Day 18 Assignment 16 Feb 2022

By K.SANJAY

|  |
| --- |
| Create a simple xml to illustrate:  a. Tag based xml with 10 products |
| Code |
| <Products>  <product>  <id>111</id>  <name>AUDI</name>  <price>100000</price>  </product>  <product>  <id>112</id>  <name>NISSAN</name>  <price>90000</price>  </product>  <product>  <id>113</id>  <name>BMW</name>  <price>80000</price>  </product>  <product>  <id>114</id>  <name>KIA</name>  <price>70000</price>  </product>  <product>  <id>115</id>  <name>ROLLS ROYCE</name>  <price>250000</price>  </product>  <product>  <id>116</id>  <name>JEEP</name>  <price>60000</price>  </product>  <product>  <id>117</id>  <name>JAGUAR</name>  <price>300000</price>  </product>  <product>  <id>118</id>  <name>BENTLY</name>  <price>400000</price>  </product>  <product>  <id>119</id>  <name>MINI</name>  <price>60000</price>  </product>  <product>  <id>1110</id>  <name>BENZ</name>  <price>500000</price>  </product>  </Products> |
| Screenshot |
|  |

|  |
| --- |
| b. Attribute based xml |
| Code |
| <Products>  <product id="111" name="AUDI" price="100000"></product>  <product id="112" name="NISSAN" price="90000"></product>  <product id="113" name="BMW" price="80000"></product>  <product id="114" name="KIA" price="70000"></product>  <product id="115" name="ROLLS ROYCE" price="250000"></product>  <product id="116" name="JEEP" price="60000"></product>  <product id="117" name="JAGUAR" price="300000"></product>  <product id="118" name="BENTLY" price="400000"></product>  <product id="119" name="MINI" price="60000"></product>  <product id="1110" name="BENZ" price="500000"></product>  </Products> |
| Screenshot |
|  |

|  |
| --- |
| Convert the above xml to JSON and display the JSON data |
| Code |
| <Products>  <product id="111" name="AUDI" price="100000"></product>  <product id="112" name="NISSAN" price="90000"></product>  <product id="113" name="BMW" price="80000"></product>  <product id="114" name="KIA" price="70000"></product>  <product id="115" name="ROLLS ROYCE" price="250000"></product>  <product id="116" name="JEEP" price="60000"></product>  <product id="117" name="JAGUAR" price="300000"></product>  <product id="118" name="BENTLY" price="400000"></product>  <product id="119" name="MINI" price="60000"></product>  <product id="1110" name="BENZ" price="500000"></product>  </Products> |
| Screenshot |
| [  {  "@id": "111",  "@name": "AUDI",  "@price": "100000"  },  {  "@id": "112",  "@name": "NISSAN",  "@price": "90000"  },  {  "@id": "113",  "@name": "BMW",  "@price": "80000"  },  {  "@id": "114",  "@name": "KIA",  "@price": "70000"  },  {  "@id": "115",  "@name": "ROLLS ROYCE",  "@price": "250000"  },  {  "@id": "116",  "@name": "JEEP",  "@price": "60000"  },  {  "@id": "117",  "@name": "JAGUAR",  "@price": "300000"  },  {  "@id": "118",  "@name": "BENTLY",  "@price": "400000"  },  {  "@id": "119",  "@name": "MINI",  "@price": "60000"  },  {  "@id": "1110",  "@name": "BENZ",  "@price": "500000"  }  ] |

|  |
| --- |
| 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 |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using MathematicsLibrary;  namespace Day\_18\_Project\_1  {  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();  }  }  } |
| Output Screenshots |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| 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 = -4;  int expected = -9999;  //Act  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected,actual);  }  [TestMethod()]  public void FactorialTest\_grater\_than\_seven()  {  //Arrange  int n = 8;  int expected = -999;  //Act  int actual = Algebra.Factorial(n);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |
| Screenshot |
|  |
| Code |
| 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)  {  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;  }  }  }  } |
| Screenshot |
|  |

|  |
| --- |
| Write the points discussed about xml in the class |
| * XML Stands for Extensible Mrakup Language * In XML wh have User defined tags * XML is a case sensitive * In XML we hav only on root tag * XML is used for Universal data transfer mchanisim to send dat across different Platforms |

|  |
| --- |
| What is the use of XML |
| * XML provides a standard method to access information, making it easier for applications and devices of all kinds to use * XML is readable and understandable, even by novices, and no more difficult to code than HTML. * Any application that can process XML can use your information, regardless of platform. |

|  |
| --- |
| Research and write the benefits of JSON over XML  ( 2 or 3 points )   * JSON is faster because it is designed specifically for data interchange. JSON encoding is terse, which requires less bytes for transit * JSON can have a substantially lower character count reducing the overhead in data transfers. * JSON is much easier to parse |

|  |
| --- |
| 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 |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MathematicsLibrary  {  public class Algebra  {  public static bool IsPalindrome(int input)  {  int rev=0, rem,m;  m = input;  while (m > 0)  {  rem = m % 10;  rev = (rev \* 10) + rem;  m = m / 10;  }  if (input == rev)  return true;  else    return false;  }  }    } |
|  |
| 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 PalindromeTest\_Input()  {  //Arrange  int input = 131;  bool expected = true;  //Act  bool actual = Algebra.IsPalindrome(input);    //Assert  Assert.AreEqual(expected, actual);    }  }  } |