# **Assignment -01: Unit Testing using NUnit**

# **Aparna Tomar – 7802150**

# **PROG8170**

# **Professor Firouzeh Sharifi Lotfabad**

# **GITHUB:**

# **#1 Source Code for Program.cs**

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. namespace Triangle
7. {
8. class Program
9. {
10. static void Main(string[] args)
11. {
12. Console.WriteLine("Welcome to Triangle Analyzer");
13. string inputString;
14. long choice, firstSide, secondSide, thirdSide;
15. string fSide, sSide, tSide;
16. do
17. {
18. Console.WriteLine("1. Enter triangle dimensions.\n2. Exit");
19. inputString = Console.ReadLine();
20. } while (!(long.TryParse(inputString, out choice) && (choice == 1 || choice == 2)));
21. while (choice == 1)
22. {
23. do
24. {
25. Console.Write("Enter length of First Side of Triangle(1-2147483647): ");
26. fSide = Console.ReadLine();
27. } while (!(long.TryParse(fSide, out firstSide) && (firstSide >= 1 && firstSide <= 2147483647)));
28. do
29. {
30. Console.Write("Enter length of Second Side of Triangle(1-2147483647): ");
31. sSide = Console.ReadLine();
32. } while (!(long.TryParse(sSide, out secondSide) && (secondSide >= 1 && secondSide <= 2147483647)));
33. do
34. {
35. Console.Write("Enter length of Third Side of Triangle(1-2147483647): ");
36. tSide = Console.ReadLine();
37. } while (!(long.TryParse(tSide, out thirdSide) && (thirdSide >= 1 && thirdSide <= 2147483647)));
38. string output = TriangleSolver.Analyze(firstSide, secondSide, thirdSide);
39. Console.WriteLine(output);
40. System.Threading.Thread.Sleep(1000);
41. do
42. {
43. Console.WriteLine("1. Enter triangle dimensions.\n2. Exit");
44. inputString = Console.ReadLine();
45. } while (!(long.TryParse(inputString, out choice) && (choice == 1 || choice == 2)));
46. }
47. if (choice == 2)
48. {
49. Console.WriteLine("BBYE!");
50. System.Threading.Thread.Sleep(1000);
51. System.Environment.Exit(1);
52. }
53. }
54. }
55. }

# **#2 Source Code of TriangleSolver.cs**

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. namespace Triangle
7. {
8. public static class TriangleSolver
9. {
10. public static string Analyze(long fSide, long sSide, long tSide)
11. {
12. //checking if sides are valid.
13. string message = "";
14. if ((fSide + sSide > tSide) && (fSide + tSide > sSide) && (sSide + tSide > fSide))
15. {
16. if (fSide == sSide && sSide == tSide)
17. {
18. message = "Given numbers ARE sides of an EQUILATERAL Triangle.";
19. }
20. else if (fSide == sSide || fSide == tSide || sSide == tSide)
21. {
22. message = "Given numbers ARE sides of an ISOSCELES Triangle.";
23. }
24. else
25. {
26. message = "Given numbers ARE sides of an SCALENE Triangle.";
27. }
28. }
29. else
30. {
31. message = "Given numbers CAN NOT be sides of a Triangle.";
32. }
33. return message;
34. }
35. }
36. }

# **#3 Source Code of TriangleSolverTest.cs**

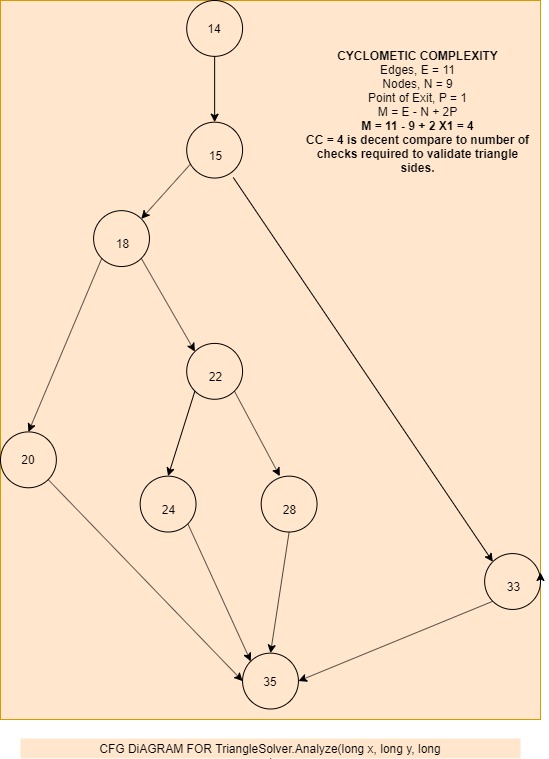
1. using System;
2. using Triangle;
3. using NUnit.Framework;
4. namespace Triangle
5. {
6. [TestFixture]
7. public class TriangleSolverTest
8. {
9. [TestCase, Description("Valid equilateral triangle.")]
10. public void Analyze\_ifThreeSidesAreEqual\_expectEquilateral()
11. {
12. Assert.AreEqual("Given numbers ARE sides of an EQUILATERAL Triangle.", TriangleSolver.Analyze(5, 5, 5));
13. }
14. [TestCase, Description("Valid isosceles triangle.")]
15. public void Analyze\_ifTwoSidesAreEqual\_expectIsosceles()
16. {
17. Assert.AreEqual("Given numbers ARE sides of an ISOSCELES Triangle.", TriangleSolver.Analyze(10, 10, 15));
18. }
19. [TestCase, Description("Valid scalene triangle.")]
20. public void Analyze\_ifNoSideEqual\_expectScalene()
21. {
22. Assert.AreEqual("Given numbers ARE sides of an SCALENE Triangle.", TriangleSolver.Analyze(5, 7, 10));
23. }
24. [TestCase, Description("Sum of two sides is less than third.")]
25. public void Analyze\_ifSumOfTwoSidesIsLessThanThird\_expectInvalidTriangle()
26. {
27. Assert.AreEqual("Given numbers CAN NOT be sides of a Triangle.", TriangleSolver.Analyze(10, 5, 1));
28. }
29. [TestCase, Description("All three sides have MAX permissible length and equilateral triangle.")]
30. public void Analyze\_ifThreeSidesAreEqualAndHoldBoundryValues\_expectEquilateral()
31. {
32. Assert.AreEqual("Given numbers ARE sides of an EQUILATERAL Triangle.", TriangleSolver.Analyze(2147483647, 2147483647, 2147483647));
33. }
34. [TestCase, Description("Two sides at maximum possible value.")]
35. public void Analyze\_ifTwoSidesAreEqualAndHoldBoundryValues\_expectIsosceles()
36. {
37. Assert.AreEqual("Given numbers ARE sides of an ISOSCELES Triangle.", TriangleSolver.Analyze(2147483647, 2147483647, 15));
38. }
39. [TestCase, Description("Tow sides at MIN and one side at MAX.")]
40. public void Analyze\_ifTwoSidesHoldMinBoundryValuesThirdSideHoldMaxValue\_expectInvalidTriangle()
41. {
42. Assert.AreEqual("Given numbers CAN NOT be sides of a Triangle.", TriangleSolver.Analyze(1, 1, 2147483647));
43. }
44. [TestCase, Description("Sum of two sides is equal to the thrid, produces invalid triangle.")]
45. public void Analyze\_ifSumOfTwoSidesIsEqualToThirdSide\_expectInvalidTriangle()
46. {
47. Assert.AreEqual("Given numbers CAN NOT be sides of a Triangle.", TriangleSolver.Analyze(12, 7, 5));
48. }
49. }
50. }

# **#4 CFG for Analyze Method**

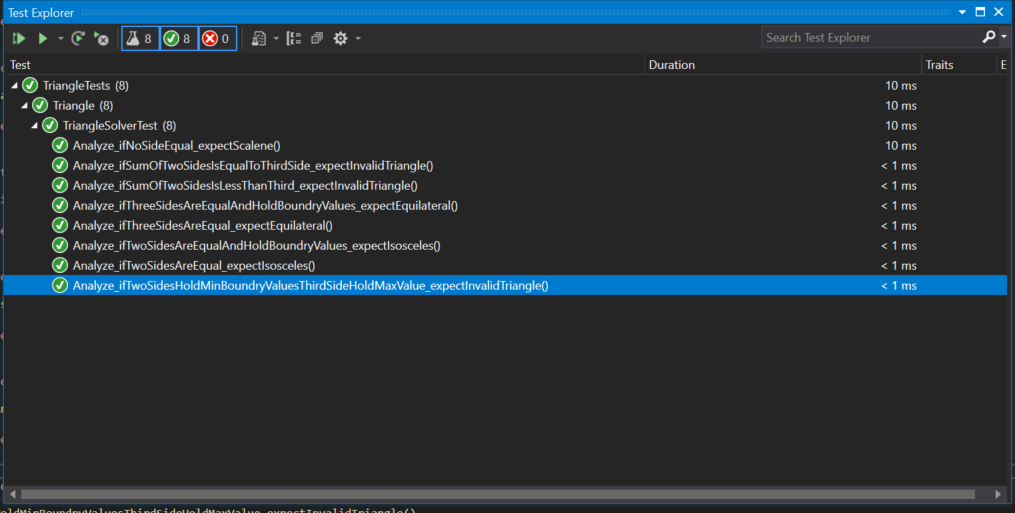
CFG is an important metrics being used in Static Unit Testing. Using CFG, we can track various paths that the code might follow.

Cyclometic Graphs tend to get complicated if routine gets bigger.

My Analyze method shows **CFG: 4, which I think is decent score which means that method which I wrote have Average Cyclometic Complexity.**



# **#5 NUnit Successful Test Runs Screenshot**



# **#6 Git Hub Push**