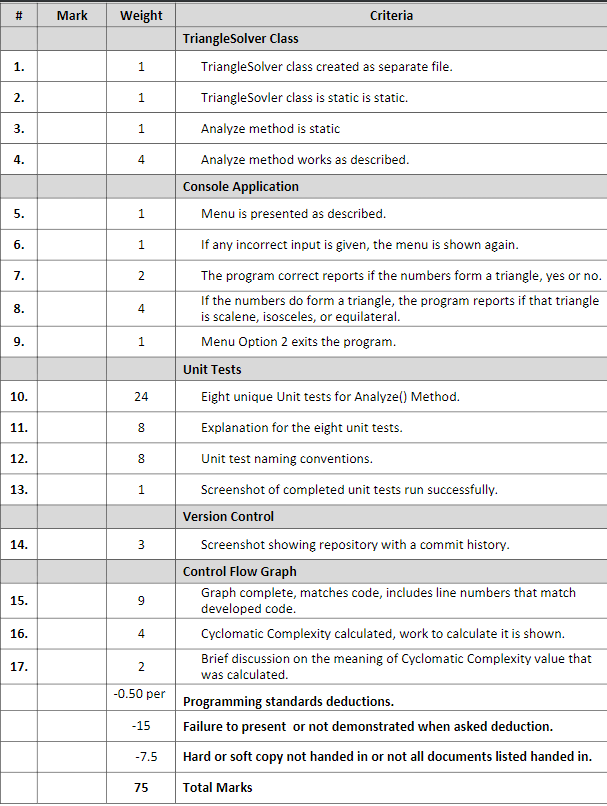
**Assignment #2**

Tymur Koltunov

StudentID 8672727

28/09/2019

**Rubrics**



**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace assignment2

{

class Program

{

static void Main(string[] args)

{

int menuoption = 0;

int a = 0, b = 0, c = 0;

do

{

do

{

Console.WriteLine("1. Enter triangle dimensions.");

Console.WriteLine("2. Exit.");

Console.Write("Enter menu option: ");

if (!(int.TryParse(Console.ReadLine().Trim(), out menuoption)) || (menuoption > 2) || (menuoption < 1))

{

Console.WriteLine("Incorrect input");

Console.WriteLine();

}

else

break;

} while(true);

switch (menuoption)

{

default:

Console.WriteLine("Something is wrong");

break;

case 1:

do

{

Console.Write("Enter dimension A: ");

if (!(int.TryParse(Console.ReadLine().Trim(), out a)) || (a < 1)) //upper limit is defined in integer, as there were no requiremets that would specify the max length of the triangle dimesion

{

Console.WriteLine("Incorrect Input");

}

else

break;

} while (true);

do

{

Console.Write("Enter dimension B: ");

if (!(int.TryParse(Console.ReadLine().Trim(), out b)) || (b < 1)) //upper limit is defined in integer, as there were no requiremets that would specify the max length of the triangle dimesion

{

Console.WriteLine("Incorrect Input");

}

else

break;

} while (true);

do

{

Console.Write("Enter dimension C: ");

if (!(int.TryParse(Console.ReadLine().Trim(), out c)) || (c < 1)) //upper limit is defined in integer, as there were no requiremets that would specify the max length of the triangle dimesion

{

Console.WriteLine("Incorrect Input");

}

else

break;

} while (true);

Console.WriteLine("Your triangle is: {0}", TriangleSolver.Analyze(a, b, c));

do

{

Console.WriteLine("Do you want to enter another triangle?[y/n]");

string answer = Console.ReadLine().Trim();

if (answer != "y" && answer != "n")

{

Console.WriteLine("Incorrect Input");

}

else if (answer == "y")

break;

else

return;

} while(true);

break;

case 2:

return;

}

} while (true);

}

}

}

**TriangleSolver class**

public static class TriangleSolver

{

public static string Analyze(int a, int b, int c)

{

string type = string.Empty;

if ((a + b > c) && (a + c > b) && (b + c > a))

{

if ((a == b) && (a == c) && (b == c))

type = "equilateral";

else if ((a == b) || (a == c) || (b == c))

type = "isosceles";

else

type = "scalene";

}

else

type = "not possible";

return type;

}

}

**Tests class**

namespace Analyze\_Tests

{

[TestFixture]

public class Tests

{

[Test]

public void Analyze\_InputA1B1C1\_ExpectedOutputEquilateral()

{

//Arrange

int a = 1;

int b = 1;

int c = 1;

string expected = "equilateral";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA2B2C3\_ExpectedOutputIsosceles()

{

//Arrange

int a = 2;

int b = 2;

int c = 3;

string expected = "isosceles";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA2B3C2\_ExpectedOutputIsosceles()

{

//Arrange

int a = 2;

int b = 3;

int c = 2;

string expected = "isosceles";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA3B2C2\_ExpectedOutputIsosceles()

{

//Arrange

int a = 3;

int b = 2;

int c = 2;

string expected = "isosceles";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA2B3C4\_ExpectedOutputScalene()

{

//Arrange

int a = 2;

int b = 3;

int c = 4;

string expected = "scalene";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA1B1C2\_ExpectedOutputNotPossible()

{

//Arrange

int a = 1;

int b = 1;

int c = 2;

string expected = "not possible";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA2B1C1\_ExpectedOutputNotPossible()

{

//Arrange

int a = 2;

int b = 1;

int c = 1;

string expected = "not possible";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA1B2C1\_ExpectedOutputNotPossible()

{

//Arrange

int a = 1;

int b = 2;

int c = 1;

string expected = "not possible";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

Assert.AreEqual(expected, actual);

}

[Test]

public void Analyze\_InputA2B5C10\_ExpectedOutputNotPossible()

{

//Arrange

int a = 2;

int b = 5;

int c = 10;

string expected = "not possible";

string actual = string.Empty;

//Act

actual = TriangleSolver.Analyze(a, b, c);

//Assert

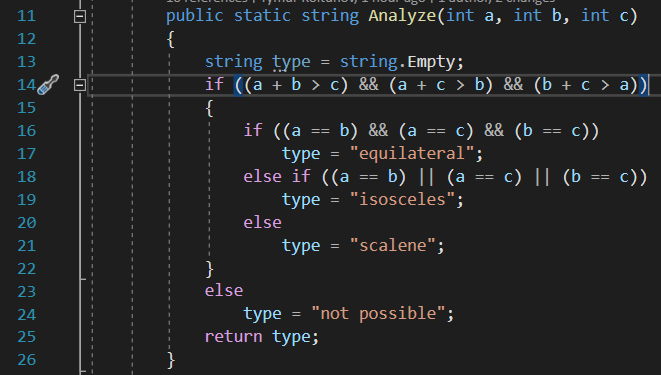
Assert.AreEqual(expected, actual);

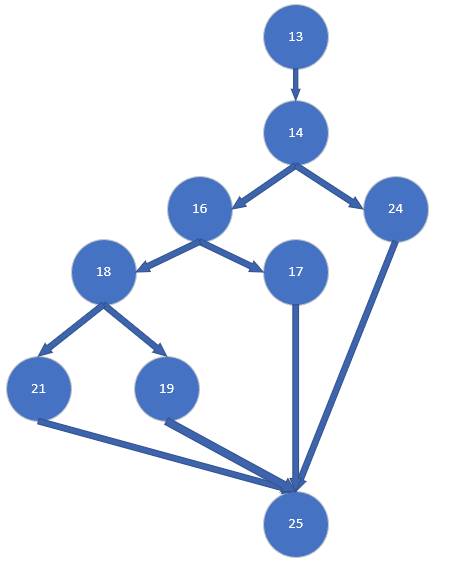
}

}

}

**Control Flow Diagram of Analyze method**





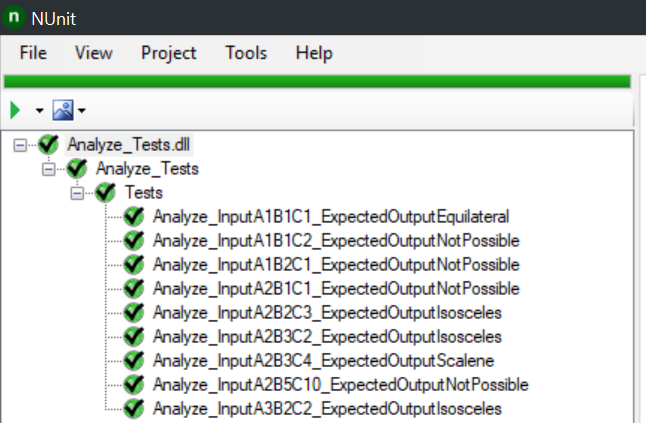
(Number of edges) minus (number of nodes) plus (2 multiplied by number of connected components).

Cyclomatic Complexity for this method is equal to 11 – 9 + 2.

Cyclomatic Complexity for this method is equal to 4.

According to McCabe methods with CC greater than 10 should be split into multiple modules. Considering that CC of method Analyze is equal to 4 we can say that method Analyze is quite simple, and testing every possible path is possible.

**NUnit**



Comments on tests:

1.a=1, b=1, c=1. Testing an equilateral triangle.

2. a=1, b=1, c=2 Testing an impossible triangle.

3. a=1, b=2, c=1 and a=2, b=1, c=1. Testing the “if” statement to work if different dimensions have impossible values.

4. a=2, b=2, c=3. Testing an isosceles triangle.

5. a=2, b=3, c=2 and a=3, b=2, c=2. Testing the “if” statement to work if different dimensions have equal value.

6. a=2, b=3, c=4. Testing a scalene triangle.

7. a=2, b=5, c=10. Testing an impossible triangle if every dimension has different values.

**GIT**

