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

```
51          calculation.displayResultHandler = { result in
52              XCTAssertEqual("6 x 2 = 12", result)
53          }
54          calculation.resolveOperation()
55      }
56
57      func testGivenNumbers_WhenContainsDivideOperator_ThenResolveOperation() {
58          calculation.addNumber("10")
59          calculation.addOperator(" / ")
60          calculation.addNumber("2")
61
62          calculation.displayResultHandler = { result in
63              XCTAssertEqual("10 / 2 = 5", result)
64          }
65          calculation.resolveOperation()
66      }
67
68      func testGivenZeroNumber_WhenDivideByZero_ThenOperationResolvedEqualZero() {
69          calculation.addNumber("0")
70          calculation.addOperator(" / ")
71          calculation.addNumber("0")
72
73          calculation.displayResultHandler = { result in
74              XCTAssertEqual("0 / 0 = 0", result)
75          }
76          calculation.resolveOperation()
77      }
78
79      func testGivenOperation_WhenContainsManyOperators_ThenOperationResolvedWithPriority(
80          calculation.addNumber("4")
81          calculation.addOperator(" - ")
82          calculation.addNumber("2")
83          calculation.addOperator(" x ")
84          calculation.addNumber("3")
85          calculation.addOperator(" + ")
86          calculation.addNumber("4")
87          calculation.addOperator(" / ")
88          calculation.addNumber("2")
89
90          calculation.displayResultHandler = { result in
91              XCTAssertEqual("4 - 2 x 3 + 4 / 2 = 0", result)
92          }
93          calculation.resolveOperation()
94      }
```

Tests Unitaires

```
89
90          calculation.displayResultHandler = { result in
91              XCTAssertEqual("4 - 2 x 3 + 4 / 2 = 0", result)
92          }
93          calculation.resolveOperation()
94      }
95
96      func testGivenOperation_WhenReplaceOperator_ThenOperationResolvedWithGoodOperator() {
97          calculation.addNumber("6")
98          calculation.addOperator(" - ")
99          calculation.addOperator(" x ")
100         calculation.addOperator(" / ")
101         calculation.addOperator(" + ")
102         calculation.addNumber("2")
103
104         calculation.displayResultHandler = { result in
105             XCTAssertEqual("6 + 2 = 8", result)
106         }
107         calculation.resolveOperation()
108     }
109 }
```