Day 18 Assignment
By
Triveni Anumolu
16-02-2022

1.What is the use of XML

- XML stands for Extensible Markup Language.
- XML is used for universal data transfer mechanism to send data across different platforms.

2. Write the points discussed about XML in the class.

- XML stands for Extensible Markup Language.
- It has user-defined tags.
- It is case sensitive.
- It can have only one root tag.
- It is used for universal data transfer mechanism to send data across different platform.
- XML files are platform independent.

Types of XML:
1.Tag based XML
2.Attribute based XML

3. Create a simple xml to illustrate:
a. Tag based xml with 10 products
b. Attribute based xml

A .Tag Based XML:

```
<Products>
   <Product1>
      <Id>1</Id>
      <Name>Bag</Name>
      <Price>500</Price>
   </Product1>
   <Product2>
      <Id>2</Id>
      <Name>Mobile</Name>
      <Price>20000</Price>
   </Product2>
   <Product3>
      <Id>3</Id>
      <Name>Table</Name>
      <Price>1000</Price>
   </Product3>
    <Product4>
      <Id>4</Id>
```

```
    <Name>TV</Name>
    <Price>50000</Price>
  </Product4>
  <Product5>
    <Id>5</Id>
    <Name>Shoes</Name>
    <Price>1800</Price>
  </Product5>
  <Product6>
    <Id>6</Id>
    <Name>KeyBoard</Name>
    <Price>1000</Price>
  </Product6>
  <Product7>
    <Id>7</Id>
    <Name>Printer</Name>
    <Price>1000</Price>
  </Product7>
   <Product8>
    <Id>8</Id>
    <Name>Air Conditioner</Name>
    <Price>50000</Price>
  </Product8>
  <Product9>
    <Id>9</Id>
    <Name>Water Purifier</Name>
    <Price>5000</Price>
  </Product9>
  <Product10>
    <Id>10</Id>
    <Name>Chair</Name>
    <Price>5000</Price>
  </Product10>
</Products>
```

Result:


Tagsxml.xml
File | C:/Users/admin/Deskto
This XML file does not appear to have any style inform
<Products>
<Product1>
<Id>1</Id>
<Name>Bag</Name>
<Price>500</Price>
</Product1>
<Product2>
<Id>2</Id>
<Name>Mobile</Name>
<Price>20000</Price>
</Product2>
<Product3>
<Id>3</Id>
<Name>Table</Name>
<Price>1000</Price>
</Product3>
<Product4>
<Id>4</Id>
<Name>TV</Name>
<Price>50000</Price>
</Product4>
<Product5>
<Id>5</Id>
<Name>Shoes</Name>
<Price>1800</Price>
</Product5>
<Product6>
<Id>6</Id>
<Name>KeyBoard</Name>
<Price>1000</Price>
</Product6>
<Product7>
<Id>7</Id>
<Name>Printer</Name>
Page 2 of 2 169 words English (U.S.)

```
    </Product6>
  ▼<Product7>
      <Id>7</Id>
      <Name>Printer</Name>
      <Price>1000</Price>
    </Product7>
  ▼<Product8>
      <Id>8</Id>
      <Name>Air Conditioner</Name>
      <Price>50000</Price>
    </Product8>
  ▼<Product9>
      <Id>9</Id>
      <Name>Water Purifier</Name>
      <Price>5000</Price>
    </Product9>
  ▼<Product10>
      <Id>10</Id>
      <Name>Chair</Name>
      <Price>5000</Price>
    </Product10>
  </Products>
```

Page 3 of 3 169 words English (U.S.)

B . Attribute based tags:

```
<Products>
   <Product1 Id="1" Name="Bag" Price="800" />
   <Product2 Id="2" Name="Shoes" Price="1800" />
   <Product3 Id="3" Name="Table" Price="2000" />
   <Product4 Id="4" Name="TV" Price="4800" />
   <Product5 Id="5" Name="Printer" Price="30000" />
   <Product6 Id="6" Name="Air Conditioner" Price="60000" />
   <Product7 Id="7" Name="Mobile" Price="2800" />
   <Product8 Id="8" Name="Book" Price="970" />
   <Product9 Id="9" Name="Chair" Price="4000" />
   <Product10 Id="10" Name="Fridge" Price="30800" />
</Products>
```

Output:

This XML file does not appear to have any style information associate


```
▼<Products>
   <Product1 Id="1" Name="Bag" Price="800"/>
   <Product2 Id="2" Name="Shoes" Price="1800"/>
   <Product3 Id="3" Name="Table" Price="2000"/>
   <Product4 Id="4" Name="TV" Price="4800"/>
   <Product5 Id="5" Name="Printer" Price="30000"/>
   <Product6 Id="6" Name="Air Conditioner" Price="60000"/>
   <Product7 Id="7" Name="Mobile" Price="2800"/>
   <Product8 Id="8" Name="Book" Price="970"/>
   <Product9 Id="9" Name="Chair" Price="4000"/>
   <Product10 Id="10" Name="Fridge" Price="30800"/>
 </Products>
```


4.Convert the above XML to JSON and display the JSON data

Attribute Based XML:

```
<Products>
   <Product1 Id="1" Name="Bag" Price="800" />
   <Product2 Id="2" Name="Shoes" Price="1800" />
   <Product3 Id="3" Name="Table" Price="2000" />
   <Product4 Id="4" Name="TV" Price="4800" />
   <Product5 Id="5" Name="Printer" Price="30000" />
   <Product6 Id="6" Name="Air Conditioner" Price="60000" />
   <Product7 Id="7" Name="Mobile" Price="2800" />
   <Product8 Id="8" Name="Book" Price="970" />
   <Product9 Id="9" Name="Chair" Price="4000" />
   <Product10 Id="10" Name="Fridge" Price="30800" />
</Products>
```

JSON data:


```
{   "Product1": {      "@Id": "1",      "@Name": "Bag",      "@Price": "800"   },
"Product2": {      "@Id": "2",      "@Name": "Shoes",      "@Price": "1800"   },
"Product3": {      "@Id": "3",      "@Name": "Table",      "@Price": "2000"   },
"Product4": {      "@Id": "4",      "@Name": "TV",      "@Price": "4800"   },   "Product5":
{      "@Id": "5",      "@Name": "Printer",      "@Price": "30000"   },   "Product6": {
"@Id": "6",      "@Name": "Air Conditioner",      "@Price": "60000"   },   "Product7": {
"@Id": "7",      "@Name": "Mobile",      "@Price": "2800"   },   "Product8": {      "@Id":
"8",      "@Name": "Book",      "@Price": "970"   },   "Product9": {      "@Id": "9",
"@Name": "Chair",      "@Price": "4000"   },   "Product10": {      "@Id": "10",
"@Name": "Fridge",      "@Price": "30800"   }}
```

5. Research and write the benefits of JSON over XML

Benefits of JSON over XML:

- JSON takes less memory compared to XML.
- JSON is easier to read in its expanded form than XML.
- JSON is faster because it is designed specifically for data interchange whereas XML is slower, because it is designed for a lot more than just data interchange.

6. For the below requirement, create a layered architecture project with separate class library for Business logic.
create console application
create windows(or desktop) application
Business Requirement:
FIND FACTORIAL OF A NUMBER:
0 = 1
positive number (up to 7) = factorial answer
n > 7 = -999 (as answer)
n < 0 = -9999 (as answer)
put the screen shots of the output and project (solution explorer) screen shot

Library:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MathLibrary
{
   public class Algebra
   {
      public static int Factorial(int n)
      {

         if (n == 0)
            return 1;
         else if (n > 7)
            return -999;
         else if (n < 0)
            return -9999;
         else
         {
            int fact = 1;
            for (int i = 1; i <= n; i++)
```

```
            {
                fact = fact * i;
            }
            return fact;
        }
    }
}
```

Console Application:

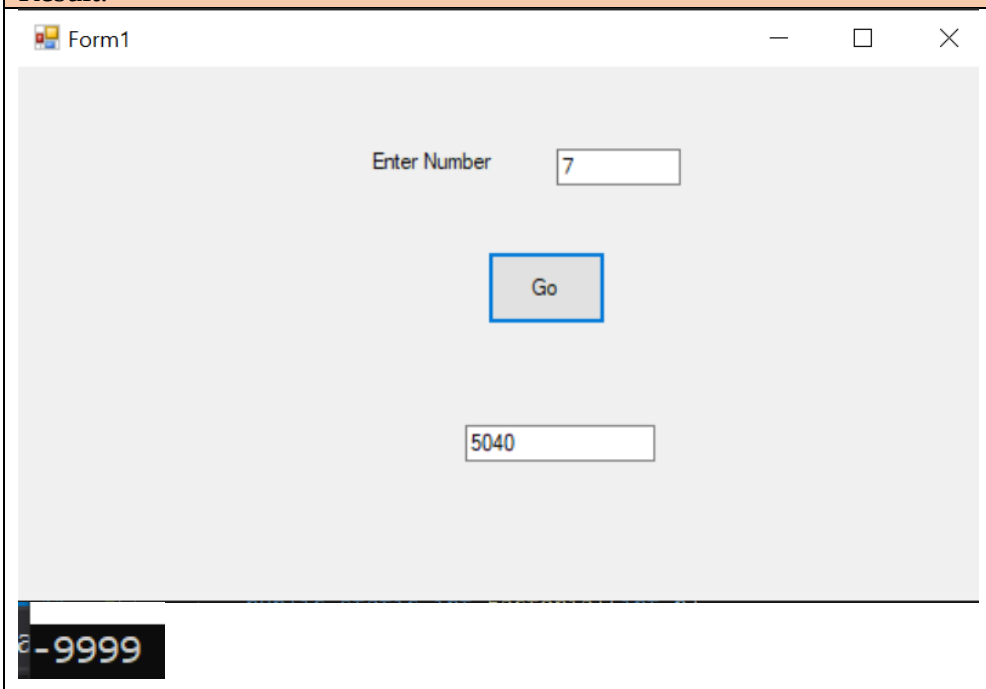
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MathLibrary;

namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(Algebra.Factorial(-5));
            Console.ReadLine();
        }
    }
}
```

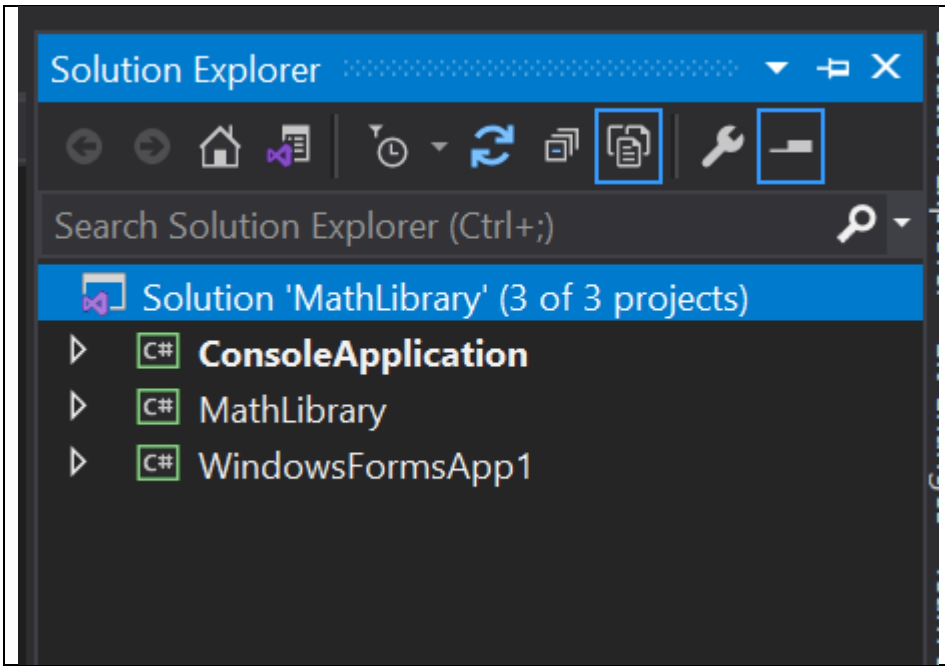
WindowForm:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MathLibrary;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
```

```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int n =Convert.ToInt32(textBox1.Text);
            int fact = Algebra.Factorial(n);
            textBox2.Text = fact.ToString();

        }
    }
}
```

Result:





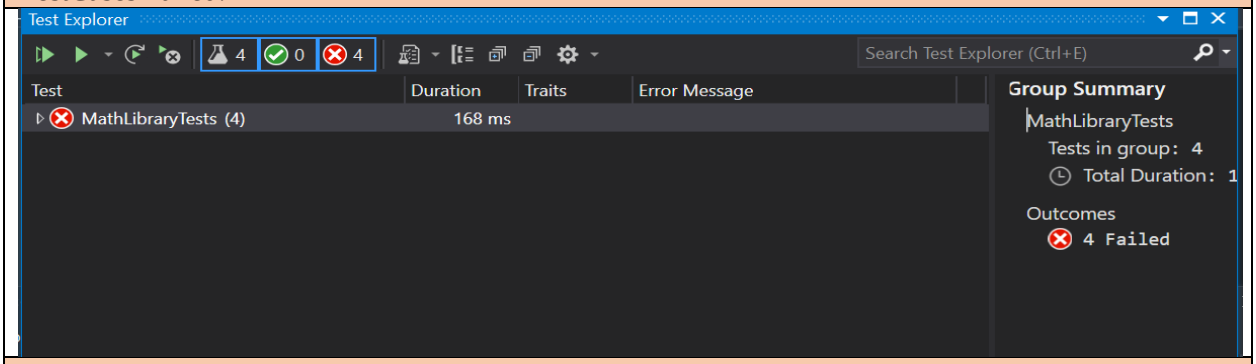
7. For the above method, Implement TDD and write 4 test cases and put the code in word document.
put the screen shot of all test cases failing.
make the test cases pass.
put the screen shot

AlgebraTests:

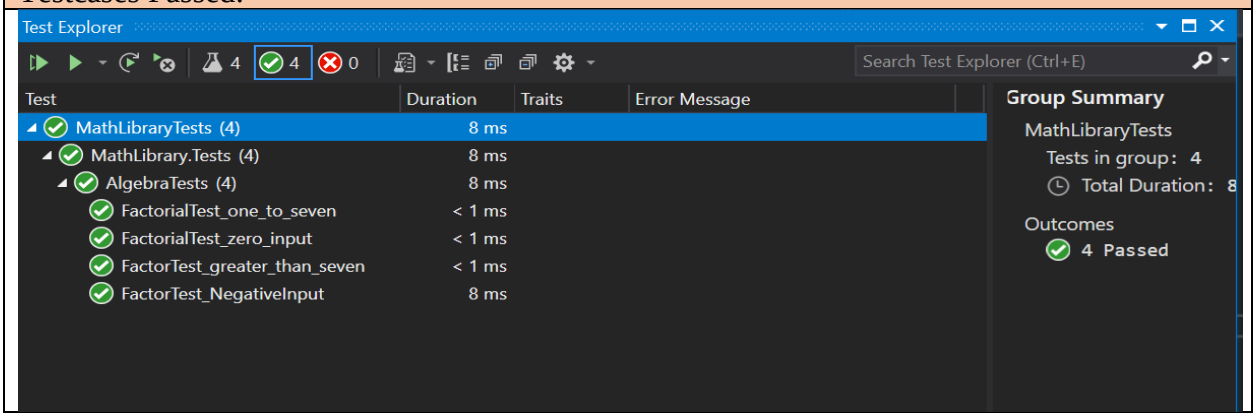
```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MathLibrary;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MathLibrary.Tests
{
    [TestClass()]
    public class AlgebraTests
    {
        [TestMethod()]
        public void FactorialTest_zero_input()
        {
            //Arrange
            int n = 0;
            int expected = 1;
```

```
            //Act
            int actual = Algebra.Factorial(n);

            //Assert
            Assert.AreEqual(expected, actual);
        }

        [TestMethod()]
        public void FactorialTest_one_to_seven()
        {
            //Arrange
            int n=2;
            int expected = 2;

            //Act
            int actual = Algebra.Factorial(n);

            //Assert
            Assert.AreEqual(expected, actual);
        }
        [TestMethod()]
        public void FactorTest_NegativeInput()
        {
            //Arrange
            int n = -6;
            int expected = -9999;

            //Act
            int actual = Algebra.Factorial(n);

            //Assert
            Assert.AreEqual(expected, actual);
        }
        [TestMethod()]
        public void FactorTest_greater_than_seven()
        {
            //Arrange
            int n = 9;
            int expected = -999;

            //Act
            int actual = Algebra.Factorial(n);

            //Assert
            Assert.AreEqual(expected, actual);
        }
    }
}
```

TestCases Failed:



Testcases Passed:



8. Add one more method to check if the number is palindrome or not in the above Algebra class and write test case for the same.

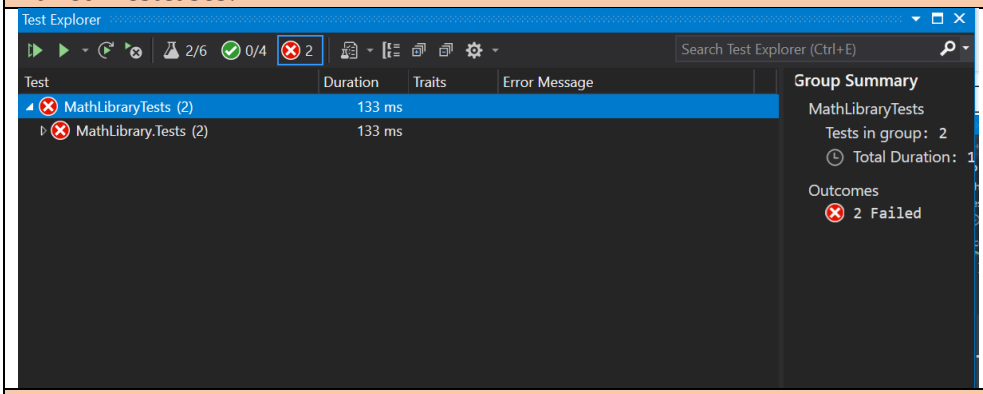
Code:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MathLibrary;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MathLibrary.Tests
{
   [TestClass()]
   public class AlgebraTests
   {
      [TestMethod()]
      public void FactorialTest_Zero_Input()
      {
         //Arrange
         int n = 0;
```

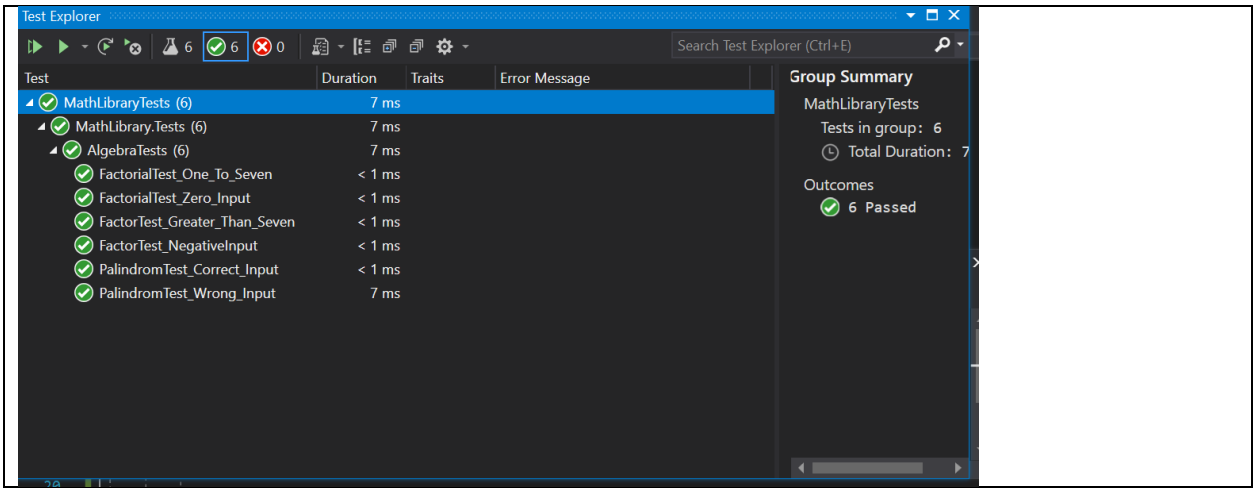
```
        int expected = 1;

        //Act
        int actual = Algebra.Factorial(n);

        //Assert
        Assert.AreEqual(expected, actual);
    }

    [TestMethod()]
    public void FactorialTest_One_To_Seven()
    {
        //Arrange
        int n=2;
        int expected = 2;

        //Act
        int actual = Algebra.Factorial(n);

        //Assert
        Assert.AreEqual(expected, actual);
    }
    [TestMethod()]
    public void FactorTest_NegativeInput()
    {
        //Arrange
        int n = -6;
        int expected = -9999;

        //Act
        int actual = Algebra.Factorial(n);

        //Assert
        Assert.AreEqual(expected, actual);
    }
    [TestMethod()]
    public void FactorTest_Greater_Than_Seven()
    {
        //Arrange
        int n = 9;
        int expected = -999;

        //Act
        int actual = Algebra.Factorial(n);

        //Assert
        Assert.AreEqual(expected, actual);
    }
```

```
        [TestMethod()]
        public void PalindromTest_Correct_Input()
        {
            //Arrange
            int n = 97779;
            string expected = "Palindrom";

            //Act
            string actual = Algebra.Palindrom(n);

            //Assert
            Assert.AreEqual(expected, actual);
        }
        [TestMethod()]
        public void PalindromTest_Wrong_Input()
        {
            //Arrange
            int n = 9719;
            string expected = "Not a Palindrom";

            //Act
            string actual = Algebra.Palindrom(n);

            //Assert
            Assert.AreEqual(expected, actual);
        }
    }
}
```

Failed Testcases:



Passed Testcases:


Test Explorer
6
6
0
Search Test Explorer (Ctrl+E)
Test
Duration
Traits
Error Message
MathLibraryTests (6)
7 ms
MathLibrary.Tests (6)
7 ms
AlgebraTests (6)
7 ms
FactorialTest_One_To_Seven
< 1 ms
FactorialTest_Zero_Input
< 1 ms
FactorTest_Greater_Than_Seven
< 1 ms
FactorTest_NegativeInput
< 1 ms
PalindromTest_Correct_Input
< 1 ms
PalindromTest_Wrong_Input
7 ms
Group Summary
MathLibraryTests
Tests in group: 6
Total Duration: 7
Outcomes
6 Passed