IT7320 – Software Development and Testing

Assignment 3

Mockito

*Compiled September 2018*

*Aleisha Perry - 2115039*

For

*Chalinor Baliuag*

*Due Date: Week 10, Trimester 2, 2018*

# 

Contents

[Part 1 2](#_Toc525135858)

[Using Mockito 2](#_Toc525135859)

[Class or Interface 3](#_Toc525135860)

[Appendix A 4](#_Toc525135861)

[GitHub 4](#_Toc525135862)

Table of Figures

[Figure 1 - Implementation class code for computeTax() method 2](#_Toc525135863)

[Figure 2 - Test runner shows red without a mock object in test case 2](#_Toc525135864)

[Figure 3 - Implementing a mock object for the interface 2](#_Toc525135865)

[Figure 4 - Test runner shows green with a mock object in test case 3](#_Toc525135866)

# Part 1

## Using Mockito

Running a JUnit test case without having created a mock object in place of the interface will fail, returning a NullPointerException. This is due to the fact that, in the implementation class, the method’s return is a function call from the interface which holds no code for how the method works. The code can be seen below:

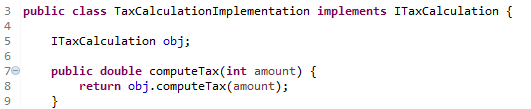


Figure 1 - Implementation class code for computeTax() method

The above image shows the code for the computeTax() method that returns the interface method.

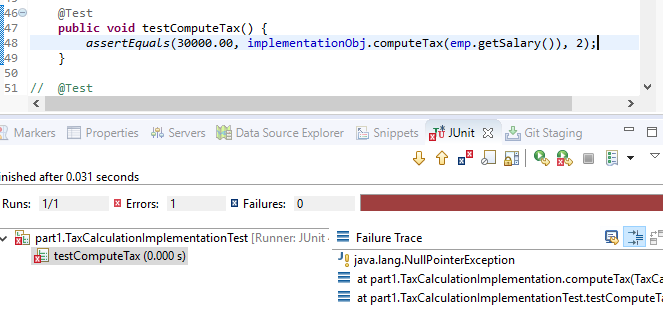


Figure 2 - Test runner shows red without a mock object in test case

The above image shows the test runner as a fail after trying to run the test without having created a mock object.

Implementing an interface into a mock object creates an object that will take the place of the real object, removing the dependencies of the real object as they can be difficult to setup, have unpredictable results, behaviour that is difficult to activate, and it could also be an interface. The code for this can be seen below:

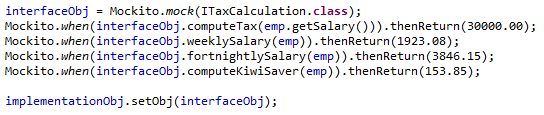


Figure 3 - Implementing a mock object for the interface

After setting up the test cases with an assertEquals comparison between expected results and actual results, the test will now return as a pass, due to the mock object being referenced instead of the interface.

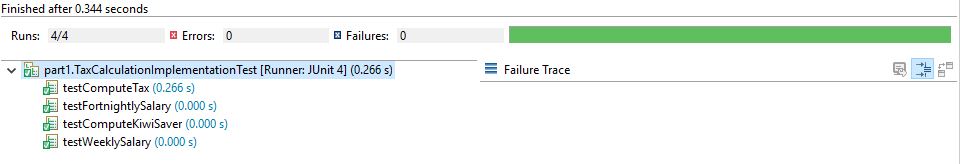


Figure 4 - Test runner shows green with a mock object in test case

## Class or Interface

I chose to create a class for this part of the assignment for several reasons, including:

1. Classes have greater flexibility than interfaces and abstract classes,
2. I understand at a confident level how classes work,
3. I haven’t attempted JUnit testing with interfaces or abstract classes, so there is uncertainty with proceeding with them, and
4. I know that a class can be tested by creating an object of that class.

# Appendix A

## GitHub

GitHub was used for the duration of this assignment in order to submit assignment in stages (Parts 1 - 4).

**Profile - Aleisha**

Remote Repository main branch (owner) link: <https://github.com/kiwiFudge/IT7320_Assignment3_Part1>