Lab 6.2: Behaviour-driven Development (BDD)

b

In this lab, you will develop some additional tests and functionality for an existing project, using the behaviour-driven development (BDD) style of testing.

BDD tests take the form of stories, expressed in the format GIVEN ... WHEN ... THEN ...

To complete the lab, you will need to write the new scenarios, as well as adding any additional plumbing code necessary to link the new elements of the scenario (such as add / multiply) to the underlying code under test (so the BDD scenario-runner knows how to execute the tests).

Ideally, you should follow a similar process to TDD when doing BDD, i.e., write a scenario, see it fail, then write the simplest code possible to make the scenario pass (finally moving on to another scenario).

Starter projects

Starter projects are available for this lab in C#, Java, and JavaScript.

Goals

1. Run the existing test scenarios, and make sure they all pass.
2. Add a new scenario (story) to show that the correct answer is produced when you subtract two numbers.
3. Add a new scenario (story) to show that the correct answer is produced when you multiply two numbers.

Java notes

The Java project is using the **JBehave** framework to support BDD testing, in combination with **JUnit**.

Even though the tests are written using BDD stories, you run the tests via **Run as > JUnit test**.

C# notes

The C# project is using the **SpecFlow** framework to support BDD testing, in combination with **NUnit**.

For the best development experience, you should add the **SpecFlow for Visual Studio** extension into your installation of Visual Studio. There are downloads available for Visual Studio 2015, 2017, and 2019.

JavaScript notes

The C# project is using the **Cucumber-JS** framework to support BDD testing, in combination with **Chai**.

First, ensure all dependencies have been installed:

```
npm install
```

To run the BDD tests, do:

```
npm run test
```