Lovepreet Singh

8319519

Assignment #2

11/02/2019

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Assignment #2: Rubric#** | **Mark** | | **Weight** | | **Criteria** |
| **TriangleSolver Class** | | | | | |
| **1.** | | 1 | | TriangleSolver class created as separate file. | |
| **2.** | | 1 | | TriangleSovler class is static is static. | |
| **3.** | | 1 | | Analyze method is static | |
| **4.** | | 4 | | Analyze method works as described. | |
| **Console Application** | | | | | |
| **5.** | | 1 | | Menu is presented as described. | |
| **6.** | | 1 | | If any incorrect input is given, the menu is shown again. | |
| **7.** | | 2 | | The program correct reports if the numbers form a triangle, yes or no. | |
| **8.** | | 4 | | If the numbers do form a triangle, the program reports if that triangle  is scalene, isosceles, or equilateral. | |
| **9.** | | 1 | | Menu Option 2 exits the program. | |
| **Unit Tests** | | | | | |
| **10.** | | 24 | | Eight unique Unit tests for Analyze() Method. | |
| **11.** | | 8 | | Explanation for the eight unit tests. | |
| **12.** | | 8 | | Unit test naming conventions. | |
| **13.** | | 1 | | Screenshot of completed unit tests run successfully. | |
| **Version Control** | | | | | |
| **14.** | | 3 | | Screenshot showing repository with a commit history. | |
| **Control Flow Graph** | | | | | |
| **15.** | | 9 | | Graph complete, matches code, includes line numbers that match developed code. | |
| **16.** | | 4 | | Cyclomatic Complexity calculated, work to calculate it is shown. | |
| **17.** | | 2 | | Brief discussion on the meaning of Cyclomatic Complexity value that  was calculated. | |
| -0.50 per | | | **Programming standards deductions.** | | |
| -15 | | | **Failure to present deduction.** | | |
| -7.5 | | | **Hard or soft copy not handed in or not all documents listed handed in.** | | |
| **75** | | | **Total Marks** | | |

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace QATestingAssignment02

{

class Program

{

static void Main(string[] args)

{

int option, side1, side2, side3; ;

do

{

Console.WriteLine("Select One of the following: ");

Console.WriteLine("1. To the Dimensions of Triangle : ");

Console.WriteLine("2. EXIT");

Console.WriteLine("Enter your choice :");

option = Convert.ToInt32(Console.ReadLine());

} while (option != 1 && option != 2);

do

{

if (option == 2)

break;

if (option == 1)

{

do

{

Console.WriteLine("The First side of the triangle: ");

side1 = Convert.ToInt32(Console.ReadLine());

} while (side1 <= 0);

do

{

Console.WriteLine("The Second side of the triangle: ");

side2 = Convert.ToInt32(Console.ReadLine());

} while (side2 <= 0);

do

{

Console.WriteLine("The Third side of the triangle");

side3 = Convert.ToInt32(Console.ReadLine());

} while (side3 <= 0);

TriangleSolver obj = new TriangleSolver();

string message = obj.Analyze(side1, side2, side3);

Console.WriteLine(message);

Console.WriteLine();

Console.WriteLine();

Console.WriteLine("1. To the Dimensions of Triangle : ");

Console.WriteLine("2. EXIT");

Console.WriteLine("Enter your choice :");

option = Convert.ToInt32(Console.ReadLine());

}

} while (option == 1 || option == 2);

}

}

}

TraingleSolverTests.cs

using QATestingAssignment02;

using System;

using NUnit.Framework;

namespace QATestingAssignment02.Tests

{

[TestFixture]

public class TriangleSolverTests

{

[Test]

public void Sides3and3and3forEquilateral()

{

//Arrange

int s1 = 3, s2 = 3, s3 = 3;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "It's Equilateral");

Console.WriteLine("Please check the result");

}

[Test]

public void Sides3and2and2forIsosceles()

{

//Arrange

int s1 = 3, s2 = 2, s3 = 2;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "It's Isosceles");

Console.WriteLine("Please check the result");

}

[Test]

public void Sides0and0and0forNOTPOSSIBLE()

{

//Arrange

int s1 = 0, s2 = 0, s3 = 0;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "All sides enterred are zero, Triangle not possible");

Console.WriteLine("Please check the result");

}

[Test]

public void SidesnegativeforNOTPOSSIBLE()

{

//Arrange

int s1 = -1, s2 = -2, s3 = -3;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "All sides enterred are less then zero, Triangle not possible");

Console.WriteLine("Please check the result");

}

[Test]

public void Sides0and1and2forNOTPOSSIBLE()

{

//Arrange

int s1 = 0, s2 = 1, s3 = 2;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "One of the side is Zero, Triangle not Posssible");

Console.WriteLine("Please check the result");

}

[Test]

public void Sides0and0and2forNOTPOSSIBLE()

{

//Arrange

int s1 = 0, s2 = 0, s3 = 2;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "Two of the side are Zero, Triangle not Posssible");

Console.WriteLine("Please check the result");

}

[Test]

public void SidesNegateiveand1and2forNOTPOSSIBLE()

{

//Arrange

int s1 = -5, s2 = 1, s3 = 2;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "One of the side is Negative, Triangle not Posssible");

Console.WriteLine("Please check the result");

}

[Test]

public void SidesNegateiveand2forNOTPOSSIBLE()

{

//Arrange

int s1 = -5, s2 = -11, s3 = 2;

TriangleSolver obj = new TriangleSolver();

//Act

String length = obj.Analyze(s1, s2, s3);

//Assert

Assert.AreEqual(length, "Two of the side are Negative, Triangle not Posssible");

Console.WriteLine("Please check the result");

}

}

}

TriangleSolver.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace QATestingAssignment02

{

public class TriangleSolver

{

public string Analyze(int s1, int s2, int s3)

{

int a = s1 + s2, b = s2 + s3, c = s3 + s1;

if ((a > s3) && (b > s1) && (c > s2))

{

//Console.WriteLine("Trinagle is Posssible.");

if (s1 > 0 && s2 > 0 && s3 > 0)

{

if (s1 == s2 && s2 == s3 && s3 == s1)

return "It's Equilateral";

if ((s1 == s2 && s2 != s3 && s3 != s1) || (s1 != s2 && s2 == s3 && s3 != s1) || (s1 != s2 && s2 != s3 && s3 == s1))

return "It's Isosceles";

if (s1 != s2 && s2 != s3 && s3 != s1)

return "It's Scalene";

}

}

//for zero inputs

else if (s1 == 0 && s2 == 0 && s3 == 0)

{

return "All sides enterred are zero, Triangle not possible";

}

else if ((s1 == 0 && s2 != 0 && s3 != 0) || (s1 != 0 && s2 == 0 && s3 != 0) || (s1 != 0 && s2 != 0 && s3 == 0))

{

return "One of the side is Zero, Triangle not Posssible";

}

else if ((s1 == 0 && s2 == 0 && s3 != 0) || (s1 == 0 && s2 != 0 && s3 == 0) || (s1 != 0 && s2 == 0 && s3 == 0))

{

return "Two of the side are Zero, Triangle not Posssible";

}

//for negative inputs

else if ((s1 < 0 && s2 >= 0 && s3 >= 0) || (s1 >= 0 && s2 < 0 && s3 >= 0) || (s1 >= 0 && s2 >= 0 && s3 < 0))

{

return "One of the side is Negative, Triangle not Posssible";

}

else if ((s1 < 0 && s2 < 0 && s3 >= 0) || (s1 < 0 && s2 >= 0 && s3 < 0) || (s1 >= 0 && s2 < 0 && s3 < 0))

{

return "Two of the side are Negative, Triangle not Posssible";

}

else if (s1 < 0 && s2 < 0 && s3 < 0)

{

return "All sides enterred are less then zero, Triangle not possible";

}

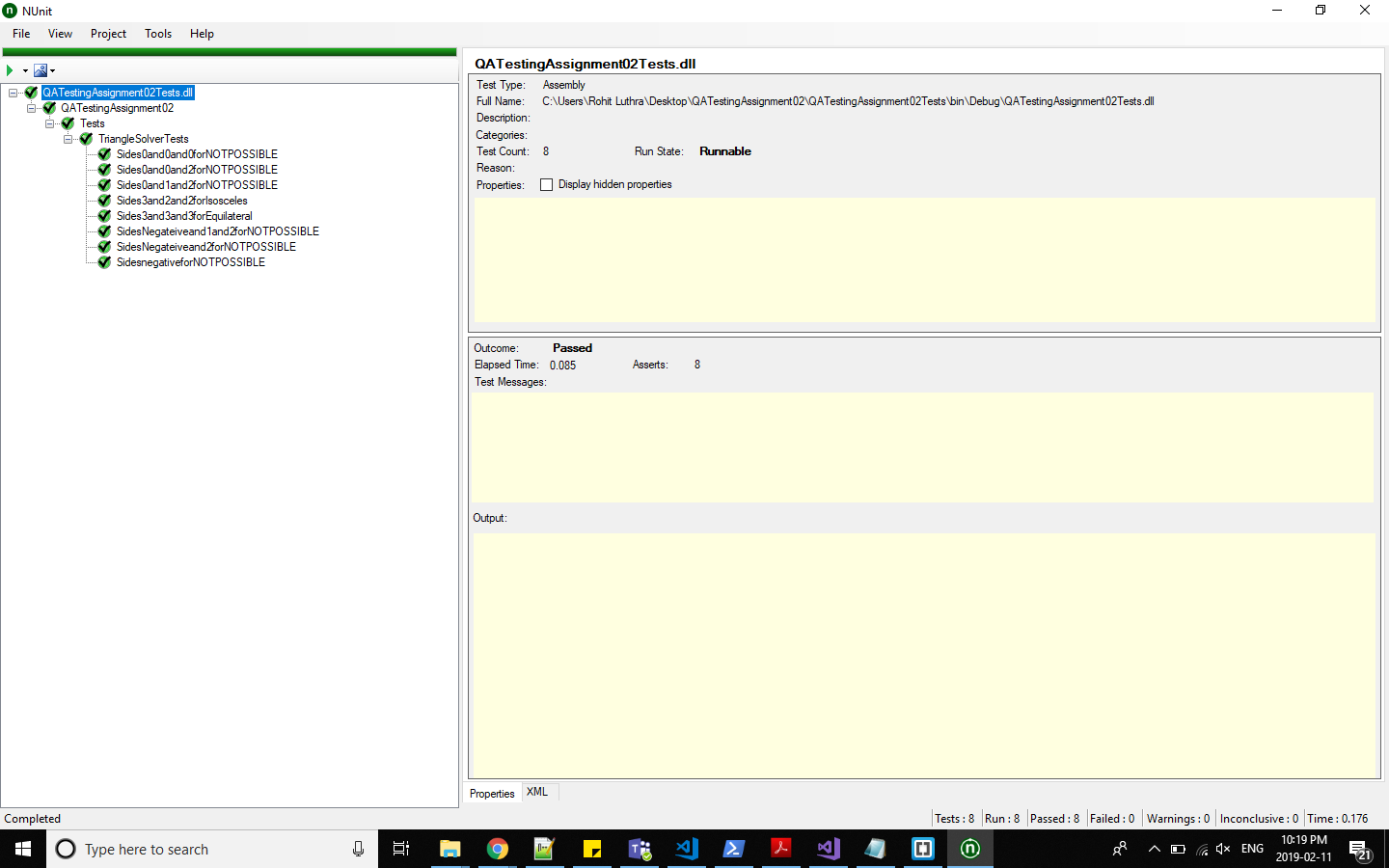
else { return "Invalid inputs for creating a triangle"; }

return "";

}

}

}



**CFG**

CC = 14-11+2 = 5