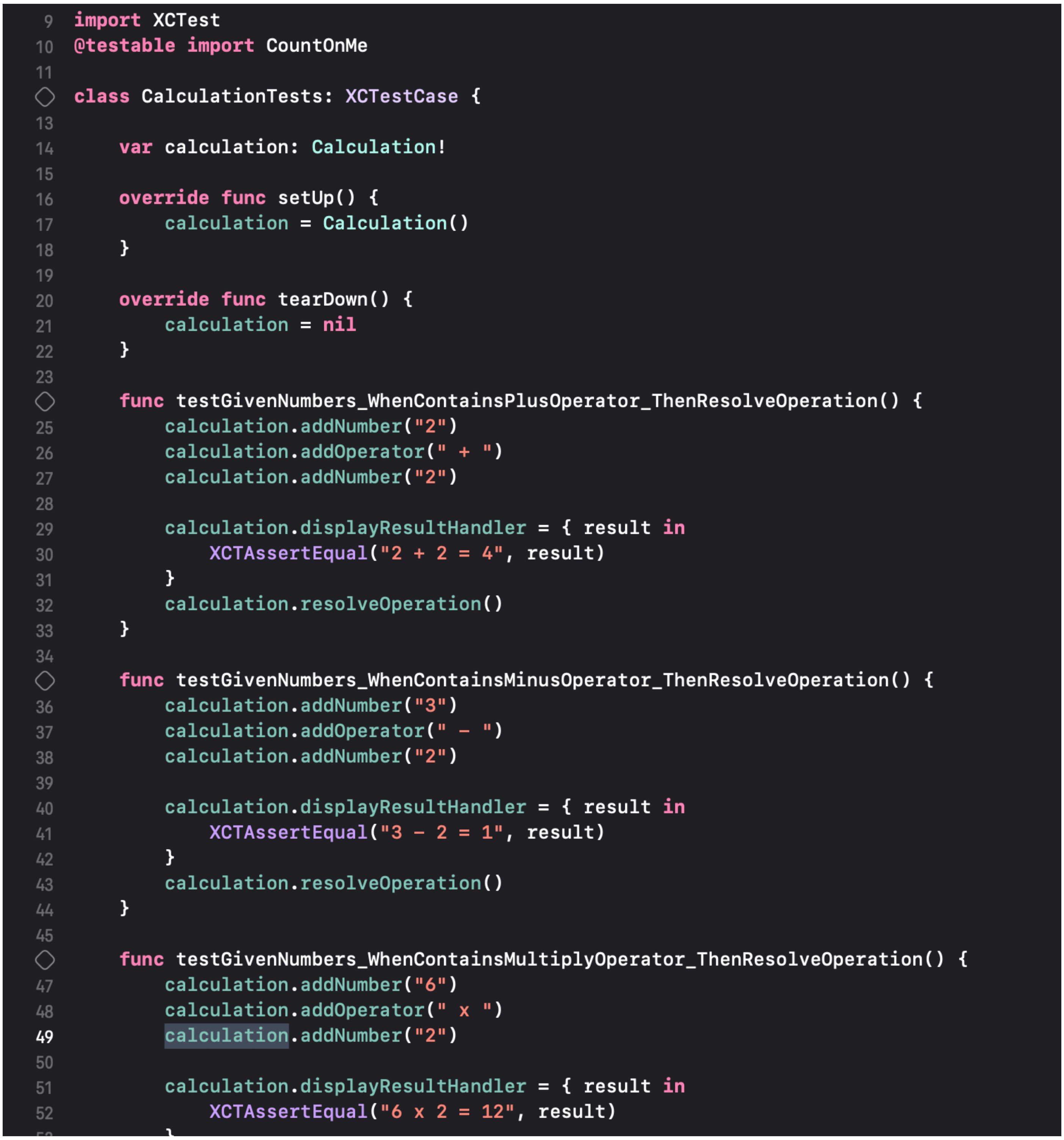
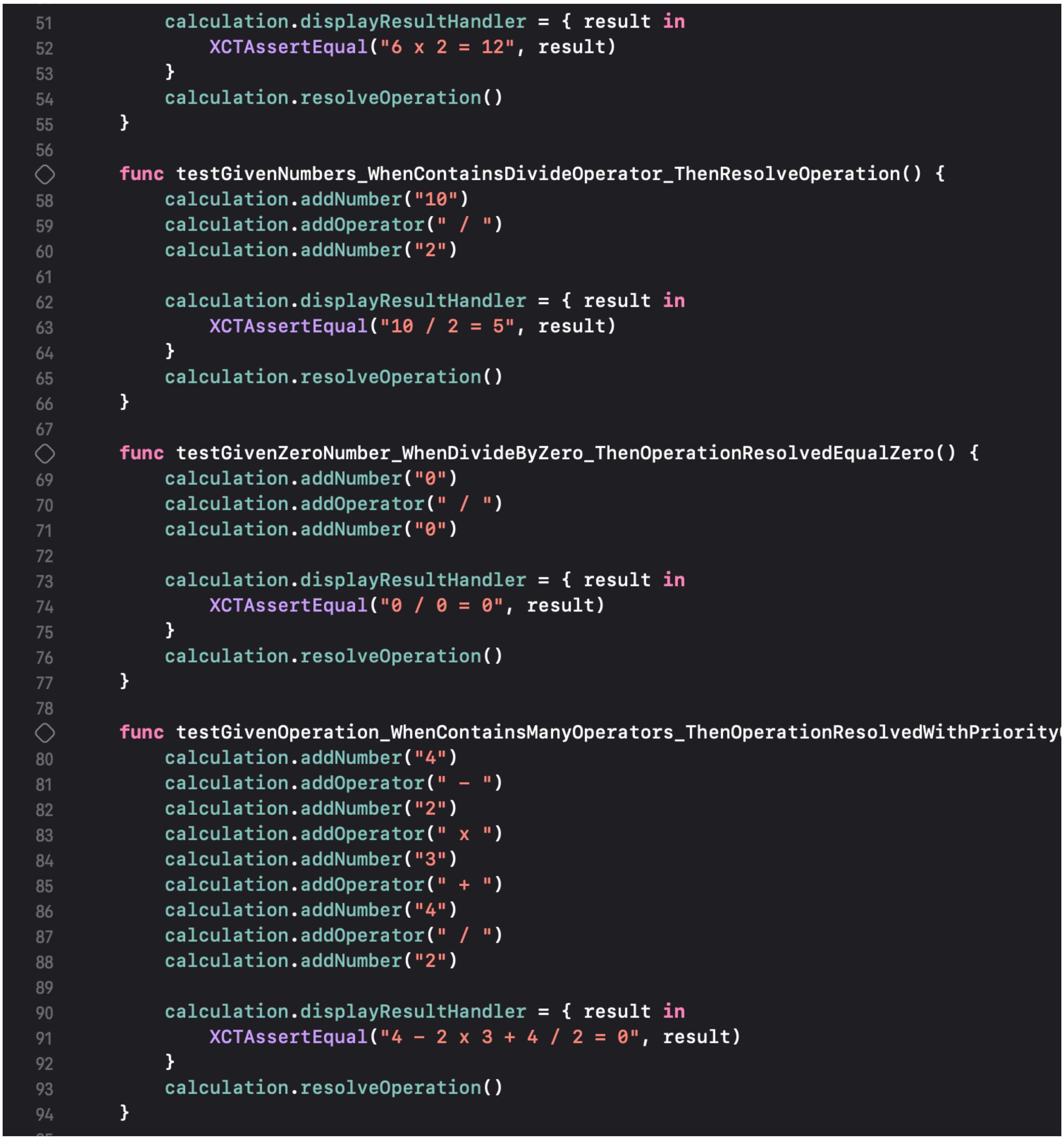
Tests Unitaires



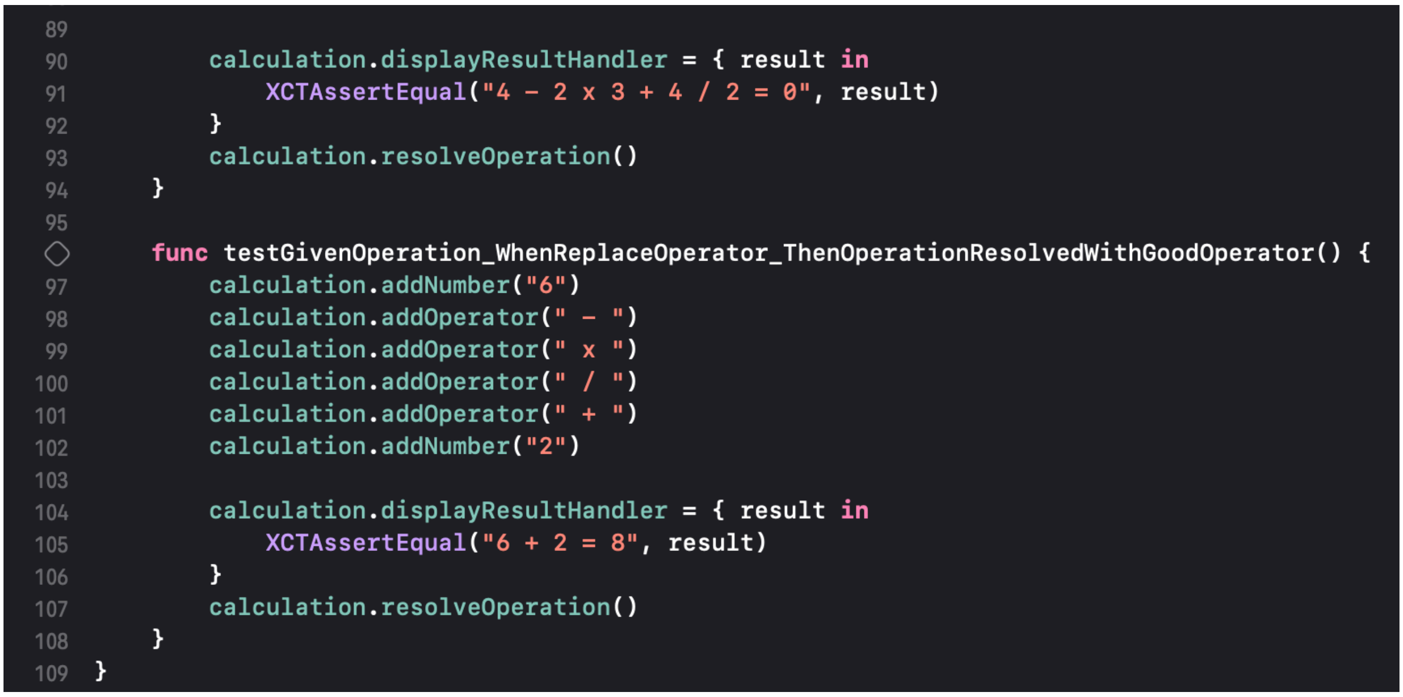
```swift
import XCTest
@testable import CountOnMe

class CalculationTests: XCTestCase {

    var calculation: Calculation!

    override func setUp() {
        calculation = Calculation()
    }

    override func tearDown() {
        calculation = nil
    }

    func testGivenNumbers_WhenContainsPlusOperator_ThenResolveOperation() {
        calculation.addNumber("2")
        calculation.addOperator(" + ")
        calculation.addNumber("2")

        calculation.displayResultHandler = { result in
            XCTAssertEqual("2 + 2 = 4", result)
        }
        calculation.resolveOperation()
    }

    func testGivenNumbers_WhenContainsMinusOperator_ThenResolveOperation() {
        calculation.addNumber("3")
        calculation.addOperator(" - ")
        calculation.addNumber("2")

        calculation.displayResultHandler = { result in
            XCTAssertEqual("3 - 2 = 1", result)
        }
        calculation.resolveOperation()
    }

    func testGivenNumbers_WhenContainsMultiplyOperator_ThenResolveOperation() {
        calculation.addNumber("6")
        calculation.addOperator(" x ")
        calculation.addNumber("2")

        calculation.displayResultHandler = { result in
            XCTAssertEqual("6 x 2 = 12", result)
```

Tests Unitaires



```swift
        calculation.displayResultHandler = { result in
            XCTAssertEqual("6 x 2 = 12", result)
        }
        calculation.resolveOperation()
    }

    func testGivenNumbers_WhenContainsDivideOperator_ThenResolveOperation() {
        calculation.addNumber("10")
        calculation.addOperator(" / ")
        calculation.addNumber("2")

        calculation.displayResultHandler = { result in
            XCTAssertEqual("10 / 2 = 5", result)
        }
        calculation.resolveOperation()
    }

    func testGivenZeroNumber_WhenDivideByZero_ThenOperationResolvedEqualZero() {
        calculation.addNumber("0")
        calculation.addOperator(" / ")
        calculation.addNumber("0")

        calculation.displayResultHandler = { result in
            XCTAssertEqual("0 / 0 = 0", result)
        }
        calculation.resolveOperation()
    }

    func testGivenOperation_WhenContainsManyOperators_ThenOperationResolvedWithPriority
        calculation.addNumber("4")
        calculation.addOperator(" - ")
        calculation.addNumber("2")
        calculation.addOperator(" x ")
        calculation.addNumber("3")
        calculation.addOperator(" + ")
        calculation.addNumber("4")
        calculation.addOperator(" / ")
        calculation.addNumber("2")

        calculation.displayResultHandler = { result in
            XCTAssertEqual("4 - 2 x 3 + 4 / 2 = 0", result)
        }
        calculation.resolveOperation()
    }
```

Tests Unitaires



```swift

        calculation.displayResultHandler = { result in
            XCTAssertEqual("4 - 2 x 3 + 4 / 2 = 0", result)
        }
        calculation.resolveOperation()
    }

    func testGivenOperation_WhenReplaceOperator_ThenOperationResolvedWithGoodOperator() {
        calculation.addNumber("6")
        calculation.addOperator(" - ")
        calculation.addOperator(" x ")
        calculation.addOperator(" / ")
        calculation.addOperator(" + ")
        calculation.addNumber("2")

        calculation.displayResultHandler = { result in
            XCTAssertEqual("6 + 2 = 8", result)
        }
        calculation.resolveOperation()
    }
}
```