**DAY 18 ASSIGNMENT PRESENTED**

**BY**

**POTUKANUMA JEEVITHA**

**16-02-2022**

|  |
| --- |
| **1.What is the use of XML?** |
| XML used for Universal data transfer mechanism to send data across different platforms. |

|  |
| --- |
| **2. Write the points discussed about XML in the class.** |
| * XML Stands for eXtensible Markup Language. * It is a Case Sensitive. * XML will have user defined tags. * Only one root tag in XML. * XML used for Universal data transfer mechanism to send data across different platforms. |

|  |
| --- |
| **3. create a simple XML to illustrate:** |
| **a.Tag based XML with 10 products:** |
| **CODE:**  <Product>  <Product>  <Name>Watch</Name>  <Brand>Fastrack</Brand>  <Price>2500</Price>  </Product>  <Product>  <Name>Ring</Name>  <Brand>Gold</Brand>  <Price>50000</Price>  </Product>  <Product>  <Name>Phone</Name>  <Brand>Realme</Brand>  <Price>32000</Price>  </Product>  <Product>  <Name>Dress</Name>  <Brand>H and M</Brand>  <Price>5000</Price>  </Product>  <Product>  <Name>Bike</Name>  <Brand>KTM</Brand>  <Price>250000</Price>  </Product>  <Product>  <Name>Chocolate</Name>  <Brand>DairyMilk</Brand>  <Price>100</Price>  </Product>  <Product>  <Name>Laptop</Name>  <Brand>HP</Brand>  <Price>50000</Price>  </Product>  <Product>  <Name>TV</Name>  <Brand>Samsung</Brand>  <Price>25000</Price>  </Product>  <Product>  <Name>Fridge</Name>  <Brand>Whirlpool</Brand>  <Price>18000</Price>  </Product>  <Product>  <Name>Pen</Name>  <Brand>Cello</Brand>  <Price>25</Price>  </Product>  </Product> |
|  |
| **b. Attribute based XML:** |
| **CODE:**  <Product>  <Product Name="Watch" Brand="Fastrack" Price="2500" />  <Product Name="Ring" Brand="Gold" Price="50000" />  <Product Name="Phone" Brand="Realme" Price="32000" />  <Product Name="Dress" Brand="H and M" Price="5000" />  <Product Name="Bike" Brand="KTM" Price="250000" />  <Product Name="Chocolate" Brand="DairyMilk" Price="100" />  <Product Name="Laptop" Brand="HP" Price="50000" />  <Product Name="TV" Brand="Samsung" Price="25000" />  <Product Name="Fridge" Brand="Whirlpool" Price="18000" />  <Product Name="Pen" Brand="Cello" Price="25" />  </Product> |
|  |

|  |
| --- |
| **4. Convert the above XML to JSON and display the JSON data:** |
| **Tagbased code converted into JSON:** |
| [https://www.freeformatter.com/3.7.0.0/img/minus.gif  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Watch",  "Brand": "Fastrack",  "Price": "2500"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Ring",  "Brand": "Gold",  "Price": "50000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Phone",  "Brand": "Realme",  "Price": "32000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Dress",  "Brand": "H and M",  "Price": "5000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Bike",  "Brand": "KTM",  "Price": "250000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Chocolate",  "Brand": "DairyMilk",  "Price": "100"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Laptop",  "Brand": "HP",  "Price": "50000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "TV",  "Brand": "Samsung",  "Price": "25000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Fridge",  "Brand": "Whirlpool",  "Price": "18000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "Name": "Pen",  "Brand": "Cello",  "Price": "25"  }  ] |
| **Attribute code converted into JSON:** |
| [https://www.freeformatter.com/3.7.0.0/img/minus.gif  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Watch",  "@Brand": "Fastrack",  "@Price": "2500"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Ring",  "@Brand": "Gold",  "@Price": "50000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Phone",  "@Brand": "Realme",  "@Price": "32000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Dress",  "@Brand": "H and M",  "@Price": "5000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Bike",  "@Brand": "KTM",  "@Price": "250000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Chocolate",  "@Brand": "DairyMilk",  "@Price": "100"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Laptop",  "@Brand": "HP",  "@Price": "50000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "TV",  "@Brand": "Samsung",  "@Price": "25000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Fridge",  "@Brand": "Whirlpool",  "@Price": "18000"  },  {https://www.freeformatter.com/3.7.0.0/img/minus.gif  "@Name": "Pen",  "@Brand": "Cello",  "@Price": "25"  }  ] |

|  |
| --- |
| **6. For the below requirement, Create a layered architecture project with separate class library for business logic.**  **a.Create console application**  **b. create desktop application**  **Business Requirement:**  **FIND FACTORIAL OF A NUMBER:**  **0=1**  **Positive number (upto 7) =factorial answer**  **>7 = -999 (as answered)**  **<0 = -9999 (as answered)**  **Put the screenshots of the output and project (solution explorer) screenshot.** |
| **CLASS IN CONSOLE APPLICATION:** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using MathsLibrary;  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();  }  }  } |
| **Algebra class in MathsLibrary:** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MathsLibrary  {  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;  }  }  }  } |
| **Desktop Application:** |
| 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 MathsLibrary;  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();  }  }  } |
|  |
| **CONSOLE APPLICATION OUTPUT:** |
|  |
| **DESKTOP APPLICATION OUTPUT:** |
|  |

|  |
| --- |
| **7. For the above method , implement TDD and write 4 test cases and put the code in word document. Put the screenshot of all test cases failing, make the test cases pass. Put the screenshot.** |
| **ALGEBRA TEST CODE:** |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using MathsLibrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MathsLibrary.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\_nwgative\_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);  }  }  } |
| **ALGEBRA CODE:** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MathsLibrary  {  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;  }  }  }  } |
| **OUTPUT:** |
|  |

|  |
| --- |
| **5. Research and write the benefits of JSON over XML.** |
| 1.JSON occupies less memory compared to XML.  2.JSON is undoubtedly easier to read in its expanded form than XML.  3.JSON much easier to parse. |

|  |
| --- |
| **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.** |
|  |
|  |
|  |
|  |