Lab 4.1 - Unit Testing

Introduction

For this lab, you will be working on a calculator application. The application itself has already been written, but the unit testing is incomplete.

The calculator only works with string parameters. For example, to add the numbers 100 and 200, you would pass the string "100,200" to the Add method. The calculation methods all return integers.

Starter projects

There is a starter project for this lab, available in C#, Java and JavaScript versions.

Goals

Your job is to complete the unit testing of the calculator class.

1. Write unit test(s) to demonstrate the correctness of each of these methods:
    - Subtract
    - Multiply
    - Divide (including what you expect to happen if you ask for a division by zero)
2. You should aim to write at least five (5) separate unit tests.
3. Try to include tests that show how the calculator methods deal with good input in various formats, as well as input which is badly formatted in various ways.

Rules

1. All tests must pass (without commenting any of them out!)
2. Do not update the functions to make the tests pass - the functions should be right. (If a test fails, it probably indicates a problem with the test itself).
3. Remember to keep your git repository updated.

Think about...

- Different types of valid data
- Different types of invalid data
- Boundary conditions

Java notes

The Java starter project uses the **JUnit** framework.

C# notes

The C# starter project uses the **NUnit** framework.

After you open the solution, you may need to right-click on the solution in solution explorer, then click **Restore NuGet packages**. After that, right-click solution and select **Clean build**, then right click solution and select **Rebuild**. After that, you should be able to run your tests OK.

JavaScript notes

The JavaScript starter project uses the **chai** and **mocha** unit testing frameworks.

After opening the JavaScript project, do:

```
npm install
```

To run your tests, do:

```
npm test
```