# SIT323 Practical Software Development, Trimester 2, 2020

## Week 4 – Practical 3

## Unit Testing

## Introduction

This practical will help you:

1. design unit tests (without writing any code)
2. implement unit tests using Visual Studio

## Task 1 – Unit Testing

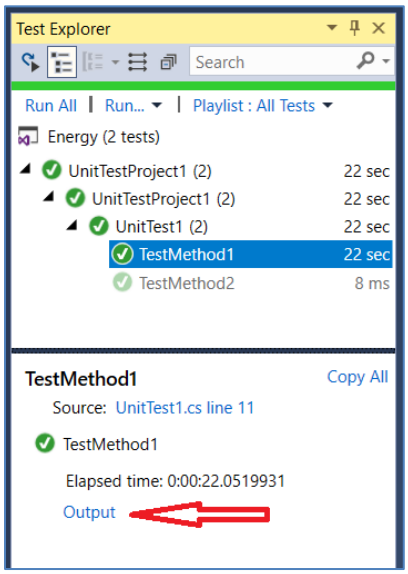
A template to help design and report results of several unit tests can be found in the file called **Week 4 - Unit Test Design Template.docx**. Examples of several unit test designs can be found in the file called **Week 4 - Unit Test Design Examples.docx**.

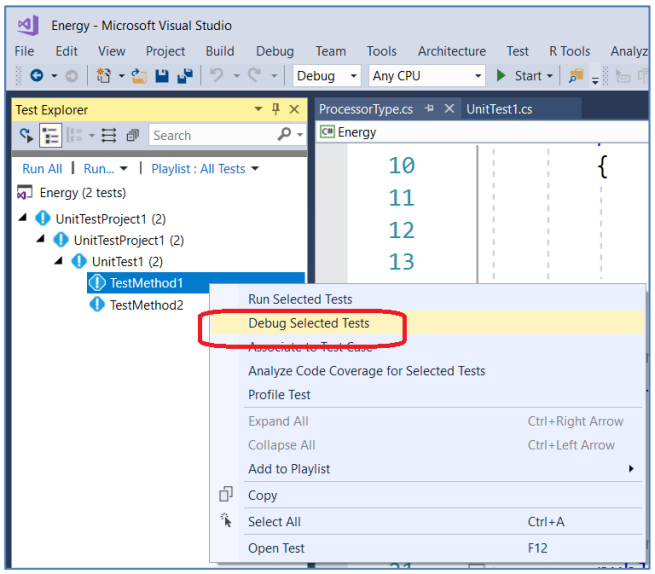
The ZIP file **Class 4 - Unit Testing.zip** contains C# code of 2 test methods, these were presented during lecture of Week 3.

Your tasks are to:

1. download and unzip **Class 4 - Unit Testing.zip**
2. open **Energy.sln** using VS 2017
3. open **Test Explorer** in VS 2017
4. run both unit tests to obtain values and fill in both **Actual Data** and **Test Result** portions for Test Case ID = 1.1 and Test Case ID = 2.1 of the Test Scenario tables found in **Week 4 - Unit Test Design Examples.docx**

There are several ways to obtain the values for Actual Data and Test Results such as including one or more Console.WriteLine() statements in unit tests. But this doesn't send data to the normal output window, instead you need to access the 'Output' link of Test Explorer. In addition, you can place breakpoints in a unit test and debug by selecting **Debug Selected Tests**.



## Task 2 – Unit Test Design

You can design a unit test by partially completing a template. During design, you need not complete the following fields: **Test Method**, **Actual Data**, **Test Result** and **Test Comment**. These four fields can be blank during design

A design can then be given to a programmer to write one or more test methods to implement the unit test(s). These test methods can then be run using Test Explorer in order to fill in these four blank fields.

For example, the following is a design for testing the EnergyPerSecond() method of the ProcessorType class in the namespace Energy.

Your task is to add 3 more test cases (1.3, 1.4, and 1.5) to the following test scenario by using different values of frequencies and coefficients, and determining the expected EPS values.

| **Test Scenario ID** | 1 | | | | | |
|---|---|---|---|---|---|---|
| **Test Description** | Check that the EnergyPerSecond method correctly performs. | | | | | |
| **Method Tested** | Public double Energy.ProcessorType.EnergyPerSecond(double frequency) | | | | | |
| **Test Case ID** | **Test Method** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 1.1 | | frequency = 1<br>c2 = 10<br>c1 = -25<br>c0 = 25 | expectedEPS = 10 | | | |
| 1.2 | | frequency = 2<br>c2 = 10<br>c1 = -25<br>c0 = 25 | expectedEPS = 15 | | | |
| 1.3 | | | | | | |
| 1.4 | | | | | | |
| 1.5 | | | | | | |

## Task 3 – Unit Test Implementation

Use the code in **Class 4 – Unit Testing.zip** to:

- Implement 4 new test methods for the 4 test cases 1.2 to 1.5 from the design of Task 2.
- Run your test methods using Test Explorer.
- Complete your unit test design table by filling in the blank fields.

## Task 4 – Unit Test Design

Your task is to create a unit test design for the C# **Trim()** method. This method removes all leading and trailing **white spaces** from a string object.

Your test cases should be based on:

1. no leading or trailing white spaces
2. leading white space but no trailing white spaces
3. trailing white space but no leading white spaces
4. both leading and trailing white spaces
5. different kinds of white spaces

## Task 5 – Unit Test Implementation

Use the code in **Class 4 – Unit Testing.zip** to:

- Implement one or more test methods for your test design of Task 4.
- Run your test methods using Test Explorer.
- Complete the unit test design table by filling in the blank fields.

---