

Day18 assignment

Bhanu Prakash Reddy Chilukuri



February 16, 2022

nb haelhcare technologies

|  |
| --- |
| 1. What is the use of XML? |

* XML used for universal data transfer mechanism to send data across different platforms.
* XML uses human language, but not computer language. XML is usable and understandable.
* XML is extendable.
* XML uses user defined tags.

|  |
| --- |
| 2. Write the points discussed about xml in the class. |

* XML stands for Extensible Markup Language.
* XML uses user defined tags.
* XML can have only one root tag.
* XML used for universal data transfer mechanism to send data across
* different platforms.
* 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 |

Tag based XML:

<Products>

<Product1>

<Name>BasketBall</Name>

<Cost>2500</Cost>

<Distributor>Ram Sports</Distributor>

<Manufacturer>NBASports</Manufacturer>

</Product1>

<Product2>

<Name>Vapor 1.3 Bat</Name>

<Cost>3500</Cost>

<Distributor>Amazon</Distributor>

<Manufacturer>Gray Nicholls</Manufacturer>

</Product2>

<Product3>

<Name>Alpha Gen Cricket Bat</Name>

<Cost>3000</Cost>

<Distributor>Amazon</Distributor>

<Manufacturer>Gray Nicholls</Manufacturer>

</Product3>

<Product4>

<Name>NanoFlare 800</Name>

<Cost>4000</Cost>

<Distributor>Flipkart</Distributor>

<Manufacturer>Yonex</Manufacturer>

</Product4>

<Product5>

<Name>Yonex Astrox 99</Name>

<Cost>7000</Cost>

<Distributor>Flipkart</Distributor>

<Manufacturer>Yonex</Manufacturer>

</Product5>

<Product6>

<Name>Shield 30 Cricket Ball</Name>

<Cost>750</Cost>

<Distributor>Sachin Sports</Distributor>

<Manufacturer>SG</Manufacturer>

</Product6>

<Product7>

<Name>AJ Bouncer Ball</Name>

<Cost>500</Cost>

<Distributor>Ram Sports</Distributor>

<Manufacturer>SG</Manufacturer>

</Product7>

<Product8>

<Name>Knee Cap</Name>

<Cost>250</Cost>

<Distributor>Krishna Sports</Distributor>

<Manufacturer>New Balance</Manufacturer>

</Product8>

<Product9>

<Name>Cricket Helmet</Name>

<Cost>1500</Cost>

<Distributor>Ram Sports</Distributor>

<Manufacturer>SG</Manufacturer>

</Product9>

<Product10>

<Name>Lebron BasketBall</Name>

<Cost>12500</Cost>

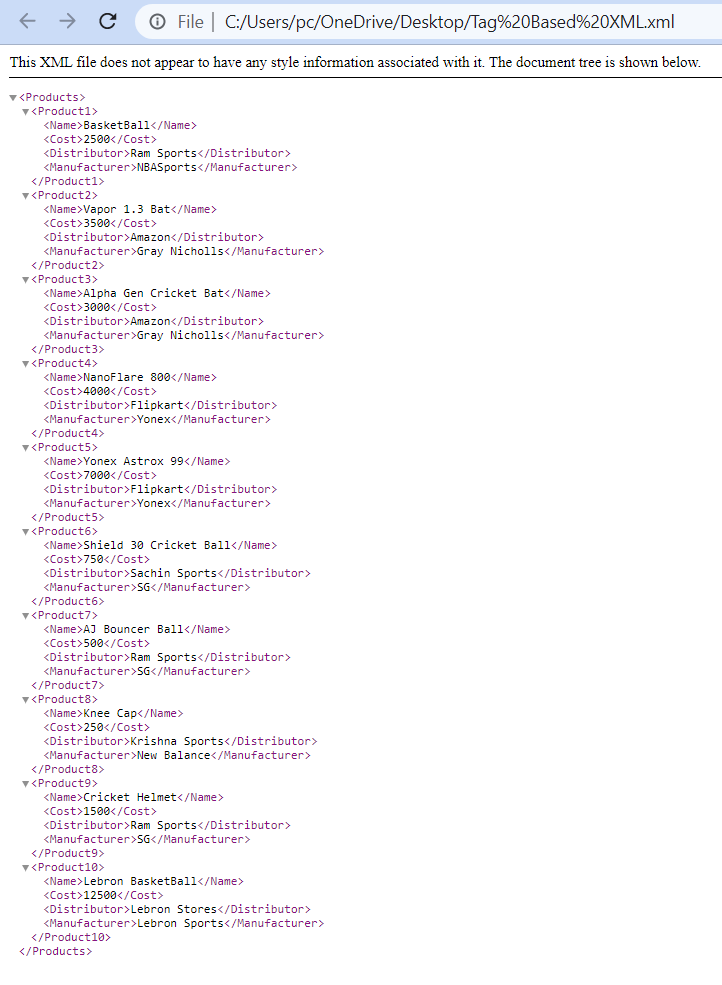
<Distributor>Lebron Stores</Distributor>

<Manufacturer>Lebron Sports</Manufacturer>

</Product10>

</Products>

Output:



Attribute Based Tag:

<Products>

<Product1 Name="BasketBall" Cost="2500" Distributor="Ram Sports" Manufacturer="NBASports"/>

<Product2 Name="Vapor1.3 Bat" Cost="3500" Distributor="Amazon" Manufacturer="Gray Nocholls"/>

<Product3 Name="AlphaGen CricketBat" Cost="2500" Distributor="Amazon" Manufacturer="Gray Nicholls"/>

<Product4 Name="NanoFlare 800" Cost="4500" Distributor="Flipkart" Manufacturer="Yonex"/>

<Product5 Name="Yonex Astr0x 99" Cost="6500" Distributor="Flipkart" Manufacturer="Yonex"/>

<Product6 Name="Shield30 CricketBall" Cost="500" Distributor="Sachin Sports" Manufacturer="SG"/>

<Product7 Name="AJ BOuncer" Cost="700" Distributor="Sacin Sports" Manufacturer="SG"/>

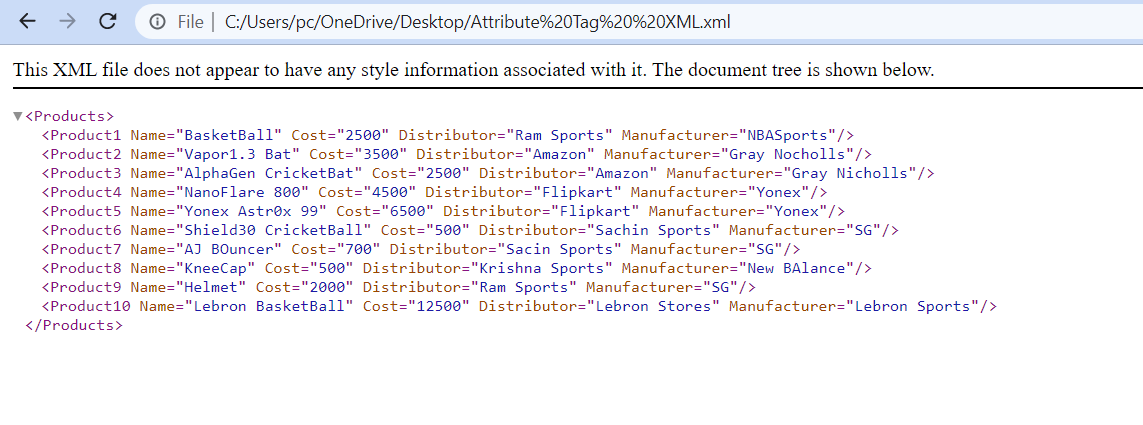
<Product8 Name="KneeCap" Cost="500" Distributor="Krishna Sports" Manufacturer="New BAlance"/>

<Product9 Name="Helmet" Cost="2000" Distributor="Ram Sports" Manufacturer="SG"/>

<Product10 Name="Lebron BasketBall" Cost="12500" Distributor="Lebron Stores" Manufacturer="Lebron Sports"/>

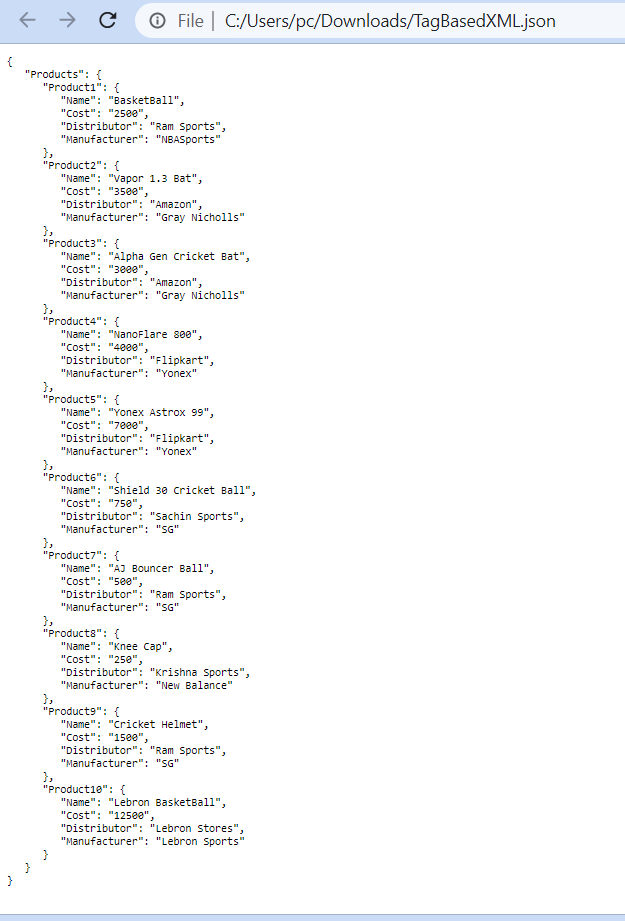
</Products>

Output:

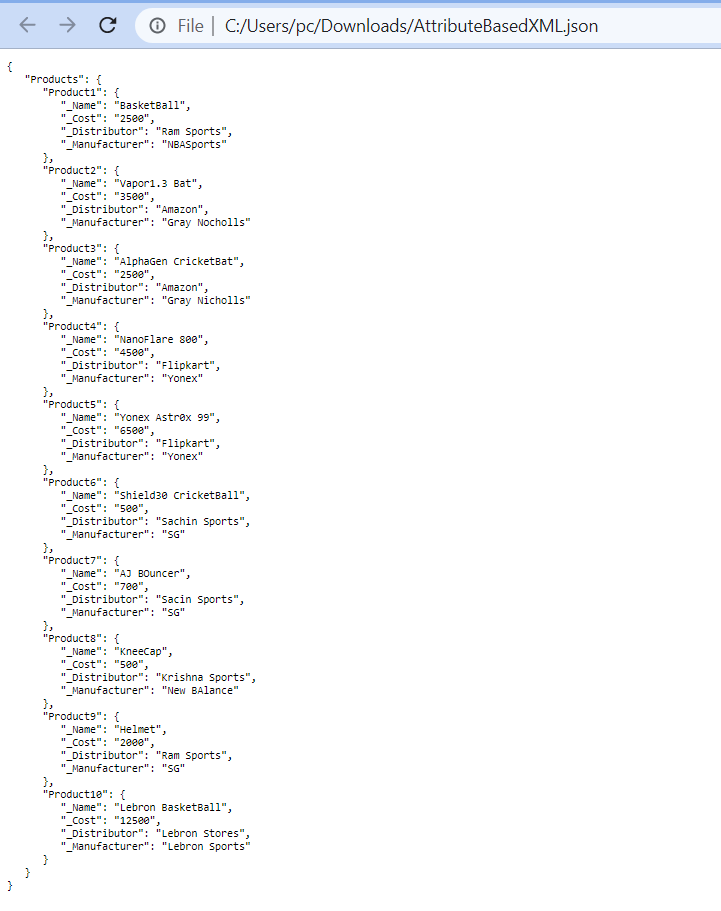


|  |
| --- |
| 4. Convert the above xml to JSON and display the JSON data. |

Tag Based XML to JSON:



Attribute Based XML to JSON:



|  |
| --- |
| 5. Research and write the benefits of JSON over XML. |

* JSON takes less memory.
* JSON requires less tags than XML.
* JSON is easier to read than the XML.
* JSON is simple text. This fact makes it suitable and safe for transferring across platforms and operating systems that do not readily share more complex document types. As text, JSON can also be readily displayed and edited in simple editors.
* JSON is compact. An average JSON string is about two thirds of the size of the same data in XML.

|  |
| --- |
| 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  > 7 = -999 (as answer)  < 0 = -9999 (as answer)  put the screen shots of the output and  project (solution explorer) screen shot |

Code:

namespace MathematicsLibrary

{

public class Algebra

{

public static int Factorial(int n)

{

int fact = 1;

if (n == 0)

return 1;

else if (n > 7)

return -999;

else if (n < 0)

return -9999;

else

for (int i = 1; i <= n; i++)

fact \*= i;

return fact;

}

}

}

using MathematicsLibrary;

namespace Day18\_Project1

{

internal class Program

{

static void Main(string[] args)

{

int n;

Console.WriteLine("Enter Number: ");

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

Console.WriteLine(Algebra.Factorial(n));

Console.ReadLine();

}

}

}

using MathematicsLibrary;

namespace WindowsApp

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

int n= Convert.ToInt32(textBox1.Text);

int fact=Algebra.Factorial(n);

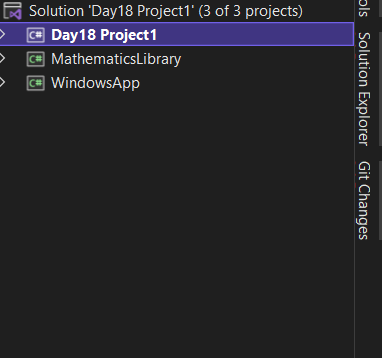
textBox2.Text = fact.ToString();

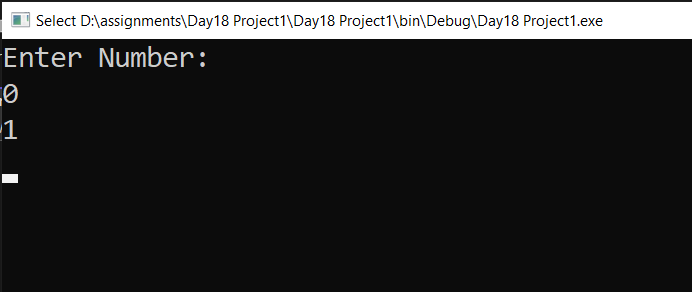
}

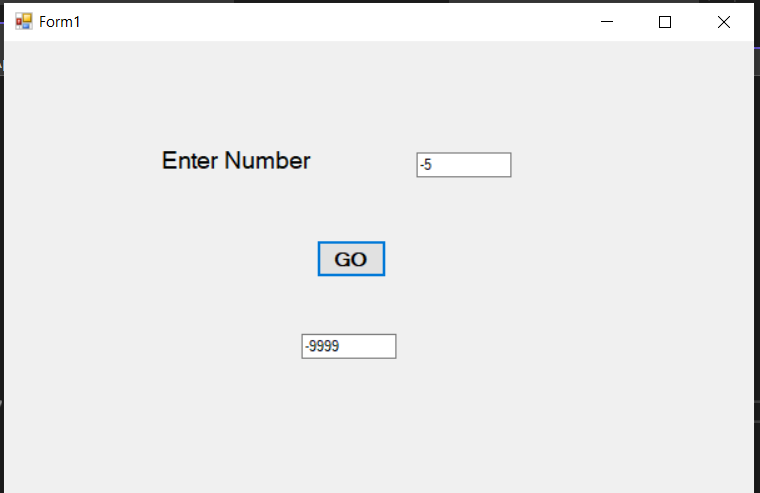
}

}

Output:







|  |
| --- |
| 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. |

Code:

namespace MathematicsLibrary.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\_Input()

{

//Arrange

int n = 4;

int expected = 24;

//Act

int actual = Algebra.Factorial(n);

//Assert

Assert.AreEqual(expected, actual);

}

[TestMethod()]

public void FactorialTest\_Negative\_Input()

{

//Arrange

int n = -5;

int expected = -9999;

//Act

int actual = Algebra.Factorial(n);

//Assert

Assert.AreEqual(expected, actual);

}

[TestMethod()]

public void FactorialTest\_Greater\_Than\_Seven\_Input()

{

//Arrange

int n = 9;

int expected = -999;

//Act

int actual = Algebra.Factorial(n);

//Assert

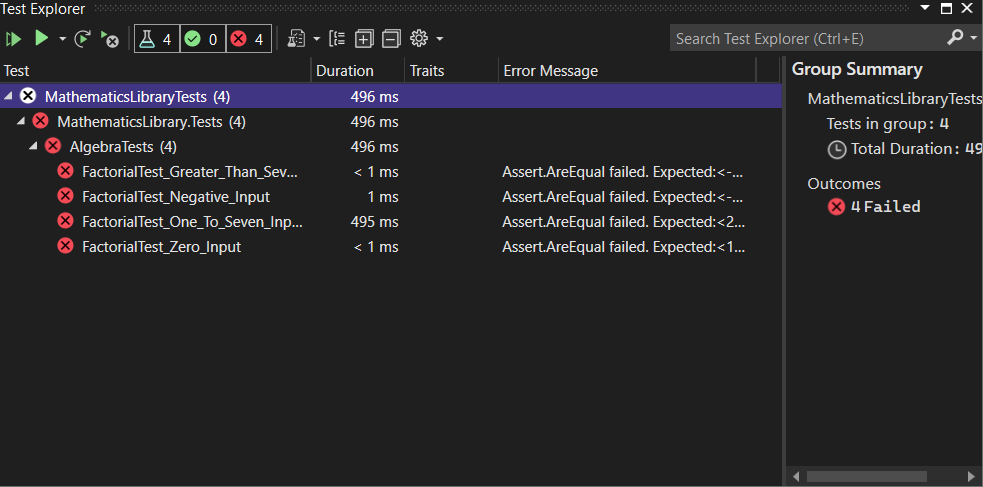
Assert.AreEqual(expected, actual);

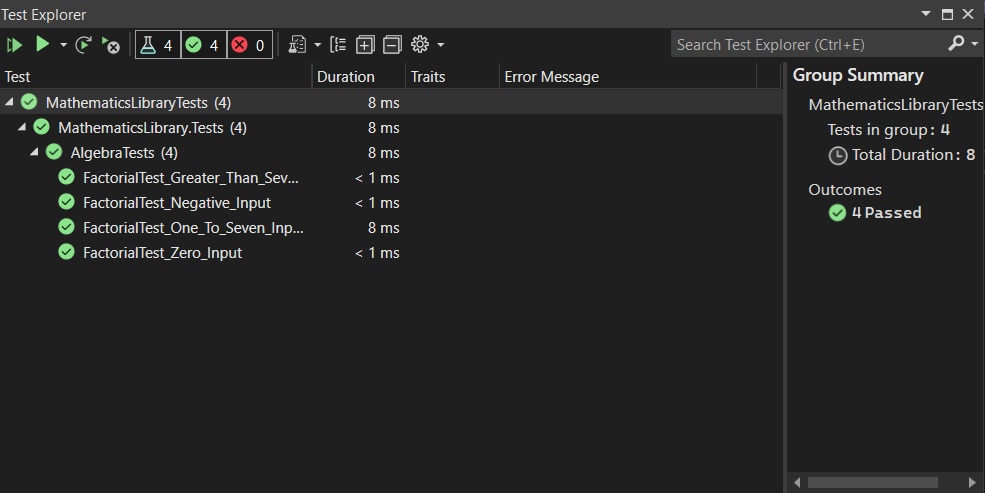
}

}

}

Output:





|  |
| --- |
| 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:

[TestMethod()]

public void Palindrome\_Right\_Input\_Test()

{

//Arrange

int n = 23532;

string expected = "Palindrome";

//Act

string actual = Algebra.Palindrome(n);

//Assert

Assert.AreEqual(expected,actual);

}

[TestMethod()]

public void Palindrome\_Wrong\_Input\_Test()

{

//Arrange

int n = 1566;

string expected = "Not Palindrome";

//Act

string actual = Algebra.Palindrome(n);

//Assert

Assert.AreEqual(expected, actual);

}

public static string Palindrome(int n)

{

int sum = 0, rem, temp;

temp = n;

while(n>0)

{

rem = n % 10;

sum = (sum \* 10) + rem;

n = n / 10;

}

if (temp == sum)

return "Palindrome";

else

return "Not Palindrome";

}

Output:

