|  |
| --- |
| **Day18 Morning Assignments**  **By**  **Manoj Yekolla**  **16-Feb-2022** |

|  |
| --- |
| **1. What is the use of XML ?** |
| * XML was designed to store and transfer data. |
| * XML uses Human ,not Computer ,Language . XML is Readable and Understandable. |
| * XML is Extendable. |

|  |
| --- |
| **2. Write the points discussed about xml in the class** |
| * XML Stands for Extensiable Markup Language . |
| * XML is a User defined Tags. |
| * XML is a Case Senstive. |
| * XML used for Universal Data Transfer mechansim to send data across different plateforms. |
| * XML in create only one root element. |

|  |
| --- |
| **3. Create a simple xml to illustrate:**  **a. Tag based xml with 10 products** |
| This XML file does not appear to have any style information associated with it. The document tree is shown below.  <Products>  <Product1>  <Id>1</Id>  <Name>Manoj Y</Name>  <Salary>1000</Salary>  </Product1>  <Product2>  <Id>2</Id>  <Name>Nithesh</Name>  <Salary>2000</Salary>  </Product2>  <Product3>  <Id>3</Id>  <Name>Gabbar</Name>  <Salary>5000</Salary>  </Product3>  <Product4>  <Id>4</Id>  <Name>praveen</Name>  <Salary>4000</Salary>  </Product4>  <Product5>  <Id>5</Id>  <Name>chakri</Name>  <Salary>8000</Salary>  </Product5>  <Product6>  <Id>6</Id>  <Name>hemanth</Name>  <Salary>10000</Salary>  </Product6>  <Product7>  <Id>7</Id>  <Name>dhinesh</Name>  <Salary>15000</Salary>  </Product7>  <Product8>  <Id>8</Id>  <Name>chandu</Name>  <Salary>8000</Salary>  </Product8>  <Product9>  <Id>9</Id>  <Name>raki</Name>  <Salary>6000</Salary>  </Product9>  <Product10>  <Id>10</Id>  <Name>Munna</Name>  <Salary>3000</Salary>  </Product10>  </Products> |
| **b. Attribute based xml** |
| This XML file does not appear to have any style information associated with it. The document tree is shown below.  <products>  <product id="1" name="manojY" salary="8990"/>  <product id="2" name="Dhawan" salary="1000"/>  <product id="3" name="Nithesh" salary="2000"/>  <product id="4" name="Munna" salary="3000"/>  <product id="5" name="Chakri" salary="4000"/>  <product id="6" name="chandhu" salary="5000"/>  <product id="7" name="dhinni" salary="6000"/>  <product id="8" name="mahi" salary="7000"/>  <product id="9" name="raki" salary="8000"/>  <product id="10" name="murali" salary="8994"/>  </products> |

|  |
| --- |
| **4. Convert the above xml to JSON and display the JSON data** |
| Jason Code : |
| [IMG_256 {IMG_257 "Id": "1", "Name": "Manoj Y", "Salary": "1000" }, {IMG_258 "Id": "2", "Name": "Krishna", "Salary": "2000" }, {IMG_259 "Id": "3", "Name": "Dhawan", "Salary": "3000" }, {IMG_260 "Id": "4", "Name": "Nithesh", "Salary": "4000" }, {IMG_261 "Id": "5", "Name": "Munnaswamy", "Salary": "5000" }, {IMG_262 "Id": "6", "Name": "Kata", "Salary": "6000" }, {IMG_263 "Id": "7", "Name": "Niranjan", "Salary": "8000" }, {IMG_264 "Id": "8", "Name": "Kohli", "Salary": "9000" }, {IMG_265 "Id": "9", "Name": "Chakri", "Salary": "10000" }, {IMG_266 "Id": "10", "Name": "Praveen", "Salary": "11000" }] |

|  |
| --- |
| **5. Research and write the benefits of JSON over XML**  **( 2 or 3 points )** |
| * Jason was less size compare to xml. |
| * Jason is Standard design for human readable data interchange. |
| * Jason Works with a tree structure and its looks like XML. |

|  |
| --- |
| **6. For the below requirement, create a layered architecture project with seperate class library for Business logic. create console 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 : Program.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using ManojLibrary1;  namespace Day18Project1  {  internal class Program  {  static void Main(string[] args)  {  int n;  Console.WriteLine("Enter a Number is :");  n=Convert.ToInt32(Console.ReadLine());  Console.WriteLine(Factorial.Algebra(n));  Console.ReadLine();  }  }  } |
| **Code :Factorial.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ManojLibrary1  {  public static class Factorial  {  public static int Algebra(int n)  {  int fact = 1;  if (n == 0)  return 1;  else if (n > 7)  return -999;  else if (n < 0)  return -999;  else  {  for (int i=1; i<=n; i++)  fact = fact \* i;  return fact;  }  }  }  } |
| Output :  Screenshot (301) |
| Screenshot (302) |
| Screenshot (303) |
| Screenshot (304) |
| Screenshot (305) |

|  |
| --- |
| **6(b) create windows(or desktop) application . Above Scenerio** |
| Code : form1.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 ManojLibrary1;  namespace Calculate\_App  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void button1\_Click(object sender, EventArgs e)  {  int n=Convert.ToInt32(textBox1.Text);  int result = Factorial.Algebra(n);  textBox2.Text=result.ToString();  }  }  } |
| Output :  Screenshot (310) |
| Screenshot (311) |
| Screenshot (312) |
| Screenshot (313) |
| Screenshot (315) |

|  |
| --- |
| **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 :FactorialTest.cs** |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using ManojLibrary1;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ManojLibrary1.Tests  {  [TestClass()]  public class FactorialTests  {  [TestMethod()]  public void AlgebraTest\_Zero\_Input()  {  //Arrange  int n = 0;  int expected = 1;  //Act  int actual = Factorial.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }    [TestMethod()]  public void AlgebraTest\_Input\_ZeroTo\_Seven()  {  //Arrange  int n = 5;  int expected = 120;  //Act  int actual = Factorial.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void AlgebraTest\_GreaterThan\_Seven()  {  //Arrange  int n = 8;  int expected = -999;  //Act  int actual = Factorial.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void AlgebraTest\_Negative\_Numbers()  {  //Arrange  int n = -4;  int expected = -999;  //Act  int actual = Factorial.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |
| **Code :Factorial.cs** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using ManojLibrary1;  namespace ManojLibrary1  {  public static class Factorial  {  public static int Algebra(int n)  {  if (n == 0)  return 1;  else if (n > 7)  return -999;  else if (n < 0)  return - 999;  else  {  int fact = 1;  for (int i = 1; i <= n; i++)  fact = fact \* i;  return fact;    }    return 0;  }  }  } |
| Output :  Screenshot (317) |
| Screenshot (322) |
| Screenshot (323) |
| Screenshot (324) |
| Screenshot (325) |

|  |
| --- |
| **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 : Palindrom.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ManojLibrary1  {  public class Palindrom  {  public static int Algebra(int n)  {  if (n == 0)  return 1;  else if (n > 7)  return -999;  else if (n < 0)  return -999;  else  {  int fact = 1;  for (int i = 1; i <= n; i++)  {  fact \*= i;  }  return fact;  }  }  public static bool IsAlgebra(int n)  {  int rev = 0, rem, t;  t = n;  while (t > 0)  {  rem = t % 10;  t = t / 10;  rev = rev \* 10 + rem;  }  if (n == rev)  return true;  else  return false;  }  }  } |
| **Code :PalindromeTest.cs** |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using ManojLibrary1;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ManojLibrary1.Tests  {  [TestClass()]  public class PalindromTests  {  [TestMethod()]  public void AlgebraTest\_Zero\_input()  {  //Arrange  int n = 0;  int expected = 1;  //Act  int actual = Palindrom.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void AlgebraTest\_Input\_ZeroTo\_Seven()  {  //Arrange  int n = 5;  int expected = 120;  //Act  int actual = Palindrom.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void AlgebraTest\_GreaterThan\_Seven()  {  //Arrange  int n = 8;  int expected = -999;  //Act  int actual = Palindrom.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void AlgebraTest\_Negative\_Numbers()  {  //Arrange  int n = -4;  int expected = -999;  //Act  int actual = Palindrom.Algebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void AlgebraTest\_Input()  {  //Arrange  int n = 181;  bool expected = true;  //Act  bool actual = Palindrom.IsAlgebra(n);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |
| Output :  Screenshot (328) |