**[TestFixture]**

* The class that is going to contain your code for testing should be attributed with this attribute
* When you attach this attribute to a class in your project, the Test Runner application will scan it for test methods

**[Test]**

To mark a method as a test method

**[SetUp]**

Any code that must be executed prior to executing a test, can be put in the functions marked with [Setup]. This frees you from repeating these lines of code in each test. Please note carefully, **that this code is executed, before each test.**

**[TearDown]**

This attribute is exactly opposite to Setup Attribute. **This code is executed after execution of each code**. In this method, for example, you can write code, to close any FileSystem Object or Database connection.

**[ExpectedException]**

At times, we may want **to test that a method should throw an exception in a particular scenario**. We can test this using this attribute.

**[Ignore] or [Ignore(“Reason”)]**

To ignore the test case

**[TestCase(2240)]**

Marks a method as parametrized test suit and provides arguments for each test case.

public void PromostandardDataServiceTest(int companyId, TestCaseConfig config)

{ }

**Mocking**

[TestCase(100, "Declined")]

[TestCase(549, "Declined")]

[TestCase(550, "Maybe")]

[TestCase(674, "Maybe")]

[TestCase(675, "We look forward to doing business with you!")]

public void MakeCreditDecision\_Always\_ReturnsExpectedResult(int creditScore, string expectedResult)

{

**mockCreditDecisionService = new Mock<ICreditDecisionService>(MockBehavior.Strict);**

**mockCreditDecisionService.Setup(p => p.GetDecision(creditScore)).Returns(expectedResult);**

systemUnderTest = new CreditDecision(mockCreditDecisionService.Object);

var result = systemUnderTest.MakeCreditDecision(creditScore);

Assert.That(result, Is.EqualTo(expectedResult));

mockCreditDecisionService.VerifyAll();

}

We'll cover the MockBehavior.Strict bit in a minute, but let's focus on that second line - this is where we configure the mock for this test. In this case, we're telling the mock, "Hey Mock, if your GetDecision method is invoked with this specific number (creditScore), return this response (expectedResult). If it gets invoked with any other number, fail the test immediately". (that's part of MockBehavior.Strict)

Next up, we execute the MakeCreditDecision method just like we did before, the only remaining step is to ask our Mock instance if all of its expectations were fulfilled using mockCreditDecisionService.VerifyAll()