using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

/// <summary>

/// File name: PTangAssignment2/TriangleSolverTest.cs

///

/// Purpose: Create a C# console application determining the correct type of a triangle, then use Git Bash

/// for version control, NUnit for unit testing, then explain the Control Flow Graph and Cyclomatic

/// Complexity in a word document

///

/// Created by Patrick Tang

///

/// History:

/// February 14, 2017 - Created

/// February 22, 2017 - Added code for unit tests

/// - Added comments and finished program

/// </summary>

namespace PTangAssignment2.Tests

{

[TestFixture]

public class TriangleSolverTest

{

/// <summary>

/// Unit Testing: Tests to make sure an equilateral triangle is valid

/// Inputs: 5, 5, 5

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsFiveFiveFive\_ReturnsEquilateral()

{

Assert.AreEqual(TriangleSolver.Analyze(5, 5, 5), "This is an Equilateral triangle");

}

/// <summary>

/// Unit Testing: Tests to make sure an isosceles triangle is valid

/// Inputs: 2, 3, 2

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsTwoThreeTwo\_ReturnsIsosceles()

{

Assert.AreEqual(TriangleSolver.Analyze(2, 3, 2), "This is an Isosceles triangle");

}

/// <summary>

/// Unit Testing: Tests to make sure an scalene triangle is valid

/// Inputs: 2, 1, 4

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsTwoFourThree\_ReturnsScalene()

{

Assert.AreEqual(TriangleSolver.Analyze(2, 4, 3), "This is a Scalene triangle");

}

/// <summary>

/// Unit Testing: Tests to make sure a triangle is invalid

/// Inputs: 3, 5, 1

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsThreeFiveOne\_ReturnsInvalid()

{

Assert.AreEqual(TriangleSolver.Analyze(3, 5, 1), "This does not form any valid triangles");

}

/// <summary>

/// Unit Testing: Tests to make sure a negative triangle is invalid

/// Inputs: -3, -4, -2

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsNegatives\_ReturnsInvalid()

{

Assert.AreEqual(TriangleSolver.Analyze(-3, -4, -2), "This does not form any valid triangles");

}

/// <summary>

/// Unit Testing: Tests to make sure that any zeroes is a definitely invalid

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsZeroZeroZero\_ReturnsInvalid()

{

Assert.AreEqual(TriangleSolver.Analyze(0, 0, 0), "No triangles can be formed with any zeroes");

}

/// <summary>

/// Unit Testing: Tests to make sure that one negative input is an invalid triangle

/// Inputs: 5, -2, 9

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsOneNegative\_ReturnsInvalid()

{

Assert.AreEqual(TriangleSolver.Analyze(5, -2, 9), "This does not form any valid triangles");

}

/// <summary>

/// Unit Testing: Tests to make sure big numbers can be valid triangle

/// Inputs: 155, 525, 451

/// </summary>

[TestCase]

public void TestingTriangleSides\_InputsBigNumbers\_ReturnsScalene()

{

Assert.AreEqual(TriangleSolver.Analyze(155, 525, 451), "This is a Scalene triangle");

}

}

}