Keval Vishalbhai Patel

9009797

Assignment #2

10-11-2024

Source Code

```
using NUnit.Framework;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TriangleSolver;

namespace TraingleSolverClassLibrary
{
    [TestFixture]
    public class Class1
    {

        // Equilateral Triangle Test
        [Test]
        public void
ValidEquilateralTriangle_Input10and10and10_OutputEquilateraltriangle()
        {
            // Arrange
            int side1 = 10;
            int side2 = 10;
            int side3 = 10;
            string expected = "Equilateral triangle";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));

        }

        // Isosceles Triangle Test 1
        [Test]
        public void
ValidIsoscelesTriangle_Input2and2and3_OutputIsoscelesTriangle()
        {
            // Arrange
            int side1 = 2;
            int side2 = 2;
            int side3 = 3;
            string expected = "Isosceles triangle";
```

```
            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Isosceles Triangle Test 2
        [Test]
        public void
ValidIsoscelesTriangle_Input4and5and4_OutputIsoscelesTriangle()
        {
            // Arrange
            int side1 = 4;
            int side2 = 5;
            int side3 = 4;
            string expected = "Isosceles triangle";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Isosceles Triangle Test 3
        [Test]
        public void
ValidIsoscelesTriangle_Input7and10and10_OutputIsoscelesTriangle()
        {
            // Arrange
            int side1 = 7;
            int side2 = 10;
            int side3 = 10;
            string expected = "Isosceles triangle";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Scalene Triangle Test 1
```

```
        [Test]
        public void
ValidScaleneTriangle_Input300and400and500_OutputScaleneTriangle()
        {
            // Arrange
            int side1 = 300;
            int side2 = 400;
            int side3 = 500;
            string expected = "Scalene triangle";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Scalene Triangle Test 2
        [Test]
        public void
ValidScaleneTriangle_Input700and500and900_OutputScaleneTriangle()
        {
            // Arrange
            int side1 = 700;
            int side2 = 500;
            int side3 = 900;
            string expected = "Scalene triangle";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Scalene Triangle Test 3
        [Test]
        public void
ValidScaleneTriangle_Input600and800and1000_OutputScaleneTriangle()
        {
            // Arrange
            int side1 = 600;
            int side2 = 800;
            int side3 = 1000;
            string expected = "Scalene triangle";
```

```
            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Scalene Triangle Test 4
        [Test]
        public void
ValidScaleneTriangle_Input1100and1300and1700_OutputScaleneTriangle()
        {
            // Arrange
            int side1 = 1100;
            int side2 = 1300;
            int side3 = 1700;
            string expected = "Scalene triangle";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Scalene Triangle Test 5
        [Test]
        public void
ValidScaleneTriangle_Input800and1500and1700_OutputScaleneTriangle()
        {
            // Arrange
            int side1 = 800;
            int side2 = 1500;
            int side3 = 1700;
            string expected = "Scalene triangle";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Zero lenght side Test 1
```

```
        [Test]
        public void ZeroLengthSide_Input0and100and200_OutputInvalidTriangle()
        {
            // Arrange
            int side1 = 0;
            int side2 = 100;
            int side3 = 200;
            string expected = "Invalid Triangle - a zero has been detected";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Zero lenght side Test 2
        [Test]
        public void ZeroLengthSide_Input300and0and400_OutputInvalidTriangle()
        {
            // Arrange
            int side1 = 300;
            int side2 = 0;
            int side3 = 400;
            string expected = "Invalid Triangle - a zero has been detected";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }

        // Zero lenght side Test 3
        [Test]
        public void ZeroLengthSide_Input500and600and0_OutputInvalidTriangle()
        {
            // Arrange
            int side1 = 500;
            int side2 = 600;
            int side3 = 0;
            string expected = "Invalid Triangle - a zero has been detected";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);
```

```
        // Assert
        Assert.That(expected, Is.EqualTo(result));
    }

    // Invalid Response Test 1
    [Test]
    public void InvalidTriangle_Input10and2and7_OutputInvalidTriangle()
    {
        // Arrange
        int side1 = 10;
        int side2 = 2;
        int side3 = 7;
        string expected = "INVALID!!";

        // Act
        string result = Triangle.AnalyzeTriangle(side1, side2, side3);

        // Assert
        Assert.That(expected, Is.EqualTo(result));
    }

    // Invalid Response Test 2
    [Test]
    public void InvalidTriangle_Input15and5and9_OutputInvalidTriangle()
    {
        // Arrange
        int side1 = 15;
        int side2 = 5;
        int side3 = 9;
        string expected = "INVALID!!";

        // Act
        string result = Triangle.AnalyzeTriangle(side1, side2, side3);

        // Assert
        Assert.That(expected, Is.EqualTo(result));
    }

    // Invalid Response Test 3
    [Test]
    public void InvalidTriangle_Input25and10and5_OutputInvalidTriangle()
    {
        // Arrange
        int side1 = 25;
```

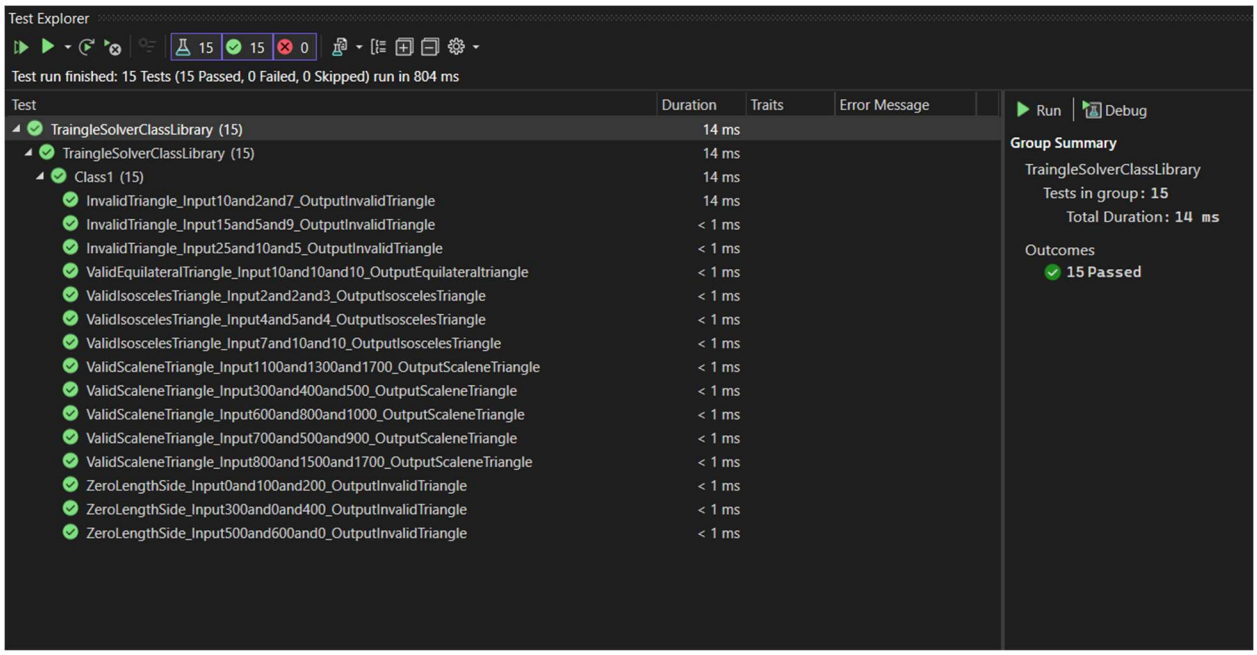
```
            int side2 = 10;
            int side3 = 5;
            string expected = "INVALID!!";

            // Act
            string result = Triangle.AnalyzeTriangle(side1, side2, side3);

            // Assert
            Assert.That(expected, Is.EqualTo(result));
        }


    }
}
```

Test Explorer Screenshot



Git Log Screenshot