**Objectives**

* Demonstration on parameterized test cases using an example
* Demonstrate on testing methods that return a value.
  + Use Assert.AreEqual, Assert.Fail, AssertionException
* Demonstrate on testing void methods
* Demonstrate on testing methods that throw an exception
  + Use try catch to catch exception and check the exception message and type
* Explain on why testing a private method is not beneficial.
* Explain mocking framework and its usage
  + Dependencies like database connections, file objects should be mocked, Moq framework

Please download the application available [here](https://cognizantonline.sharepoint.com/:u:/r/sites/GTP-Solutions/Gencsharepath/Shared%20Documents/Internship2020/FSE/DotNet/02%20-%20NUnit,%20C%23%204.5,%20ASP.Net%20Core/Handson/CalcLibrary.zip?csf=1&e=aLxB66). This will be used to write Unit test cases.

**Parameterized test cases**

1. Create test case to verify the subtraction feature of the calculator with various input types.

* Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.
* Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.
* Use Assert.Equal to check the actual and expected results

1. Create a test case to verify the multiplication concepts of calculator

* Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.
* Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.
* Use Assert.Equal to check the actual and expected results

1. Create a test case to verify the division logic of the calculator

* Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.
* Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.
* Use Assert.Equal to check the actual and expected results
* In one of the inputs, provide the divisor value to be 0
  + Use Try Catch block to catch the ArgumentException
  + Use Assert.Fail to notify the user that the test case has failed. Give the message “Division by zero” in the Assert.Fail, which will be notified to the user. This message will be seen in the test explorer.

**Test void methods**

In the MathLibrary class there is a property “GetResult”. The result of every operation is stored in a variable ‘result’. This value is accessed by the property.

The class also has a method “AllClear” that sets the value of the result variable to 0.

* Create a test method ‘TestAddAndClear’
* Invoke the Addition method of the math class library
* Verify if the expected and Actual results match using Assert.AreEqual
* Invoke the ‘AllClear’ method
* Use Assert.AreEqual to check if the result is 0 or not