# Workshop

# Custom Unit Testing Framework

### Overview

In this workshop, you need to build a custom unit testing framework on your own, following the basic principles of OOP. Use it to write tests for the provided class.

### Setup

You are provided with a **skeleton**,which contains the following items:

* **CustomUnitTesting** project– it contains the following folders:
  + **Asserts**
  + **Attributes**
  + **Exceptions**
  + **TestRunner**
  + **Utilities**
* **CustomUnitTesting.Tests** project – it contains a public class **MyTestClass**
* **MyProgram** project– it contains a **StartUp** class with a written logic inside it. Its purpose is to give you info about the passed and failed tests.

# Structure

## CustomUnitTestingFramework

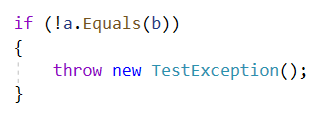
### Asserts

In the **Asserts** folder in the **CustomUnitTesting** project create the following classes:

#### Assert

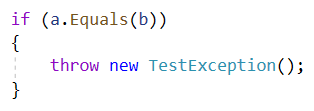
It’s a public static class and it needs to hold the Assert methods, which you are going to use when you write tests later.

Write **public static void** **AreEqual** method, which accepts parameters: object a, object b. It should compare if two objects are equal. Here is a logic you can use for it:

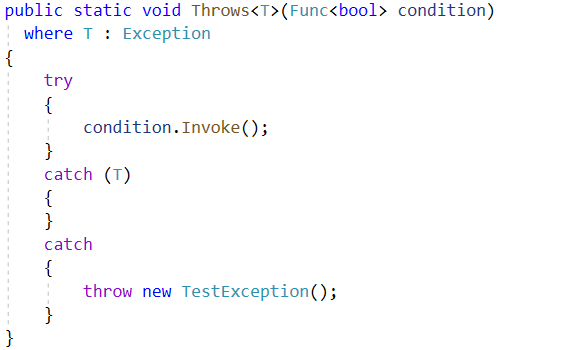


You are going to create the **TestException** class later.

Write **public static void** **AreNotEqual** method, which checks if two objects are different. Again, it accepts the same parameters as the previous method. You can use the following logic for it:



Write public static void Throws<T>(Func<bool> condition) method, which makes sure that a certain condition throws an exception. You can use the following logic:



### Attributes

In the **Attributes** folder in the **CustomUnitTesting** project create the following classes:

#### TestClassAttribute

This is a public class, which inherits **Attribute**

#### TestMethodAttribute

This is also a public class, which inherits **Attribute**

### Exceptions

In the **Exceptions** folder in the **CustomUnitTesting** project create the following class:

#### TestException

A public class, which inherits the **Exception** class.

### TestRunner

In the **TestRunner** folder in the **CustomUnitTesting** project create the following class:

#### TestRunner

A public class, which must implement the given interface in the **Contracts** subfolder.

The **TestRunner** class holds the functionality that runs the tests. The class must hold a private readonly ICollection<string> resultInfo. Don't forget to initialize the collection in the constructor. Inside the Run method, you need to get all the classes, which have the **TestClassAttribute.** Then you need to run the tests for all the methods inside those classes that have the **TestMethodAttribute**. You can use the following logic:



The method must return the **resultInfo** collection.

### Utilities

In the **Utilities** folder in the **CustomUnitTesting** project create the following class:

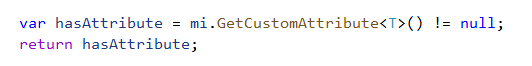
#### ReflectionHelper

This is a **public static class**, which checks if a given class / method has an attribute. It holds a single method:

public static bool HasAttribute<T>(this MemberInfo mi)

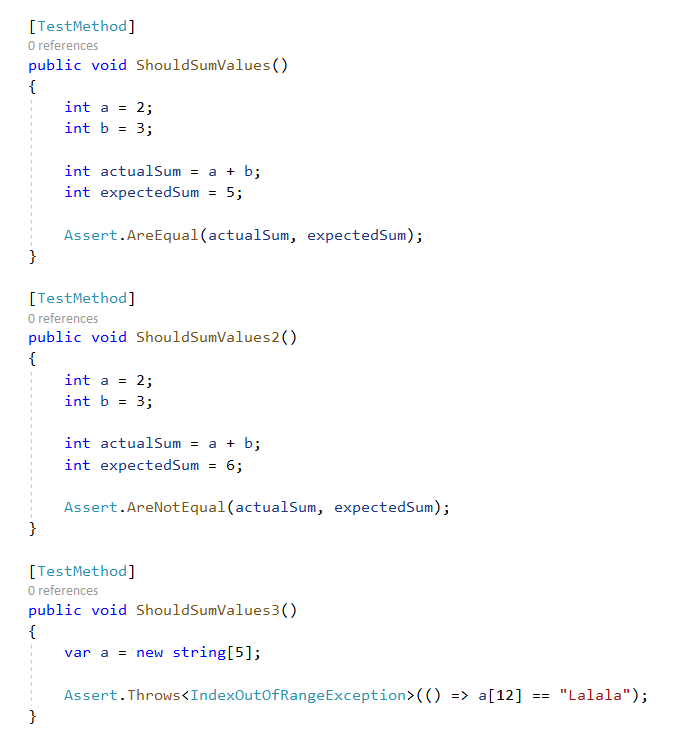
where T : Attribute

You can use the following logic for it:



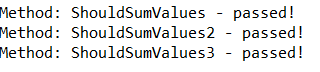
## CustomUnitTestingFramework.Tests

In the **CustomUnitTestingFramework.Tests** project, create a class MyTestClass, which has **TestClassAttribute** and all the methods in it are with a **TestMethodAttribute**. You need to have three methods, that use the methods from your custom testing framework. They should look like this:



## MyProgram

In the **MyProgram** project you are given a **StartUp** class, with a written logic inside it. Try to run your program and don't forget to set MyProgram as a start up project. You need to receive the following output on the console:



Congratulations! Now you can try to right your own test class using your custom framework!