Lab 4.3 - Code Coverage

Introduction

A set of unit tests have already been written for you, but they could be better - in particular, they could exercise more of the code which has been written.

Starter projects

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

Goals

Set up a code coverage analyser in your IDE, and measure the code coverage that is achieved when the unit tests are run.

Examine the areas where code coverage is below 100%

Determine what you could do to increase the code coverage, ideally to 100%. Would more tests help?

Try implementing your changes, re-running the tests, and re-measuring the code coverage.

Keep going until you get the coverage % as high as you can.

Java notes

Code coverage is already built into Eclipse. You can activate it by running the project using:

Coverage As > JUnit Test

However, if there is no suitable code coverage tool available in your environment, then try these:

- Eclemma / Emma (Eclipse) https://www.eclemma.org/
- IntelliJ (JetBrains) https://www.jetbrains.com/help/idea/code-coverage.html
- NetBeans http://wiki.netbeans.org/TutorialCodeCoveragePlugin

C# notes

The free versions of Visual Studio do not include code coverage tools built-in. You need to follow these steps to install a coverage tool.

- Tools menu > Extensions and updates > Manage extensions
- Online > Visual Studio Marketplace > search for **dotcover**
- Install the JetBrains tool DotCover
- Restart Visual Studio
- In the Extensions menu, you may need to agree to start a trial of the **dotCover** product
- To generate code coverage statistics while running the project's unit tests, use:
  ReSharper menu > Unit Tests > Cover all unit tests from solution

JavaScript notes

Install the code coverage package named **nyc**

```
npm install nyc --save-dev
```

Add an npm task to package.json

```
"coverage": "nyc npm run test"
```

To run the code coverage tool in the terminal (while running all the unit tests in the project), use:

```
npm run coverage
```