Digital & Technology Solutions

Degree Apprenticeship

Test Driven Development –

Level 4 –

20 credits -

Written by: Peter Behague

Checked by:

Programme Leader Approval

UoR Approval:

Approved for Single / Multiple Use:

Review (Multiple Use): 12 months from creation

# Assessment Brief

This assessment brief provides details of the overall assessment for your module. Where a module has multiple components these are listed below. It will provide outline details of the examination and specific instructions for any coursework elements. [Section 1](#_Section_1_Module) provides the detail of the assessment and [Section 2](#_Section_2_General) provides overarching details to support your assessment.

[Component 1](#_Exam_Overview_(Delete): Summative Assignment (75%)

Description: Coding assignment – Economic Info.

[Component 2](#_Coursework_Brief): Summative Assignment (25%)

Description: 1500 - 2500 word report

The mark will be awarded based on an overall judgement of both components. Both elements must be attempted and an overall mark of at least 40% must be achieved to pass the module.

# Submission details

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Date | Time | URL |
| Coding Assignment | 19/1/2020 | 23.59 |  |
| Report | 19/1/2020 | 23.59 |  |

# Module Learning Outcome Assessment Matrix

|  |  |  |
| --- | --- | --- |
| Learning Outcome | Comp1 | Comp2 |
| Able to write unit tests using a unit test environment such as Junit, NUnit or Microsoft’s “Unit Testing Framework” |  | X |
| Demonstrate knowledge of the TDD approach to software development and understand its strengths and weaknesses. | X | X |
| Demonstrate an understanding of the key techniques and strategies of TDD, and principles of refactoring code | X | X |
| Explain the differences between Stub and Mock objects and demonstrate when it is appropriate to use one or the other. | X | X |
| Explain the benefits and use of the Dependency Injection and other design patterns in TDD | X | X |
| Collect and evaluate information from a number of authoritative sources so as to make an informed decision on the choice of a mocking framework |  | X |

# Section 1 Module Specific Assessment Briefs

# Coursework Brief

**All submissions must have a completed cover sheet (see Appendix A) attached to your submission.**

# Summative Assessment Brief – Part 1, Economic Info TDD Development Project

Percentage of Module: 75%

# **Test Driven Development - TDD Development Project – Solar System Information**

## **Overview**

The project should be completed using the scenario below unless an appropriate actual scenario in the work environment is available. In the case of the latter, the work place scenario must be agreed with your tutor before beginning the assignment.

This project should demonstrate all the learning outcomes of the course.

It is very important in test driven development to only test the code that you are responsible for writing, and not code that your component is dependent on.  With this in mind, clearly demonstrate where (in your submission) any dependencies existed, and critically discuss the different categories of test that can be used in relation to your work.

**Note: As part of the project you are expected to provide documentary evidence that you developed the code in a TDD manner (i.e. screenshots alongside written explanations and comments). This evidence should be submitted as a Word or Open document format, inside a zip file that also contains the project code. Please do not submit PDF type documents.**

### Scenario/context

You have been assigned a development task to create a class called SolarSystemInformation. This class will provide geographic information about the planets, moons, comets and large asteroids located in our solar system.

The final implementation of this class will use an external web service component that will provide information related to the astronomical objects which are identified by a code that specifies the object classification and orbit code. You are to provide the SolarSystemInformation class which uses this web service. The web service is currently in development by another team and will not be available to you during development.

# SolarSystemInformation **class**

The following public methods and properties are required:

## **Constructor**

A constructor is required that will receive the following parameters:

#### **userID**

A string value in the form XX9999 where XX must be two upper case alpha characters and 9999 must be four digits each between 0 and 9. Note, they cannot be four consecutive zeros. Your code should do the validation of this value before calling the web service. If the validation fails or the ID cannot be authenticated by the web service component, the objectName and objectType will be set to “Not allowed” and all other fields will be set to appropriate dummy values (zeroes or spaces).

#### **password**

A string of characters with a combination of upper and lowercase, numbers and symbols, at least 10 characters in length. Your code should do the validation of this value before calling the web service. If the validation fails or the password cannot be authenticated by the web service component, the objectName and objectType will be set to “Not allowed” and all other fields will be set to appropriate dummy values (zeroes or spaces).

## **Fields/getters and setters**

**Field: astronomicalObjectClassificationCode**

Type: String.

Provide a public getter method that returns the official Astronomical Object Classification Code – which should be formatted to the rules given earlier. Provide a private setter.

**Field: objectType**

Type: String.

Provide a public getter method that returns the type of astronomical object (I.E. Star, Planet, Moon, Dwarf Planet, Asteroid or Comet) – which should be Pascal cased. Provide a private setter.

**Field: objectName**

Type: String.

Provide a public getter method that returns the name of the astronomical object – which should be Pascal cased. Note if the object is an asteroid it is quite likely its name will be preceded by a number. Provide a private setter.

**Field: exists**

Type: Boolean.

Provide a public getter method for this, the value tells the client whether the astronomical object classification code is known or not. Provide a private setter.

**Field: orbitalPeriod**

Type: Int.

Provide a public getter that returns the time in (earth) days (to the nearest whole day) it takes the object to make one complete orbit of its host. Provide a private setter

**Field: radius**

Type: BigDecimal.

Provide a public getter method that returns the mean radius of the object n kilometres. Provide a private setter.

**Field: semiMajorAxis**

Type: BigDecimal.

Provide a public getter method that returns the Semi-Major Axis (SMA) of the object in thousands of kilometres. The result must be returned using scientific notation (standard format) and should be rounded to 2 decimal places. For more information about exponential format see <http://www.mas.ncl.ac.uk/~nwhf1/notation.html>

E.G. If the distance is given as 61230000000km. Using scientific notation, it would be 6.1 \* 1010km. Or 6.12E+010km (2dp)

Provide a private setter.

**Field: mass**

Type: BigDecimal.

Provide a public getter method that returns the mass of the object in kilograms. The result must be returned using scientific notation (standard format) and should be rounded to 2 decimal places. Provide a private setter.

## **Methods**

## initialiseAOCDetails()

Provide an **initialiseAOCDetails()** method that takes an **astronomicalObjectClassificationCode as a parameter. The code is a** string type variable that represents an astronomical object’s classification and Semi-Major Axis (SMA) which can only consist of alpha characters and digits. The classification part of the code comes first and is made up of a capital letter that denotes the type of astronomical object (Star, Planet etc.) This is followed by three alpha characters the first of which must be uppercase and the other two lower case lowercase letters or in the case of many asteroids the three alpha characters may be preceded by a number between 1 and 8 digits long.

The Semi-Major Axis (SMA) forms the second part of the code and is made up of between 1 and 3 digits followed by one of the following sets of characters, “T”, “M”, “B”, “L”, “TL” . Examples include:

SSun27TL (Star, Sun, sma 27 thousand light years) \*

PMer58M (Planet, Mercury, sma 58 million km)

PEar150M (Planet, Earth, sma 150 million km)

MMoo384T (Moon, Moon, sma 384 thousand km)

MPho9T (Moon, Phobos, sma 9 thousand km)

DCer416M (Dwarf planet, Ceres, sma 416 million km)

DPlu6B (Dwarf planet, Pluto, sma 6 billion km)

APal416M (Asteroid, Pallas, sma 416 million km)

A3Jun401M (Asteroid, 3 Juno, sma 401 million km)

A99942Apo138M (Asteroid, 99942 Apophis, sma 138 million km)

CHal3B (Comet, Halley, sma 3 billion km)

\* The sun orbits the centre of its galaxy, the Milky Way.

Types of astronomical objects obtainable from the web service are as follows:

|  |  |
| --- | --- |
| S | Star |
| P | Planet |
| M | Moon |
| D | Dwarf Planet |
| A | Asteroid |
| C | Comet |

Distances are shown as:

|  |  |
| --- | --- |
| T | Thousand Kilometres |
| M | Million Kilometres |
| B | Billion Kilometres |
| L | Light-Years |
| TL | Thousand Light-Years |

Note: the sma is short for the semi-major axis which is effectively the average distance of the object from the sun. All astronomical objects’ orbits are elliptical in shape. As an object orbits its host its distance from the host will vary. The closest distance in the orbit is known as the Perihelion and the furthest is the Aphelion. With planets these distances are usually relatively close, but the orbit of some comets takes them very close to the sun and way out beyond the orbit of Pluto. In these cases, the sma isn’t terribly illuminating.

Your code should validate the astronomicalObjectClassificationCode before calling the web service. An exception should be raised if the code is not formatted correctly. If it’s found to be acceptable, the astronomicalObjectClassificationCode will be passed to the web service’s getStatusInfo method, which will return a comma separated list of values.

This initialiseAOCDetails method of the SolarSystemInformation class should also raise an exception if the external web service is unable to provide correctly formatted information. Note, this does **not** include the situation where the web service returns “No such classification or SMA code”. In this situation the SolarSystemInformation class should set the exists field (see below) to false.

**Note:** As a policy you should write defensive logic. I.E. validation should be done in the function where it is mentioned.

## toString()

Provide a ‘**toString()**’ method that returns a string of text formatted like this:

Object Type, Name [AstronomicalObjectClassificationCode] SMA km, mass kg

Where the SMA is given in thousands of kilometres and the mass is in kilograms and both use scientific notation

e.g. Planet, Earth [PEar150M] 150E+003 km, 5.97E+024 kg

**Other Notes**

You may implement the class with any additional private methods you require.

# **Web Service class**

Code for the concrete implementation of the web service component to return live stock information is currently in development.

The web service component will have the following methods:

**authenticate ( String userID, String password )**

Return type is a boolean value.

This method will authenticate the current user. The method must be called before using any other method within the web service.

If the user is authenticated, then the method returns true, otherwise it returns false.

**getStatusInfo ( String astronomicalObjectClassificationCode)**

Return type is a string.

When a valid astronomical object classification code is provided, a comma separated string of values will be returned. The values in this return will be

- Astronomical object classification code

- Object Type (I.E.Star, Planet, Moon, Dwarf Planet, Asteroid or Comet)

- Object Name (E.G. Earth, Mars, Phobos)

- Orbital Period (in earth days to the nearest whole day)

- Radius (in kilometres)

- Semi-Major Axis (in kilometres)

- Mass in kilograms

All numerical values should be greater than or equal to zero.

When an astronomical object classification code does not exist, the method will simply return “No such astronomical object classification code”.

The following link can be used to find examples of astronomical data to use in your tests: <https://en.wikipedia.org/wiki/List_of_Solar_System_objects>

# Submission

The assignment documentation should be zipped into a single folder, then emailed to [bruce.watson@qa.com](mailto:bruce.watson@qa.com), by the deadline.

## Assessment Criteria

* Core class(es) under test code meets specification
* Readability of code in class(es) under test (and elsewhere)
* Efficiency of code in class(es) under test
* Readability of code in Test classes
* Efficiency of code in Test classes. Use of set-up and tear-down methods, stubs and mocks
* Test method names and coverage
* Evidence the development was conducted in a TDD fashion
* Use of Mocking framework(s), handling of expected exceptions and dependency injection

## Marking Rubric

**TDD Final Assignment - TDD Development Project**

| **TDD Final Assignment - TDD Development Project** | | |
| --- | --- | --- |
| **Criteria** | **Ratings** | **Pts** |
| PROJECT: Core class(es) under test code meets specification | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 10.0 Pts  Excellent: The core requirements are fully being met. | 8.0 Pts  Good: The program works and meets most of the core requirements. | 5.0 Pts  Fair: The program delivers on most of the core requirements and produces mostly correct results. | 2.0 Pts  Poor: The program compiles (or almost does) but produces mostly incorrect results. | 0.0 Pts  Unsatisfactory: The code does not compile, and logic is lacking. | | 10.0 pts |
| PROJECT: Readability of code in class(es) under test (and elsewhere) | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 10.0 Pts  Excellent: The code is exceptionally well organized and very easy to follow It is largely self-commenting but and makes good use of comments where necessary. | 8.0 Pts  Good: The code is well organized is easy to read. It is largely self-commenting and mostly makes good use of comments where necessary. | 5.0 Pts  Fair: The logic is mostly followable and readable with limited and/or not very helpful comments. | 2.0 Pts  Poor: The code is readable only by someone who knows what it is supposed to be doing. There are either no comments or very poor comments. | 0.0 Pts  Unsatisfactory: The code is poorly organized and very difficult to read any comments are less than helpful. | | 10.0 pts |
| PROJECT: Efficiency of code in class(es) under test | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 10.0 Pts  Excellent: The code is extremely efficient without sacrificing readability and understanding. It is extremely well refactored with no duplication of code. | 8.0 Pts  Good: The code is pretty efficient without sacrificing readability and understanding. It is well refactored with little or no duplication of code. | 5.0 Pts  Fair: The code is fairly efficient. Refactoring could be better. | 2.0 Pts  Poor: The code is brute force and unnecessarily long. There is little evidence of refactoring. | 0.0 Pts  Unsatisfactory: There is far more code than is necessary. There has been no attempt at refactoring. | | 10.0 pts |
| PROJECT: Readability of code in Test classes | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 10.0 Pts  Excellent: The code is exceptionally well organized and very easy to follow. It is largely self-commenting but and makes good use of comments where necessary. | 8.0 Pts  Good: The code is well organized is easy to read. It is largely self-commenting and mostly makes good use of comments where necessary. | 5.0 Pts  Fair: The logic is mostly followable and readable with limited and/or not very helpful comments. | 2.0 Pts  Poor: The code is readable only by someone who knows what it is supposed to be doing. There are either no comments or very poor comments. | 0.0 Pts  Unsatisfactory: The code is poorly organized and very difficult to read any comments are less than helpful. | | 10.0 pts |
| PROJECT: Efficiency of code in Test classes. Use of set-up and tear-down methods, stubs and mocks. | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 10.0 Pts  Excellent: The code is extremely efficient without sacrificing readability and understanding. It is extremely well refactored with no duplication of code. | 8.0 Pts  Good: The code is pretty efficient without sacrificing readability and understanding. It is well refactored with little or no duplication of code. | 5.0 Pts  Fair: The code is fairly efficient. Refactoring could be better. | 2.0 Pts  Poor: The code is brute force and unnecessarily long. There is little evidence of refactoring. | 0.0 Pts  Unsatisfactory: There is far more code than is necessary. There has been no attempt at refactoring. | | 10.0 pts |
| PROJECT: Test method names and coverage | |  |  | | --- | --- | | 20.0 Pts  Up to 20 marks are available for ensuring the tests are well named and comprehensively test the required functionality | 0.0 Pts  No marks | | 20.0 pts |
| PROJECT: Evidence the development was conducted in a TDD fashion | |  |  |  |  | | --- | --- | --- | --- | | 15.0 Pts  Excellent: Use of the TDD approach was thoroughly demonstrated, via screenshots and annotations | 10.0 Pts  Good: Use of TDD approach was demonstrated but not comprehensively | 5.0 Pts  Fair: A moderate amount of evidence was presented but there may be gaps or a lack of explanation (annotations) | 0.0 Pts  Little or no evidence was presented | | 15.0 pts |
| PROJECT: Use of Mocking framework(s), handling of expected exceptions and dependency injection | |  |  |  |  | | --- | --- | --- | --- | | 15.0 Pts  Excellent: The use of mocking framework was exemplary as was the handling of exceptions and the use of dependency injection. | 10.0 Pts  Good: At least two of the three techniques/tools were used well. An attempt was made to use the other. | 5.0 Pts  Fair: One of the three techniques/tools was used well. Attempts were made at using the two others | 0.0 Pts  Unsatisfactory: None of the tools/techniques were used at an acceptable level | | 15.0 pts |
| Total points: 100.0 | | |

# Section 2 General Assessment Brief Guidance

## How will we support you with your assessment?

You will be briefed on the assessment and any questions can be dealt with by your tutor. All the course content supports you in your assessment. Pay attention to areas of the course highlighted as being of particular value to your assessment.

Feedback is normally provided once by the tutor (**note**: tutors cannot provide feedback on a whole assignment and will provide feedback on up to 1/3rd of an assignment or some aspect of it that runs throughout the work, so discuss with the tutor what part you wish them to focus on).

As a good, if not essential practice, you should reflect on any previous feedback that you have received from completing similar assessments on any other CMDA modules. Make sure you are not repeating same mistakes and building on your strengths. **Paying attention to feedback is the easiest way to gain more from the assessment and to improve your mark.**

## Word Counts

The UoR Teaching and Learning team states: All assignments have a +/- 10% allowance on the wordcount. The University of Roehampton provide the following guidance on exceeding the word count:

* **Up to 10%.** No penalty.
* **Between 11% and 15%.**  A discretionary penalty of no more than 5 marks may be applied, and this will be reflected in the feedback given.
* **Between 16% and 40%.**  A penalty of 10 marks will be applied and this will be reflected in the feedback on the piece of work.
* **Above 40%.** A penalty of 15 marks will be applied and this will be reflected in the feedback on the piece of work.

The penalised mark may reduce the mark below the pass level.

Word count **INCLUDES ANYTHING IN TABLES** and **DOES NOT INCLUDE ANYTHING IN DIAGRAMS** unless they are unnecessarily or unduly wordy. Work included in the appendices is not directly marked but if referred to in the main body of the script is taken into consideration in the awarding of marks.

Referencing: All sources of knowledge used **MUST** be referenced using the Roehampton version of the Harvard System. The Harvard System is very easy to use once you become familiar with it. See here for more details <http://libguides.roehampton.ac.uk/c.php?g=604242&p=4247622>

Written Report Format: The prescribed format for any written work *(excluding any multi-media presentation – for this see specific guidelines as stated in the guidance sections)* is: Times New Roman/Arial, 11pt or 12pt, 1.5 line spacing, Normal Margins (Top and Bottom- 2.54cm, Left and Right2.54cm).

Anything you consider should be included in your mark needs to be in your main script. If these are tables, diagrams or multi-media links consider how much time they will take to absorb by the reader.

Excessive use of extra material within the word count or equivalent will be penalised and feedback given explaining the penalty in line with Roehampton University policy.

Ensure any diagrams, tables etc. that are within the script are given a title and referred to in the body of your assignment. Do not expect the reader to understand how they fit into your argument unless you explain how they do. Tables are included in the word count. Diagrams are not included but will be considered if they appear to be a way of avoiding word count by being very wordy diagrams. Please be careful here. If you are in any doubt about what counts as word count consult your tutor and/or stay on the side of caution.

## Late submissions, incorrect submissions or failure to submit

In line with Roehampton academic regulations (which can be downloaded from this [link](https://www.roehampton.ac.uk/corporate-information/quality-and-standards/academic-regulations/)), a penalty will be imposed on work submitted after the deadline, or after the revised deadline in the case of a learner who has been granted an extension under the provisions of Section 16, as follows.

1. Where the learner submits work up to and including 14.00, seven calendar days after the deadline, the percentage mark for the component of assessment will be capped at 40% for all undergraduate modules at Levels 4–6.
2. Where the learner submits work after 14.00, seven calendar days after the deadline, the percentage mark for the component of assessment will be set to zero.

Any assignment (either a part of assignment or whole of assignment) that is not submitted correctly will not be marked and will be awarded 0% (zero).   The tutor will not inform learners that they need to resubmit correctly. No feedback will be provided on these assignments (or parts of).

Your deadline will be set to give sufficient time for the Independent Assessor to review the project report before the presentation takes place.

**It is the responsibility of the learner to ensure they can submit to the correct location and on time.**