МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Гомельский государственный технический

университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчёт по лабораторной работе №1

По дисциплине «Объектно-ориентированное программирование»

**«Работа с файловой системой. Потоки ввода/вывода»**

Выполнил:

Студент: группы ИТИ-21

Говядкова П.Ю. Принял: преподаватель Карабчикова Е.А.

Гомель 2021

**Цель работы:** изучить основы XML, получить навыки работы с XML документами средствами .NET (с использованием DOM дерева или SAX парсера согласно варианту), освоить средства валидации XML документов с применением XSD.

**Задание:**

* Разработать библиотеку классов для работы с XML файлом согласно варианту. Для работы с XML использовать DOM или SAX (XML-серилизацию или LINQ2XML использовать нельзя!)
* Разработать *WPF*-приложение, которое может использоваться для открытия, отображения, редактирования и, если это необходимо, сохранения измененного содержимого *XML* файлов для хранения документов согласно варианту.
* Приложение должно быть простым в использовании и включать в себя полную обработку исключений.
* Создать *unit*-тесты для тестирования разработанных классов и приложения.
* Разработать схему проверки *XML* документа, на правильность структуры и введенных знаний с использованием *XSD* схемы. В *XSD* обязательно задавать типы данных, которые принимают узлы, и указывать пределы допустимых значений. В случае если документ не соответствует схеме, выводить соответствующее сообщение.
* При написании и оформлении кода обязательно руководствоваться Code Convention, принципами ООП, SOLID и использовать элементы авто документирования с генерацией соответствующих файлов.
* Приложение должно быть простым в использовании и включать в себя полную обработку исключений.
* Написать unit-тесты и исходные файлы (не менее 30 записей) для тестирования разработанных библиотечных классов, тестирование должно покрывать более 80% библиотечного кода.



Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

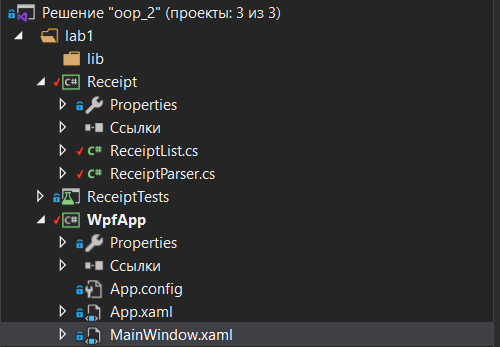


Рисунок 2 – Структура решения

При запуске программы открывается основное меню (Рисунок 2).

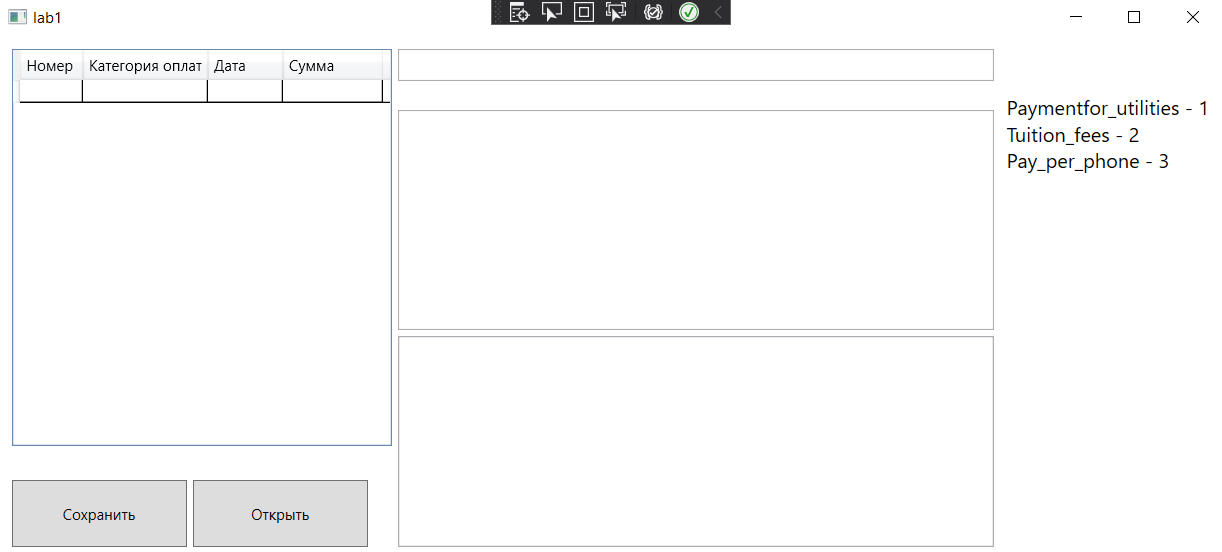


Рисунок 2 – Основное меню программы

После нажатии кнопки “Открыть” запускается диалоговое окно для открытия *xml*-файла (Рисунок 3) и после выбора файла считывается информация и выводится на экран (Рисунок 4).

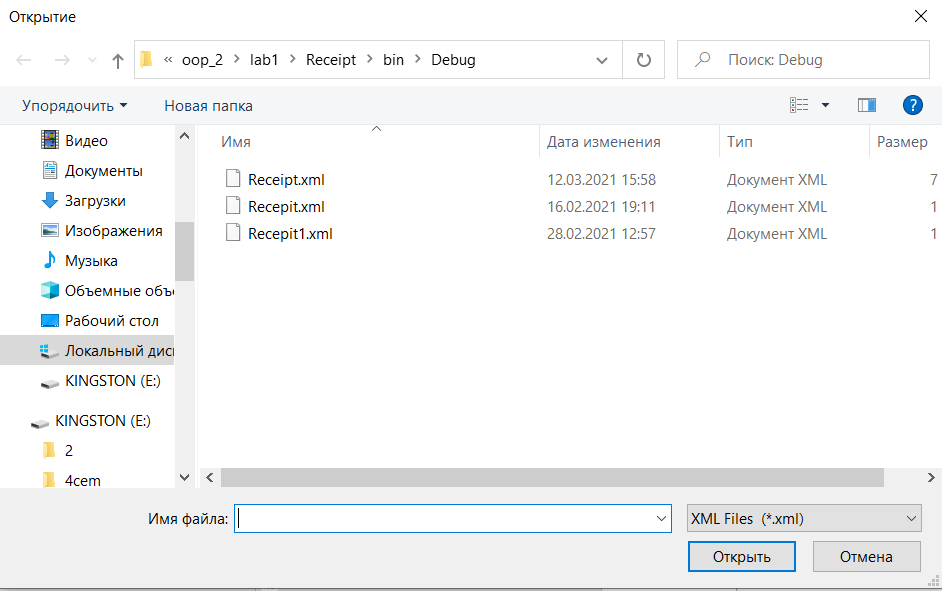


Рисунок 3 – Диалоговое окно выбора файла

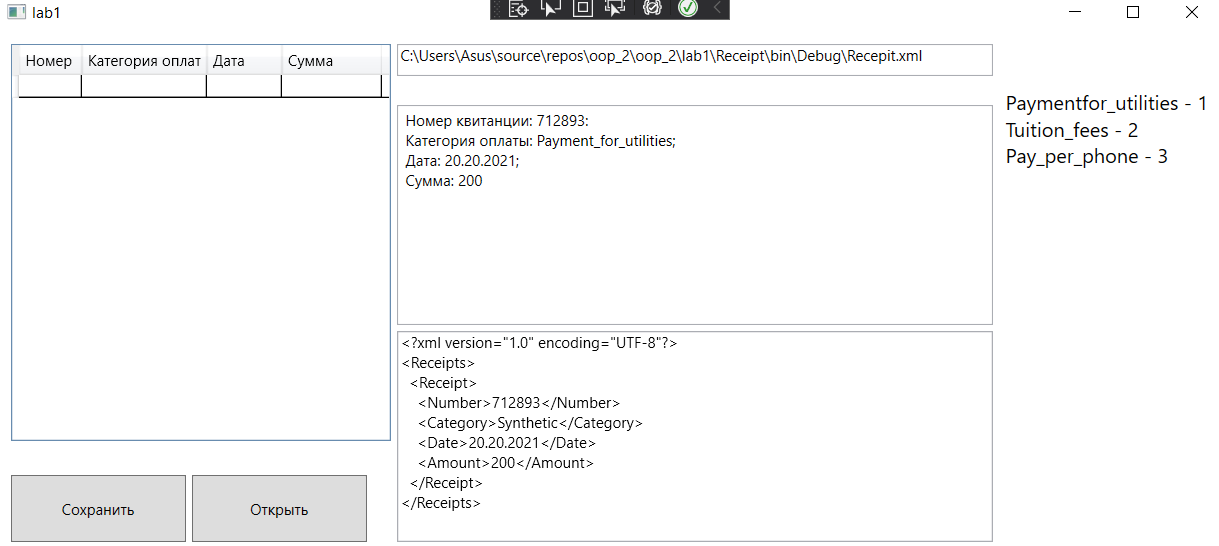


Рисунок 4 – Вывод информации на экран

После добавления данных в таблицу (Рисунок 5) и нажатия кнопки «Сохранить», создается xml-файл с данными из таблицы (Рисунок 6).

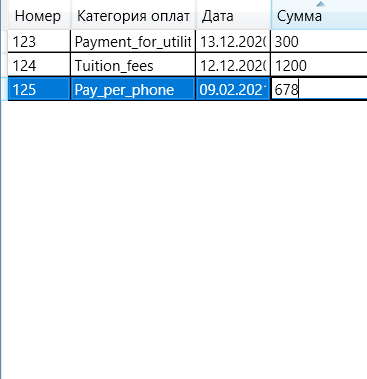


Рисунок 5 – Заполнение таблицы

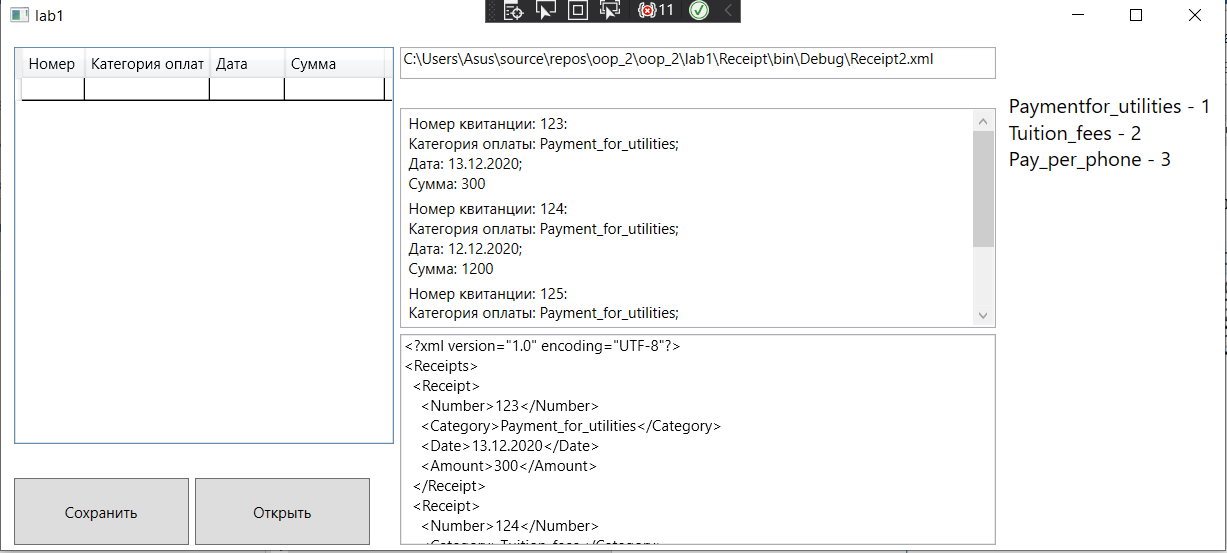


Рисунок 6 – Результат создания нового файла

Результаты прохождения unit-тестов представлены на рисунке 7.

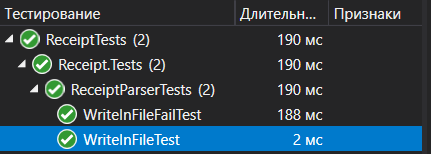


Рисунок 6 – Результаты прохождения тестов.

**Вывод:** изучены основы синтаксиса объектно-ориентированного языка программирования. Реализован класс для работы со строками, XML, DOM-парсер, XSD, обработка исключений.

**ПРИЛОЖЕНИЕ А**

Листинг класса *ReceiptCategory*.

namespace Receipt

{

/// <summary>

///

/// </summary>

public enum ReceiptCategory

{

/// <summary>

/// Оплата коммунальных услуг

/// </summary>

Payment\_for\_utilities = 1,

/// <summary>

/// Плата за обучение

/// </summary>

Tuition\_fees,

/// <summary>

/// Плата за телефон

/// </summary>

Pay\_per\_phone,

}

/// <summary>

///

/// </summary>

public class ReceiptList

{

/// <summary>

/// Получает или устанавливает номер.

/// </summary>

/// <value>

/// The number.

/// </value>

public int Number { get; set; }

/// <summary>

/// Получает или устанавливает категорию услуги.

/// </summary>

/// <value>

/// The category.

/// </value>

public ReceiptCategory Category { get; set; }

/// <summary>

/// Получает или устанавливает дату.

/// </summary>

/// <value>

/// The date.

/// </value>

public string Date { get; set; }

/// <summary>

/// Получает или устанавливает сумму.

/// </summary>

/// <value>

/// The amount.

/// </value>

public double Amount { get; set; }

/// <summary>

/// Конструктор для класса квитанция.

/// </summary>

/// <param name="number">The number.</param>

/// <param name="category">The category.</param>

/// <param name="date">The date.</param>

/// <param name="amount">The amount.</param>

public ReceiptList(int number, ReceiptCategory category, string date, double amount)

{

Number = number;

Category = category;

Date = date;

Amount = amount;

}

/// <summary>

/// Конструктор для класса квитанция.

/// </summary>

public ReceiptList()

{

}

/// <summary>

/// Конвертация в string.

/// </summary>

/// <returns>

/// A <see cref="System.String" /> that represents this instance.

/// </returns>

public override string ToString()

{

return "Номер квитанции: " + Number

+ ":\nКатегория оплаты: " + Category.ToString()

+ ";\nДата: " + Date.ToString()

+ ";\nСумма: " + Amount.ToString();

}

}

}

Листинг класса ReceiptParser.

using System;

using System.Collections.Generic;

using System.Xml;

using System.Xml.Schema;

namespace Receipt

{

/// <summary>

///

/// </summary>

public class ReceiptParser

{

/// <summary>

/// Класс предназначенный для работы с DOM-деревом в с#

/// </summary>

private XmlDocument xmlDocument;

/// <summary>

/// Класс для добавления xsd-схемы к нашему xml-документу

/// </summary>

private XmlReaderSettings xmlReaderSettings;

/// <summary>

/// Вспомогательный класс для создания класса XmlDocument

/// </summary>

private XmlReader reader;

/// <summary>

/// Конструктор для создания парсера при передеаче пути по которому лежит xml-документ

/// </summary>

/// <param name="xmlFilePath">The XML file path.</param>

public ReceiptParser(string xmlFilePath)

{

SetXmlSettings();

reader = XmlReader.Create(xmlFilePath, xmlReaderSettings);

SetXmlDocument(reader);

}

/// <summary>

/// Реализуем метод Dispose т.к. этот класс работает с потоками.

/// </summary>

public void Dispose()

{

DisposeManagedResources();

}

/// <summary>

/// Освобождаем управляемые ресурсы.

/// </summary>

protected virtual void DisposeManagedResources()

{

reader.Dispose();

}

/// <summary>

/// Устанавливает XML - документ.

/// </summary>

private void SetXmlSettings()

{

XmlTextReader xmlTextReader = new XmlTextReader("C:\\Users\\Asus\\source\\repos\\oop\_2\\oop\_2\\lab1\\Receipt\\bin\\Debug\\Recepit.xsd");

XmlSchema xmlSchema = XmlSchema.Read(xmlTextReader, ValidationCallback);

xmlReaderSettings = new XmlReaderSettings();

xmlReaderSettings.Schemas.Add(xmlSchema);

xmlReaderSettings.ValidationType = ValidationType.Schema;

}

/// <summary>

/// Sets the XML document.

/// </summary>

/// <param name="reader">The reader.</param>

private void SetXmlDocument(XmlReader reader)

{

xmlDocument = new XmlDocument();

xmlDocument.Load(reader);

}

/// <summary>

/// Обработчик события, на случай если xml-документ не соотвествует xsd-схеме.

/// </summary>

/// <param name="sender">The sender.</param>

/// <param name="args">The <see cref="ValidationEventArgs"/> instance containing the event data.</param>

private static void ValidationCallback(object sender, ValidationEventArgs args)

{

if (args.Severity == XmlSeverityType.Warning) Console.Write("Warning");

else if (args.Severity == XmlSeverityType.Error) Console.Write("Error");

Console.WriteLine(args.Message);

}

/// <summary>

/// Считываем xml-документ с помощью XPath.

/// </summary>

/// <returns></returns>

/// <exception cref="Exception">

/// Number reading error

/// or

/// Date reading error

/// or

/// or

/// Error reading the amount

/// or

/// Error reading object creation

/// </exception>

public List<ReceiptList> GetReceipts()

{

List<ReceiptList> receipts = new List<ReceiptList>();

int number = 0;

string date = "";

double amount = 0;

ReceiptCategory category = ReceiptCategory.Payment\_for\_utilities;

XmlElement root = xmlDocument.DocumentElement;

XmlNodeList receiptNodeList = root.SelectNodes("Receipt");

foreach (XmlNode node in receiptNodeList)

{

try

{

number = Convert.ToInt32(node.SelectSingleNode("Number").InnerText);

}

catch

{

throw new Exception("Number reading error");

}

try

{

date = node.SelectSingleNode("Date").InnerText;

}

catch

{

throw new Exception("Date reading error");

}

try

{

amount = Convert.ToDouble(node.SelectSingleNode("Amount").InnerText);

if (amount <= 0)

{

throw new Exception();

}

}

catch

{

throw new Exception("Error reading the amount");

}

try

{

receipts.Add(new ReceiptList(number, category, date, amount));

}

catch

{

throw new Exception("Error reading object creation");

}

}

return receipts;

}

/// <summary>

/// Запись в файл.

/// </summary>

/// <param name="receiptList">The receipt list.</param>

/// <param name="filePath">The file path.</param>

public void WriteInFile(List<ReceiptList> receiptList, string filePath)

{

XmlElement root = xmlDocument.DocumentElement;

XmlNodeList receiptNodeList = root.SelectNodes("Receipts");

root = xmlDocument.DocumentElement;

root.RemoveAll();

foreach (ReceiptList elem in receiptList)

{

XmlElement receipt = xmlDocument.CreateElement("Receipt");

XmlElement number = xmlDocument.CreateElement("Number");

XmlElement category = xmlDocument.CreateElement("Category");

XmlElement date = xmlDocument.CreateElement("Date");

XmlElement amount = xmlDocument.CreateElement("Amount");

number.AppendChild(xmlDocument.CreateTextNode(elem.Number.ToString()));

category.AppendChild(xmlDocument.CreateTextNode(elem.Category.ToString()));

date.AppendChild(xmlDocument.CreateTextNode(elem.Date.ToString()));

amount.AppendChild(xmlDocument.CreateTextNode(elem.Amount.ToString()));

receipt.AppendChild(number);

receipt.AppendChild(category);

receipt.AppendChild(date);

receipt.AppendChild(amount);

root = xmlDocument.DocumentElement;

root.AppendChild(receipt);

receipt.RemoveAllAttributes();

Dispose();

xmlDocument.Save(filePath);

}

}

}

}

Листинг XSD-схемы.

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Receipts">

<xs:complexType>

<xs:sequence>

<xs:element name="Receipt" maxOccurs="unbounded" minOccurs="0">

<xs:complexType>

<xs:sequence>

<xs:element type="xs:string" name="Number"/>

<xs:element type="xs:string" name="Category"/>

<xs:element type="xs:string" name="Date"/>

<xs:element type="xs:string" name="Amount"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Листинг класса *MainWindow.xaml*.

using Microsoft.Win32;

using Receipt;

using System;

using System.Collections.Generic;

using System.IO;

using System.Windows;

using System.Windows.Controls;

namespace WpfApp

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

/// <seealso cref="System.Windows.Window" />

/// <seealso cref="System.Windows.Markup.IComponentConnector" />

public partial class MainWindow : Window

{

/// <summary>

/// Список квитанций.

/// </summary>

public List<ReceiptList> receiptList;

/// <summary>

/// Список квитанций.

/// </summary>

public List<ReceiptList> table;

/// <summary>

/// Парсер

/// </summary>

private ReceiptParser parser;

/// <summary>

/// Initializes a new instance of the <see cref="MainWindow"/> class.

/// </summary>

public MainWindow()

{

InitializeComponent();

receiptList = new List<ReceiptList>();

ReceiptTable.AutoGenerateColumns = false;

UpdateTable();

}

/// <summary>

/// Updates the table.

/// </summary>

public void UpdateTable()

{

table = new List<ReceiptList>();

for (var i = 0; i < receiptList.Count; i++)

{

try

{

table.Add(new ReceiptList()

{

Number = receiptList[i].Number,

Category = receiptList[i].Category,

Date = receiptList[i].Date,

Amount = receiptList[i].Amount

});

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

ReceiptTable.AutoGenerateColumns = false;

ReceiptTable.ItemsSource = table;

ReceiptTable.Items.Refresh();

}

/// <summary>

/// Handles the Click event of the Save control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void Save\_Click(object sender, RoutedEventArgs e)

{

try

{

SaveFileDialog dlg = new SaveFileDialog();

dlg.Filter = "XML Files |\*.xml";

if (dlg.ShowDialog() == true)

{

receiptList.Clear();

parser.WriteInFile(table, dlg.FileName);

parser.Dispose();

MessageBox.Show("Файл сохранён");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Handles the Click event of the Open control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="RoutedEventArgs"/> instance containing the event data.</param>

private void Open\_Click(object sender, RoutedEventArgs e)

{

try

{

OpenFileDialog dlg = new OpenFileDialog();

dlg.Filter = "XML Files |\*.xml";

if (dlg.ShowDialog() == true)

{

receiptList.Clear();

MessageBox.Show(dlg.FileName);

parser = new ReceiptParser(dlg.FileName);

Path.Text = dlg.FileName;

try

{

List<ReceiptList> receipts = parser.GetReceipts();

if (receipts != null && receipts.Count != 0)

{

ParserText.Items.Clear();

foreach (ReceiptList receipt in receipts)

ParserText.Items.Add(receipt.ToString());

}

else ParserText.Items.Add("Ошибка считывания данных");

}

catch

{

MessageBox.Show("Ошибка считывания данных");

}

OriginalText.Text = File.ReadAllText(dlg.FileName);

UpdateTable();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Handles the SelectionChanged event of the ReceiptTable control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="SelectionChangedEventArgs"/> instance containing the event data.</param>

private void ReceiptTable\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

/// <summary>

/// Handles the 1 event of the TextBox\_TextChanged control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="TextChangedEventArgs"/> instance containing the event data.</param>

private void TextBox\_TextChanged\_1(object sender, TextChangedEventArgs e)

{

}

/// <summary>

/// Handles the SelectionChanged event of the ParserText control.

/// </summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="SelectionChangedEventArgs"/> instance containing the event data.</param>

private void ParserText\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

}

}