|  |
| --- |
| Day 18 Assignment  BY  Nanam Vaishnavi  16 – FEB – 2022 |

|  |
| --- |
| 1. **What is the use of XML** |
| 1) XML is used for universal data transfer mechanism to send data across different platform.  2) We can make use of XML in Web publishing, Web searching and General applications. |

|  |
| --- |
| 1. **Write the points discussed about xml in the class.** |
| 1) **XML – Extensible Mark-up Language**  2) It consists of User – defined tags.  3) XML is a Case Sensitive  4) XML consists only one root tag.  5) It is used for universal data transfer mechanism to send data across different platform. |

|  |
| --- |
| 1. **Create a simple xml to illustrate:**   **a. Tag based xml with 10 products**  **b. Attribute based xml** |
| * **Tag based xml with 10 products** |
| <Products>  <Product>  <ID>501</ID>  <Item>Laptop</Item>  <Price>45000</Price>  </Product>  <Product>  <ID>502</ID>  <Item>Mobile</Item>  <Brand>Vivo</Brand>  </Product>    <Product>  <ID>503</ID>  <Item>Tablet</Item>  <Brand>Amazon</Brand>  </Product>    <Product>  <ID>504</ID>  <Item>Headsets</Item>  <Brand>Bot</Brand>  </Product>    <Product>  <ID>505</ID>  <Item>Bluetooth</Item>  <Brand>Realm</Brand>  </Product>    <Product>  <ID>506</ID>  <Item>Mouse</Item>  <Brand>DELL</Brand>  </Product>    <Product>  <ID>507</ID>  <Item>Desktop</Item>  <Brand>HP</Brand>  </Product>    <Product>  <ID>508</ID>  <Item>Refrigerator</Item>  <Brand>Whirlpool</Brand>  </Product>  <Product>  <ID>509</ID>  <Item>Air Conditioner</Item>  <Brand>LG</Brand>  </Product>    <Product>  <ID>510</ID>  <Item>Washing Machine</Item>  <Brand>Bosch</Brand>  <Price>45000</Price>  </Product>  </Products> |
| **OUTPUT** |
|  |
| * **Attribute based xml** |
| <Products>  <Product ID="1201" Item="Mobile" Brand="Oppose"/>  <Product ID="1211" Item="Washing Machine" Brand="LG"/>  <Product ID="1231" Item="Laptop" Brand="Lenovo"/>  <Product ID="1241" Item="Headsets" Brand="Scull Candy"/>  <Product ID="1251" Item="Mouse" Brand="DELL"/>  <Product ID="1261" Item="Refrigerator" Brand="Whirlpool"/>  <Product ID="1271" Item="Watch" Brand="Henley"/>  <Product ID="1281" Item="Tablet" Brand="Amazon"/>  <Product ID="1291" Item="Bluetooth" Brand="OnePlus"/>  <Product ID="1301" Item="Air Conditioner" Brand="Bosch"/>  </Products> |
| **OUTPUT** |
|  |

|  |
| --- |
| 1. **Convert above XML to JSON and display the JSON data** |
| [https://www.freeformatter.com/3.7.0.0/img/minus.gif  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1201",  "@Item": "Mobile",  "@Brand": "Oppose"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1211",  "@Item": "Washing Machine",  "@Brand": "LG"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1231",  "@Item": "Laptop",  "@Brand": "Lenovo"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1241",  "@Item": "Headsets",  "@Brand": "Scull Candy"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1251",  "@Item": "Mouse",  "@Brand": "DELL"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1261",  "@Item": "Refrigerator",  "@Brand": "Whirlpool"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1271",  "@Item": "Watch",  "@Brand": "Henley"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1281",  "@Item": "Tablet",  "@Brand": "Amazon"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1291",  "@Item": "Bluetooth",  "@Brand": "OnePlus"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@ID": "1301",  "@Item": "Air Conditioner",  "@Brand": "Bosch"  }  ] |

|  |
| --- |
| 1. **Research and write the benefits of JSON Over XML (2 or 3 points)** |
| **Benefits of JSON Over XML**  1) JSON requires less tags than XML.  2) You can get JSON data from anywhere, not just your own domain.  3) JSON is easier to read than XML. |

|  |
| --- |
| 1. **For the below requirement, create a layered architecture project with separate class library for Business logic.**  * **Create Console Application** * **Create Windows Application**   **Business Requirement :**   * **FIND FACTORIAL OF A NUMBER :**   **0 = 1**  **Positive number(up to 7) = factorial answer**  **>7 = -999**  **<0 = -9999**  **Put the screenshots of output and solution explorer screenshot.** |
| **Algebra’s** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  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 = fact \* i;  return fact;    }  }  }  } |
| **Program.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using MathematicsLibrary;  namespace Day18Project2  {  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();  }  }  } |
| **Form.cs** |
| 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 MathematicsLibrary;  namespace MyApp1  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void button1\_Click(object sender, EventArgs e)  {  int n = Convert.ToInt32(textBox1.Text);  int res = Algebra.Factorial(n);  textBox2.Text = res.ToString();  }  }  } |
| **OUTPUT** |
| **Greater than 7 = -999 Factorial of a Number**      **0 = 1 Negative Values** |

|  |
| --- |
| **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** |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using MathematicsLibrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  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 = 5;  int expected = 120;  // Act  int actual = Algebra.Factorial(n);  // Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_Negative\_Input()  {  // Arrange  int n = -2;  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 = 8;  int expected = -999;  // Act  int actual = Algebra.Factorial(n);  // Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void AddTest()  {  // Arrange  int a = 6, b = 8;  int expected = 14;  //Act  int actual = Algebra.Add(a, b);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |
| **OUTPUT** |
| **All Test cases failed**    **All Test cases Passed** |