|  |
| --- |
| **Day-18 Assignment**  **By**  **M.Pallavi** |

|  |
| --- |
| 1. What is the use of XML |
| XML stands for Extensible Markup **L**anguage.  XML is used to transport and the data on internet and between different programs  XML can be used for **offloading and reloading of databases**.  XML can be used to store and arrange the data, which can customize your data handling needs  XML can easily be merged with style sheets to create almost any desired output. |

2. Write the points discussed about xml in the class

XML is  Extensible Markup **L**anguage

Xml is Case sensitive Language

Xml has only one root Tag.

Xml consists of user defined tags.

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

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

[

{

"@id": "1",

"@name": "ravi",

"@salary": "7000"

},

{

"@id": "2",

"@name": "rani",

"@salary": "7000"

},

{

"@id": "3",

"@name": "raju",

"@salary": "7000"

},

{

"@id": "4",

"@name": "lakshman",

"@salary": "7000"

},

{

"@id": "5",

"@name": "raghu",

"@salary": "7000"

},

{

"@id": "6",

"@name": "veena",

"@salary": "7000"

},

{

"@id": "7",

"@name": "vani",

"@salary": "7000"

},

{

"@id": "8",

"@name": "anitha",

"@salary": "7000"

}

]

5. Research and write the benefits of JSON over XML ( 2 or 3 points )

* JSON is supported by multiple data structures, unlike XML which only supports type text/string data structure
* JSON contains the data in the form of **key-value**pairs
* It supports all browsers.
* All major JavaScript frameworks offer support JSON.
* Its syntax is straightforward.

|  |
| --- |
| 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 |
| Class Library:  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ClassLibrary1  {  public class Class1  {  public static int Factorial(int n)  {  int fact = 1;  if (n == 0)  return n;  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;  }  }  }    Console application:  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using ClassLibrary1;  namespace pgmonConsoleapp  {  internal class Program  {  static void Main(string[] args)  {  int n;  Console.WriteLine("enter n value");  n=Convert.ToInt32(Console.ReadLine());  Console.WriteLine(Class1.Factorial(n));  Console.ReadLine();  }  }  }  Window application :  sing System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using ClassLibrary2;  namespace WindowsFormsApp3  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void Form1\_Load(object sender, EventArgs e)  {  }  private void label1\_Click(object sender, EventArgs e)  {  }  private void button1\_Click(object sender, EventArgs e)  {  int n = Convert.ToInt32(textBox1.Text);  int result = Algebra.Factorial(n);  textBox2.Text= result.ToString();  }  private void textBox1\_TextChanged(object sender, EventArgs e)  {  }  }  }  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 |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using ClassLibrary2;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ClassLibrary2.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\_Negative\_Input()  {  // Arrange  int n = -31;  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 = 20;  int expected = -999;  // Act  int actual = Algebra.Factorial(n);  // Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void FactorialTest\_Input()  {  // Arrange  int n = 7;  int expected = 5040;  // Act  int actual = Algebra.Factorial(n);  // Assert  Assert.AreEqual(expected, actual);  }        }  }  Output: |
|  |

|  |
| --- |
| Program : 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:  sing System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ClassLibrary2  {  public class Algebra  {  public static int Factorial(int n)  {  if (n == 0)  return 1;  if (n < 0)  return -9999;  if (n > 7)  return 999;  else  {  int fact = 1;  for (int i = 1; i <= n; i++)  {  fact \*= i;  }  return fact;  }  }  public static bool Palindrome(int n)  {  int rev = 0, rem, m;  m = n;  while (m > 0)  {  rem = m % 10;  m = m / 10;  rev = rev \* 10 + rem;  }  if (n == rev)  return true;  else  return false;  }  }  } |
| Output: |

