|  |
| --- |
| **Day 18 assignment**  **By**  **Paluru Mounika**  **16-02-2022** |

|  |
| --- |
| **1.What is the use of XML?** |
| **1**.XML is used for universal data transfer mechanism.  **2**.The XML is standard is a flexible way to create information and electronically share structured data via the public internet ,as well as via corporate network.  **3**.Xml is used to describe data. |
|  |

|  |
| --- |
| **2.Write the points discussed about XML in the class.** |
| 1.XML has only one root tag.  2.XML is case sensitive.  3.XML will have user defined tags.  4.XML is used for universal data transfer mechanism. |

|  |
| --- |
| **3.** **Create a simple xml to illustrate:**  **a. Tag based xml with 10 products**  **b. Attribute based xml** |
| **Tag based XML:** |
| Code:  <products>  <product>  <ID1>1234</ID1>  <name>phone</name>  <price>10000</price>  </product>  <product>  <ID2>134</ID2>  <name>laptop</name>  <price>40000</price>  </product>  <product>  <ID3>145</ID3>  <name>watch</name>  <price>2000</price>  </product>    <product>  <ID4>115</ID4>  <name>washing machine</name>  <price>25000</price>  </product>  <product>  <ID5>1236</ID5>  <name>refrigirator</name>  <price>30000</price>  </product>  <product>  <ID6>1789</ID6>  <name>tv</name>  <price>45000</price>  </product>  <product>  <ID7>3456</ID7>  <name>car</name>  <price>500000</price>  </product>  <product>  <ID8>145</ID8>  <name>airconditioner</name>  <price>50000</price>  </product>  <product>  <ID9>145</ID9>  <name>cooler</name>  <price>20000</price>  </product>  <product>  <ID10>145</ID10>  <name>microoven</name>  <price>25000</price>  </product>  </products> |
| Output: |
| **Attribute based XML:** |
| **Code:**  **<product>**  **<product ID1="1234" Name="phone" price="10000"/>**  **<product ID2="3456" Name="laptop" price="40000"/>**  **<product ID3="899" Name="watch" price="2000"/>**  **<product ID4="1134" Name="washing machine" price="25000"/>**  **<product ID5="1089" Name="refrigirator" price="30000"/>**  **<product ID6="2345" Name="cooler" price="20000"/>**  **<product ID7="8903" Name="Airconditioner" price="50000"/>**  **<product ID8="2367" Name="tv" price="25000"/>**  **<product ID9="2134" Name="microoven" price="15000"/>**  **<product ID10="1256" Name="bluetooth" price="1500"/>**  **</product>**   |  | | --- | | **Output:** | |

|  |
| --- |
| **4. Convert the above xml to JSON and display the JSON data.** |
| **Tag based Xml:** |
| **Code:**  {  "products": {  "product": [  {  "ID1": "1234",  "name": "phone",  "price": "10000"  },  {  "ID2": "134",  "name": "laptop",  "price": "40000"  },  {  "ID3": "145",  "name": "watch",  "price": "2000"  },  {  "ID4": "115",  "name": "washing machine",  "price": "25000"  },  {  "ID5": "1236",  "name": "refrigirator",  "price": "30000"  },  {  "ID6": "1789",  "name": "tv",  "price": "45000"  },  {  "ID7": "3456",  "name": "car",  "price": "500000"  },  {  "ID8": "145",  "name": "airconditioner",  "price": "50000"  },  {  "ID9": "145",  "name": "cooler",  "price": "20000"  },  {  "ID10": "145",  "name": "microoven",  "price": "25000"  }  ]  }  } |
| **Attribute based xml:** |
| **Code:**  {  "product": {  "product": [  {  "\_ID1": "1234",  "\_Name": "phone",  "\_price": "10000"  },  {  "\_ID2": "3456",  "\_Name": "laptop",  "\_price": "40000"  },  {  "\_ID3": "899",  "\_Name": "watch",  "\_price": "2000"  },  {  "\_ID4": "1134",  "\_Name": "washing machine",  "\_price": "25000"  },  {  "\_ID5": "1089",  "\_Name": "refrigirator",  "\_price": "30000"  },  {  "\_ID6": "2345",  "\_Name": "cooler",  "\_price": "20000"  },  {  "\_ID7": "8903",  "\_Name": "Airconditioner",  "\_price": "50000"  },  {  "\_ID8": "2367",  "\_Name": "tv",  "\_price": "25000"  },  {  "\_ID9": "2134",  "\_Name": "microoven",  "\_price": "15000"  },  {  "\_ID10": "1256",  "\_Name": "bluetooth",  "\_price": "1500"  }  ]  }  } |

|  |
| --- |
| **5. Research and write the benefits of JSON over XML**  **( 2 or 3 points ).** |
| 1.Advantage of using JSON file size smaller ;thus transferring data is faster than XML.  2.JSON is compact and very easy to read.  3.The simplicity of its structure and minimal syntax makes JSON easier to be used and read by human |

|  |
| --- |
| 6. For the below requirement, create a layered architecture  project with seperate class library for Business logic.    create console application  create windows(or desktop) application  Business Requirement:  FIND FACTORIAL OF A NUMBER:  0 = 1    positive number (upto 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:**  **Algebra.cs:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Author:Paluru Mounika  //purpose:algebra class  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace MathematicLibrary  {    public class Algebra  {  ///summary  ///factorial of number  ///summary  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:**  **Code:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using MathematicLibrary;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Author:Paluru Mounika  //Purpose:factorial of number  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace day18project1  {  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();  }  }  } |
| **Windowsapp:**  **Code**:  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 MathematicLibrary;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Authore:Paluru Mounika  //Purpose: factorial of number  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace MyApp  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void button1\_Click(object sender, EventArgs e)  {  int n=Convert.ToInt32(textBox1.Text);  int result=Algebra.Factorial(n);  textBox2.Text = result.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.** |
| **Algebra:**  **Code:**  using Microsoft.VisualStudio.TestTools.UnitTesting;  using MathematicLibrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Author:Paluru Mounika  //Purpose:algebra.factorial of a number  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace MathematicLibrary.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\_greater\_than\_Seven\_Input()  {  //Arrange  int n = 9;  int expected = -999 ;  //Act  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual (expected, actual);  }  [TestMethod()]  public void FactorialTest\_Negitive\_Input()  {  //Arrange  int n = -3;  int expected =-9999 ;  //Act  int actual=Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected , actual);  }  }  } |
| **AlgebraTest:**  **Code:**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Author:paluru Mounika  //purpose:factorial of a number  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace MathematicLibrary  {    public class Algebra  {  public static int Factorial(int n)  {  if (n == 0)  return 1;  else if (n < 0)  return -9999;  else if (n > 7)  return -999;  else  {  int fact = 1;  for(int i=1;i<=n;i++)  fact=fact\*i;  return fact;  }  return 0;  }  }        } |
| Output: |
|  |