**ASSIGNMENT 2**

**SOFTWARE QUALITY ASSURANCE TECHNIQUES**

**SUMITTED BY**

**VIVEK KATROJU-8565947**

**DATE: 20/06/2019**

1. Program Code:
2. using System;
3. using System.Collections.Generic;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks; 6

7 namespace ClassLibrary1 8 {

9 class Program

10 {

11 public static void Main(string[] args) 12 {

1. Program ts = new Program();
2. ts.Go();
3. Console.ReadKey(); 16

17 }

18 public void Go()

19 {

1. double side1, side2, side3;
2. int input;
3. //helps keep program running
4. while(true)

24 {

1. //program initialization starts here
2. try

27 {

28 Console .Write("-------------------------\n" );

29 Console .Write(" Find type of Triangle\n" );

30 Console .Write("-------------------------\n" );

1. Console .WriteLine("\n1.Enter the Triangle dimensions\n" );
2. Console .WriteLine("\n2.Exit \n"); 33
3. //lets user decide to run program or not and requests to choose correct option
4. if ((!int.TryParse(Console.ReadLine(), out input)) || input <= 0 || input > 2)

36 {

1. Console.WriteLine("Please Enter the Correct option: ");
2. continue ;

39 }

40 Console .Write("------------------------\n" );

1. //gets the input of side 1 of a triangle
2. if (input == 1)

43 {

44 do

45 {

1. Console .Write("Enter Side 1 of a triangle\n" );
2. side1 = Convert.ToDouble(Console.ReadLine()); 48
3. // while (!int.TryParse(Console.ReadLine(), out side1)) ;
4. } while (side1 <= 0);

51

52

1. //gets the input of side 2 of a triangle
2. do

55 {

1. Console .Write("Enter Side 2 of a triangle\n" );
2. side2 = Convert.ToDouble(Console.ReadLine()); 58

59

60 } while (side2 <= 0);

61

1. //gets the input of side 3 of a triangle
2. do

64 {

1. Console .Write("Enter Side 3 of a triangle\n" );
2. side3 = Convert.ToDouble(Console.ReadLine());
3. } while (side3 <= 0);

68

69 Console .WriteLine(triangleSolver.Analyze(side1, side2, side3));

70 }

1. //condition to exit the program
2. else if (input == 2)

73 {

74 Environment .Exit(0);

75 }

1. //condition if invalid input is not valid
2. else

78 {

79 Console .WriteLine("input is not valid"); 80 }

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 81 |  |  |  |  | } |
| 82 |  |  |  |  |  |
| 83 |  |  |  |  | catch (Exception) |
| 84 |  |  |  |  | { |
| 85 |  |  |  |  | Console .WriteLine("Enter numbers only"); |
| 86 |  |  |  |  | } |
| 87 |  |  |  | } |  |
| 88 |  |  |  |  |  |
| 89 |  |  | } |  |  |
| 90 |  |  |  |  |  |
| 91 |  | } |  |  |  |
| 92 | } |  |  |  |  |
| 93 |  |  |  |  |  |

1. TRIANGLE SOLVER CODE
2. using System;
3. using System.Collections.Generic;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks; 6

7 namespace ClassLibrary1 8 {

9 public static class triangleSolver 10 {

11 public static string Analyze(double side1, double side2, double side3) 12 {

1. //rules for forming an triangle
2. double sSide1 = side1 + side2;
3. double sSide2 = side2 + side3;
4. double sSide3 = side3 + side1;
5. //condition to check triangle is formed or not
6. if (sSide1 > side3 && sSide2 > side1 && sSide3 > side2) 19 {
7. //condition of equilateral triangle
8. if (side1 == side2 && side2 == side3)

22 {

23 return " This is a equilateral triangle!!!" ; 24 }

1. //condition of isosceles triangle
2. else if (side1 == side2 || side2 == side3 || side3 == side1)

27 {

28 return "This is a isosceles triangle!!!" ; 29 }

30 else

31 {

32 return "This is a Scalene triangle!!!" ; 33 }

34

35 }

1. //if any conditions are not met
2. else return "Triangle cannot be formed" ;

38 }

39 }

40 }

41

1. TRIANGLE TEST CASES
   1. using System;
   2. using System.Collections.Generic;
   3. using System.Linq;
   4. using System.Text;
   5. using System.Threading.Tasks;
   6. using NUnit.Framework;
   7. using ClassLibrary1; 8

9 namespace triangleTestClass 10 {

1. [TestFixture]
2. public class triangelTestCases 13 {

14

1. //test of equilateral triangle
2. [Test]
3. public void checkEuilateralTriangle() 18 {
4. //Arrange
5. int side1 = 2, side2 = 2, side3 = 2; 21
6. //Act
7. String check = triangleSolver.Analyze(side1, side2, side3);
8. //Assert
9. Assert.AreEqual(check, " This is a equilateral triangle!!!" ); 26 }
10. //test case when triangle conditions area not met
11. [Test]
12. public void Sides2and8and6forNotPossible() 30 {
13. //Arrange
14. int side1 = 1, side2 = 7, side3 = 5; 33
15. //Act
16. String check = triangleSolver.Analyze(side1, side2, side3);
17. //Assert
18. Assert.AreEqual(check, "Triangle cannot be formed" ); 38

39 }

1. //test case when one of input size is negative
2. [Test]
3. public void Sides2and8andNegative6forNotPossible() 43 {
4. //Arrange
5. int side1 = 2, side2 = 8, side3 = -6; 46
6. //Act
7. String check = triangleSolver.Analyze(side1, side2, side3);
8. //Assert

50

51

52 }

Assert.AreEqual(check, "Triangle cannot be formed" );

1. //test case when any one side size is zero
2. [Test]
3. public void anySidesare0()

56 {

1. //Arrange
2. int side1 = 0, side2 = 8, side3 = 2; 59
3. //Act
4. String check = triangleSolver.Analyze(side1, side2, side3);
5. //Assert
6. Assert.AreEqual(check, "Triangle cannot be formed" ); 64 }
7. //test case to check Isosceles Triangle
8. [Test]
9. public void checkIsoscelesTriangle()

68 {

1. //Arrange
2. int side1 = 8, side2 = 8, side3 = 6; 71
3. //Act
4. String check = triangleSolver.Analyze(side1, side2, side3);
5. //Assert
6. Assert.AreEqual(check, "This is a isosceles triangle!!!" ); 76 }
7. //test case t0 check Scalene triangle
8. [Test]
9. public void checkScaleneTriangle()

80 {

1. //Arrange
2. int side1 = 3, side2 = 4, side3 = 5; 83
3. //Act
4. String check = triangleSolver.Analyze(side1, side2, side3);
5. //Assert
6. Assert.AreEqual(check, "This is a Scalene triangle!!!" ); 88 }
7. //test case to check when two sides are negative
8. [Test]
9. public void checkInvalidInput()

92 {

1. //Arrange
2. int side1 =- 3, side2 = 4, side3 = -4 ; 95
3. //Act
4. String check = triangleSolver.Analyze(side1, side2, side3);
5. /Assert

|  |  |  |  |
| --- | --- | --- | --- |
| 99 |  |  | Assert.AreEqual(check, "Triangle cannot be formed" ); |
| 100 |  |  | } |
| 101 |  |  | //test case when size in decimals |
| 102 |  |  | [Test] |
| 103 |  |  | public void sizeInDecimals() |
| 104 |  |  | { |
| 105 |  |  | //Arrange |
| 106 |  |  | double side1 = 3.5, side2 = 4, side3 = 4; |
| 107 |  |  |  |
| 108 |  |  | //Act |
| 109 |  |  | String check = triangleSolver.Analyze(side1, side2, side3); |
| 110 |  |  | //Assert |
| 111 |  |  | Assert.AreEqual(check, "This is a isosceles triangle!!!" ); |
| 112 |  |  | } |
| 113 |  | } |  |
| 114 | } |  |  |
| 115 |  |  |  |



CFG DIAGRAM

Cyclomatic Complexity Calculation:

M = E - N + 2P

E= number of edges= 12

N= number of nodes=10

M=12-10+2(1)= 4

The number 4 for the CC means that the program is relatively easy to test and not very complex.