**Topic 5: Introduction to Unit Testing**

Top of Form

Bottom of Form

**Content**

**Reading**

[Reading](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286951_1)

Unit Testing

* + Microsoft:docs.microsoft.com
    - [Unit test basics](https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019)
    - [Getting started with unit testing](https://docs.microsoft.com/en-us/visualstudio/test/getting-started-with-unit-testing?view=vs-2019)
    - [Unit Testing C# in .NET Core using dotnet test and xUnit](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test)
    - [Create and run unit tests for managed code](https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019)
    - [Best practices for unit testing](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices)
  + GeeksForGeeks: geeksforgeeks.com
    - [Unit Testing/Software Testing](https://www.geeksforgeeks.org/unit-testing-software-testing/)
  + TutorialsPoint: tutorialspoint.com
    - [Unit Testing](https://www.tutorialspoint.com/software_testing_dictionary/unit_testing.htm)
    - [Unit Testing for C# Code](https://www.tutorialspoint.com/Unit-Testing-for-Chash-Code)
  + C# Corner: c-sharpcorner.com
    - [C# Corner A Basic Introduction to C# Unit Test for Beginners](https://www.c-sharpcorner.com/article/a-basic-introduction-of-unit-test-for-beginners/)

**Unit Testing**

[Unit Testing](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286951_1)

So for you have learned that testing your code is important. You manually tested for cases that would produce expected results and cases that would cause an exception to be thrown.

Manual testing can get laborious and tedious. You may forget some cases. You (or another programmer) may change your code at a future date that breaks the code. Perhaps you forgot all the test cases or the other programmer never new the test cases. A solution to both the laborious testing and having maintainable code is to incorporate Unit Tests.

Unit Test are small pieces of code containing tests that automatically test code for determined cases of expected and unexpected behaviors.

To build and help organize these tests, there are [Unit Testing Frameworks](https://stackify.com/unit-test-frameworks-csharp/).

In C# there are many options for Unit Testing. In some languages there is one predominant Unit Testing Frameworks (Java has JUnit) and other languages there are many options to choose from (C++ has several). C# has a few major testing frameworks.

* + MSTest--Microsoft Testing Tool began as a command tester, it is already integrated in your IDE
  + [NUnit](https://nunit.org/)--This was ported from JUnit, adapted to C# and is now for many .NET platforms (requires download)
  + [xUnit](https://xunit.net/)--xUnit is an open source tool for testing, for C#, F#, VB.NET among other .NET languages (requires download)

You will learn xUnit for this course, but you might get a job as C# programmer with a company that uses one of the others, or even a mixture.

There is even a way to program where you write the tests first, then the code. This is called Test-Driven Development. Even if you write the Unit Test after you write your code, you can use it to find errors in your code. The idea of Unit Testing is to think of all the cases that might happen and include a test for each.

**Testing Setup in Visual Studio**

[Testing Setup in Visual Studio](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286951_1)

First you will need to set up a project in your IDE to test your code. A Visual Studio solution is a container that can hold one or more projects. You will want to keep your unit tests organized in a separate project for now. See the video below for adding a project to an existing solution for testing.

https://youtu.be/WgFKV8ktf0w

**Writing Unit Tests**

[Writing Unit Tests](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286951_1)

When you write a Unit Test, it has 3 basic parts, in the following order:

* + ARRANGE: Setup any variables for input, actual output, expected output and any other state setting variables.
  + ACT: perform tests to validate
  + ASSERT: use the framework assertion to compare actual output and expected output.

Example:

https://youtu.be/K4uf\_nEV\_8g

[Fact]

public void PassingCaseTest()

{

// Arrange

string fName = "Megan";

string lName = "RAPINOE";

string rank = "1";

string expected = "Megan RAPINOE 1";

string actual;

// Act

actual = MedalWinnerProgram.MedalWinner(fName, lName, rank);

// Assert

Assert.Equal(expected, actual);

}

When you run your Unit Test, there four general types of results:

* + Passed--there were no exception thrown or no error
  + Failed-- an exception was thrown or error
  + Ignore-- the unit test was skipped
  + Inconclusive-- the test was not able to be run with the current available data)

As you begin, you will see the first 3 types of results. You want all your tests to pass.

**Example Unit Testing**

[Example Unit Testing](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286951_1)

**MedalWinner.cs**

using System;

namespace MedalWinner

{

public class MedalWinnerProgram

{

public static string MedalWinner(string fName, string lName, string rank)

{

string result = null;

result = fName + " " + lName + " " + Convert.ToInt32(rank).ToString();

return result;

}

static void Main(string[] args)

{

// Call a method that returns the user's first name first letter capitalized only

// and the last name all capitalized followed by rank

string winner1 = MedalWinner("Megan", "rapinoE", "1");

string winner2 = MedalWinner("USA", "Women's Soccer","1");

string winner3 = MedalWinner("USA", "Women's Soccer", "1.5"); // DO NOT CHANGE THIS LINE

Console.WriteLine("And the winner is ... {0:G}", winner1);

Console.WriteLine("Expected: Megan RAPINOE 1");

Console.WriteLine("And the winner is ... {0:G}", winner2);

Console.WriteLine("Expected: Usa WOMEN'S SOCCER 1");

Console.WriteLine("And the winner is ... {0:G}", winner3);

Console.WriteLine("Expected: Usa WOMEN'S SOCCER 1.5");

}

}

}

**UnitTest1.cs**

using MedalWinner;

using Microsoft.VisualStudio.TestPlatform.TestHost;

using System;

using Xunit;

namespace IntroUnitTestingTests

{

public class UnitTest1

{

[Fact]

public void PassingCaseTest()

{

// Arrange

string fName = "Megan";

string lName = "RAPINOE";

string rank = "1";

string expected = "Megan RAPINOE 1";

string actual;

// Act

actual = MedalWinnerProgram.MedalWinner(fName, lName, rank);

// Assert

Assert.Equal(expected, actual);

}

[Fact]

public void FailingCaseTest()

{

// Arrange

string fName = "Megan";

string lName = "rapinoe";

string rank = "1";

string expected = "Megan RAPINOE 1";

string actual;

// Act

actual = MedalWinnerProgram.MedalWinner(fName, lName, rank);

// Assert

Assert.Equal(expected, actual);

}

}

}

[**UnitTest Debugging**](https://dmacc.blackboard.com/webapps/assignment/uploadAssignment?content_id=_7286978_1&course_id=_102593_1&group_id=&mode=view)

Complete the attached tutorial. DO NOT CHANGE Unit Tests, only edit the methods in the Program.cs file. Submit a screen shot that shows the method MedalWinner and the passing Unit Tests. Name the screenshot file PassingTestsLastName. Also, zip the entire project, call it UnitTestDebuggingLastName.zip. Do not place your screen shot in the zip file, attach the two separately.

[C# First Unit Test.pdf](https://dmacc.blackboard.com/bbcswebdav/pid-7286978-dt-content-rid-101466144_1/xid-101466144_1) [C# First Unit Test.pdf - Alternative Formats](https://dmacc.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_102593_1&content_id=_7286951_1)