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 №2

По дисциплине «Объектно-ориентированное программирование»

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

Выполнил:

Студент группы ИТП-21

Коркуц С. И.

Принял:

ассистент Карабчикова Е.А.

Гомель 2020

**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.

• Выделенное курсивом поле оформить как перечисление.

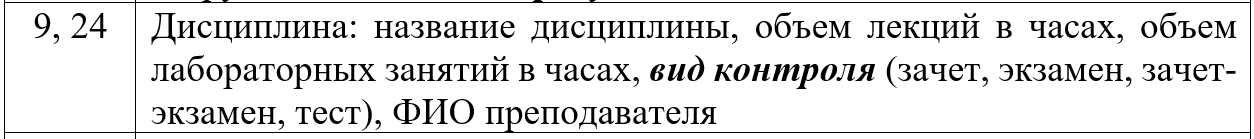


Рисунок 1 – Вариант задания

На рисунке 2 изображена структура решения.

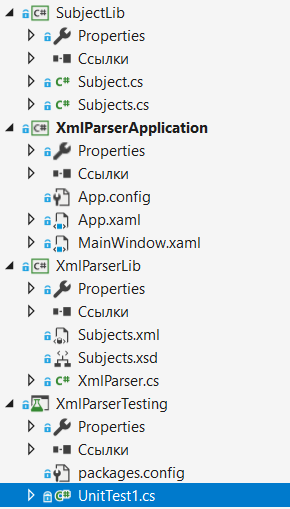


Рисунок 2 – Структура решения

При запуске программы открывается основное меню (Рисунок 3).

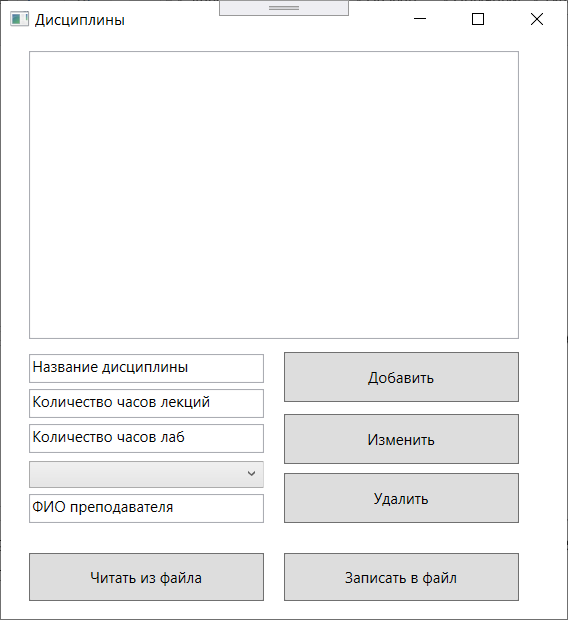


Рисунок 3 – Основное меню программы

После нажатии кнопки “Чтение из файла” создается список дисциплин (рисунок 4).

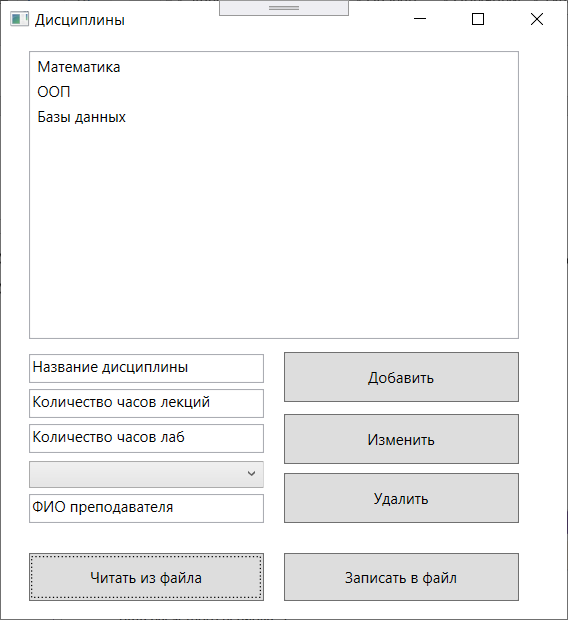


Рисунок 4 – Создание списка дисциплин

При выборе дисциплины, информация о ней появляется в полях окна (Рисунок 5).

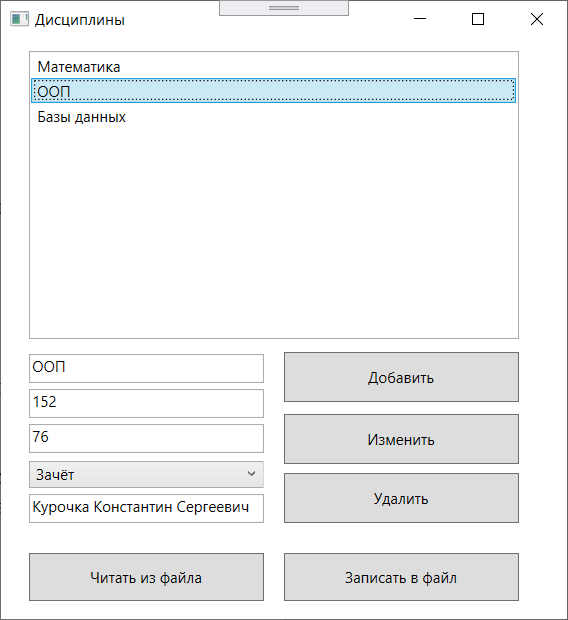


Рисунок 5 – Появление информации об объекте

Попробую изменить значения и добавить новый предмет (рисунок 6).

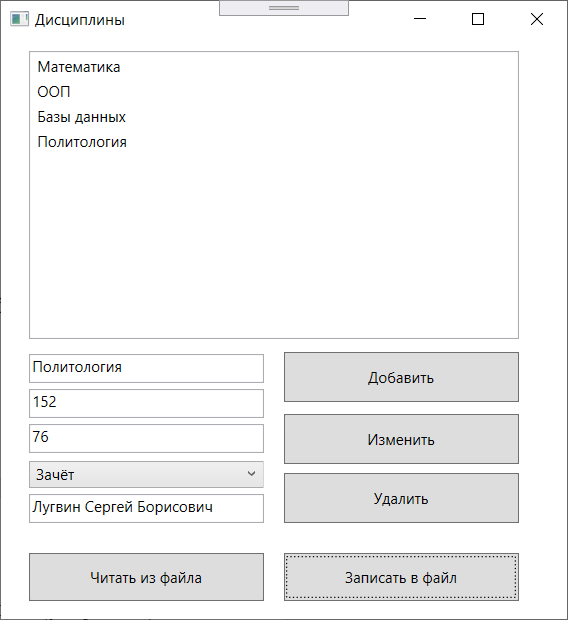


Рисунок 6 – добавление нового предмета

После нажатия кнопки “Записать в файл” все изменения записываются в файл.

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

**ПРИЛОЖЕНИЕ А**

Листинг класса *Subject*.

namespace SubjectLib

{

internal enum ControlType

{

Зачёт,

Экзамен,

ЗачётЭкзамен,

Тест

}

/// <summary>

/// Структура, описывающая дисциплинуы

/// </summary>

internal struct Subject

{

public string Name { get; }

public int LectureHours { get; }

public int LabHours { get; }

public ControlType SubControlType { get; }

public Educator Educator { get; }

/// <summary>

/// Конструктор структуры дисциплина

/// </summary>

/// <param name="name">Название дисциплины</param>

/// <param name="lecHours">Количество лекционных часов</param>

/// <param name="labHours">Количество лабораторных часов</param>

/// <param name="cType">Тип контроля знаний</param>

/// <param name="educator">Преподаватель этой дисциплины</param>

public Subject(string name, int lecHours, int labHours, ControlType cType, Educator educator)

{

Name = name;

LectureHours = lecHours;

LabHours = labHours;

SubControlType = cType;

Educator = educator;

}

public override string ToString()

{

return $"Дисциплина {Name}, часы {LectureHours}/{LabHours}, " +

$"тип контроля {SubControlType.ToString()}, преподаватель: {Educator.ToString()}";

}

public string[] GetAttributes()

{

return new string[] {Name, LectureHours.ToString(), LabHours.ToString(),

SubControlType.ToString("G"), Educator.FirstName, Educator.LastName, Educator.MiddleName };

}

}

/// <summary>

/// Структура, содержащая ФИО преподавателя

/// </summary>

internal struct Educator

{

public string FirstName { get; } // имя

public string LastName { get; } // фамилия

public string MiddleName { get; } // отчество

/// <summary>

/// Конструктор структуры преподаватель

/// </summary>

/// <param name="fName">Имя преподавателя</param>

/// <param name="lName">Фамилия преподавателя</param>

/// <param name="mName">Отчество преподавателя</param>

public Educator(string fName, string lName, string mName)

{

FirstName = fName;

LastName = lName;

MiddleName = mName;

}

public override string ToString()

{

return $"{LastName} {FirstName} {MiddleName}";

}

}

}

**ПРИЛОЖЕНИЕ Б**

Листинг класса *Subjects*.

using System;

using System.Collections.Generic;

namespace SubjectLib

{

public class Subjects

{

List<Subject> subjects;

public int Length

{

get => subjects.Count;

}

public Subjects()

{

subjects = new List<Subject>();

}

private ControlType FromStringToControlType(string cType)

{

switch(cType)

{

case "Экзамен":

return ControlType.Экзамен;

case "Зачёт":

return ControlType.Зачёт;

case "ЗачётЭкзамен":

return ControlType.ЗачётЭкзамен;

case "Тест":

return ControlType.Тест;

default:

throw new Exception("Нет такого способа контроля знаний");

}

}

public void AddSubject(string name, int lecHours, int labHours, string cType,

string eduFirstName, string eduLastName, string eduMiddleName)

{

subjects.Add(new Subject(name, lecHours, labHours, FromStringToControlType(cType),

new Educator(eduFirstName, eduLastName, eduMiddleName)));

}

public int SearchIndexByName(string name)

{

int? ind = null;

int i = 0;

foreach (Subject s in subjects)

{

if (s.Name == name)

{

ind = i;

break;

}

i += 1;

}

if (ind != null)

return (int)ind;

else

throw new Exception("Нет дисциплины с таким именем");

}

public void RemoveSubject(string name)

{

subjects.RemoveAt(SearchIndexByName(name));

}

public void EditSubject(string name, int lecHours, int labHours, string cType,

string eduFirstName, string eduLastName, string eduMiddleName)

{

subjects[SearchIndexByName(name)] = new Subject(name, lecHours, labHours,

FromStringToControlType(cType), new Educator(eduFirstName, eduLastName, eduMiddleName));

}

public string[] this[int index]

{

get

{

return subjects[index].GetAttributes();

}

}

}

}

**ПРИЛОЖЕНИЕ В**

Листинг XSD-схемы *Subjects.xsd*.

<?xml version="1.0" encoding="utf-8"?>

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

<xs:element name="Subjects">

<xs:complexType>

<xs:sequence>

<xs:element name="Subject" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element name="Educator">

<xs:complexType>

<xs:attribute name="firstName"/>

<xs:attribute name="lastName"/>

<xs:attribute name="middleName"/>

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute name="name" type="xs:string"/>

<xs:attribute name="lecHours">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minExclusive value="0"/>

<xs:maxExclusive value="200"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="labHours">

<xs:simpleType>

<xs:restriction base="xs:integer">

<xs:minExclusive value="0"/>

<xs:maxExclusive value="200"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

<xs:attribute name="controlType">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="Экзамен"/>

<xs:enumeration value="Зачёт"/>

<xs:enumeration value="ЗачётЭкзамен"/>

<xs:enumeration value="Тест"/>

</xs:restriction>

</xs:simpleType>

</xs:attribute>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

**ПРИЛОЖЕНИЕ Г**

Листинг класса *MainWindow.xaml*.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using SubjectLib;

using XmlParserLib;

namespace XmlParserApplication

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Subjects subjects;

public MainWindow()

{

InitializeComponent();

subjects = new Subjects();

}

/// <summary>

/// Записывает имена дисциплин в список

/// </summary>

void RecordToList()

{

subjectsList.Items.Clear();

for(int i = 0; i < subjects.Length; i++)

{

subjectsList.Items.Add(subjects[i][0]);

}

}

/// <summary>

/// Чтение из файла

/// </summary>

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

/// <param name="e"></param>

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

subjects = new Subjects();

XmlParser.ReadFromXmlFile(subjects);

subjectsList.SelectedItem = null;

RecordToList();

}

/// <summary>

/// Запись выбранного элемента в текстовые поля

/// </summary>

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

/// <param name="e"></param>

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

{

try

{

string[] attributes = subjects[subjects.SearchIndexByName(subjectsList.SelectedItem.ToString())];

subjectNameBox.Text = attributes[0];

lecHoursBox.Text = attributes[1];

labHoursBox.Text = attributes[2];

switch (attributes[3])

{

case "Зачёт":

controlTypeBox.SelectedItem = controlTypeBox.Items[0];

break;

case "Экзамен":

controlTypeBox.SelectedItem = controlTypeBox.Items[1];

break;

case "ЗачётЭкзамен":

controlTypeBox.SelectedItem = controlTypeBox.Items[2];

break;

case "Тест":

controlTypeBox.SelectedItem = controlTypeBox.Items[3];

break;

default:

throw new Exception("Что-то пошло не так");

}

educatorBox.Text = $"{attributes[5]} {attributes[4]} {attributes[6]}";

}

catch(NullReferenceException){ } //При чтении из файла список очищается и

//объект SelectedItem ни на что не указывает

}

/// <summary>

/// Записывает дисциплины в файл

/// </summary>

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

/// <param name="e"></param>

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

{

XmlParser.WriteToXmlFile(subjects);

}

/// <summary>

/// Изменение данных о дисциплине

/// </summary>

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

/// <param name="e"></param>

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

try

{

int lecHours = Convert.ToInt32(lecHoursBox.Text);

int labHours = Convert.ToInt32(labHoursBox.Text);

if (lecHours < 0 || lecHours > 200 || labHours < 0 || labHours > 200)

{

throw new Exception("Превышен диапазон часов занятий");

}

string[] educator = educatorBox.Text.Split();

subjects.EditSubject(subjectNameBox.Text, lecHours, labHours, controlTypeBox.Text,

educator[1], educator[0], educator[2]);

}

catch

{

MessageBox.Show("Введены неверный данные");

}

}

/// <summary>

/// Удаление записи

/// </summary>

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

/// <param name="e"></param>

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

subjects.RemoveSubject(subjectsList.SelectedItem.ToString());

RecordToList();

}

/// <summary>

/// Добавить запись

/// </summary>

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

/// <param name="e"></param>

private void Button\_Click\_4(object sender, RoutedEventArgs e)

{

string[] educator = educatorBox.Text.Split();

subjects.AddSubject(subjectNameBox.Text, Convert.ToInt32(lecHoursBox.Text),

Convert.ToInt32(labHoursBox.Text), controlTypeBox.Text, educator[1], educator[0], educator[2]);

RecordToList();

}

}

}