# **Отчёт по лабораторной работе №8**

## **Предмет: Microsoft .NET**

## **Выполнил: ст. гр. О-18-ПРИ-рпс-Б, Лядов. В.С.**

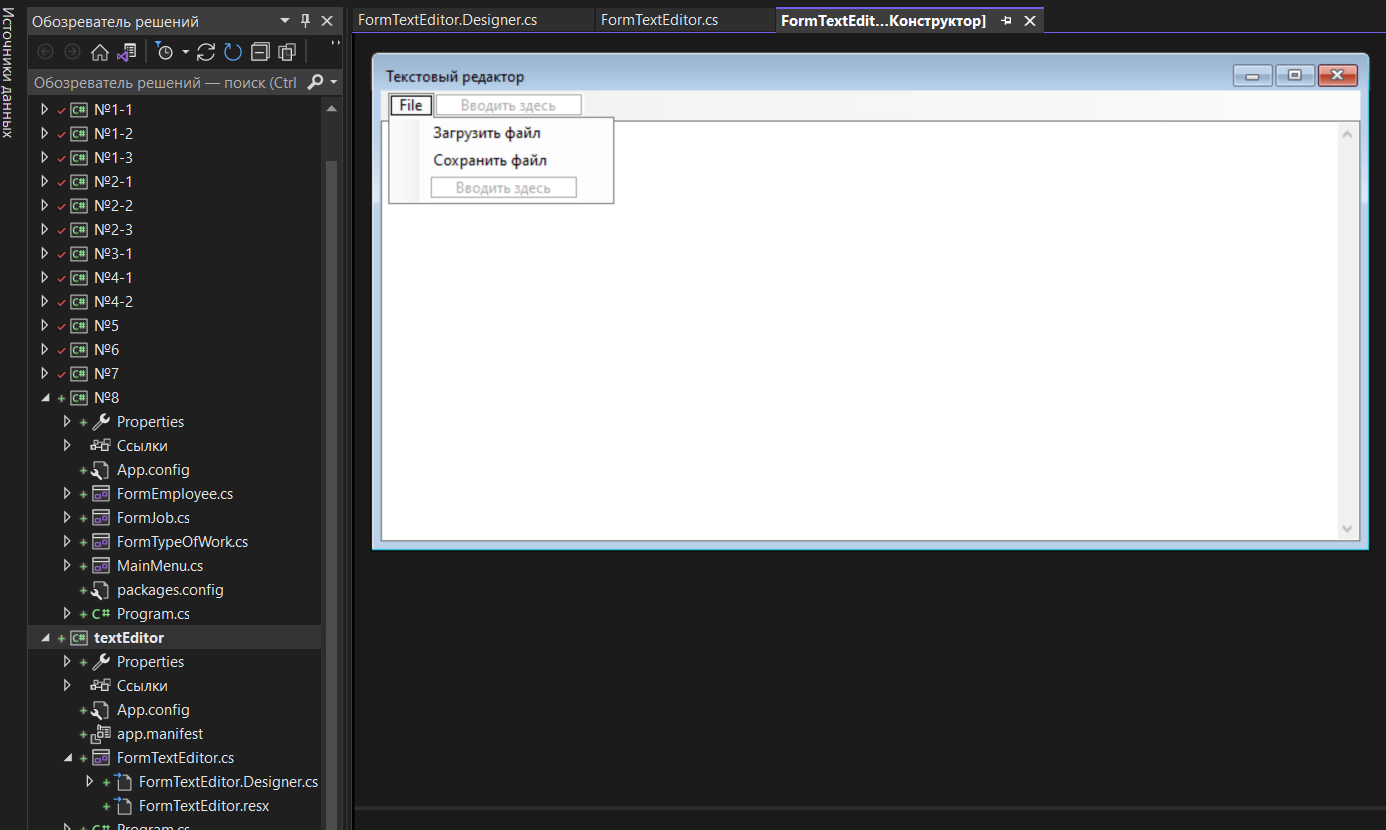
## Постановка задач

1. Разработать Windows Forms приложение для просмотра,  редактирования и сохранения текстового файла. Программа должна автоматически перезагружать содержимое загруженного файла, если оно изменилось другим процессом.
2. Добавьте в предыдущую лабораторную работу возможность загружать и сохранять коллекции в файлы, использую сериализацию и десериализацию в 3-х форматах: XML, JSON и двоичный.

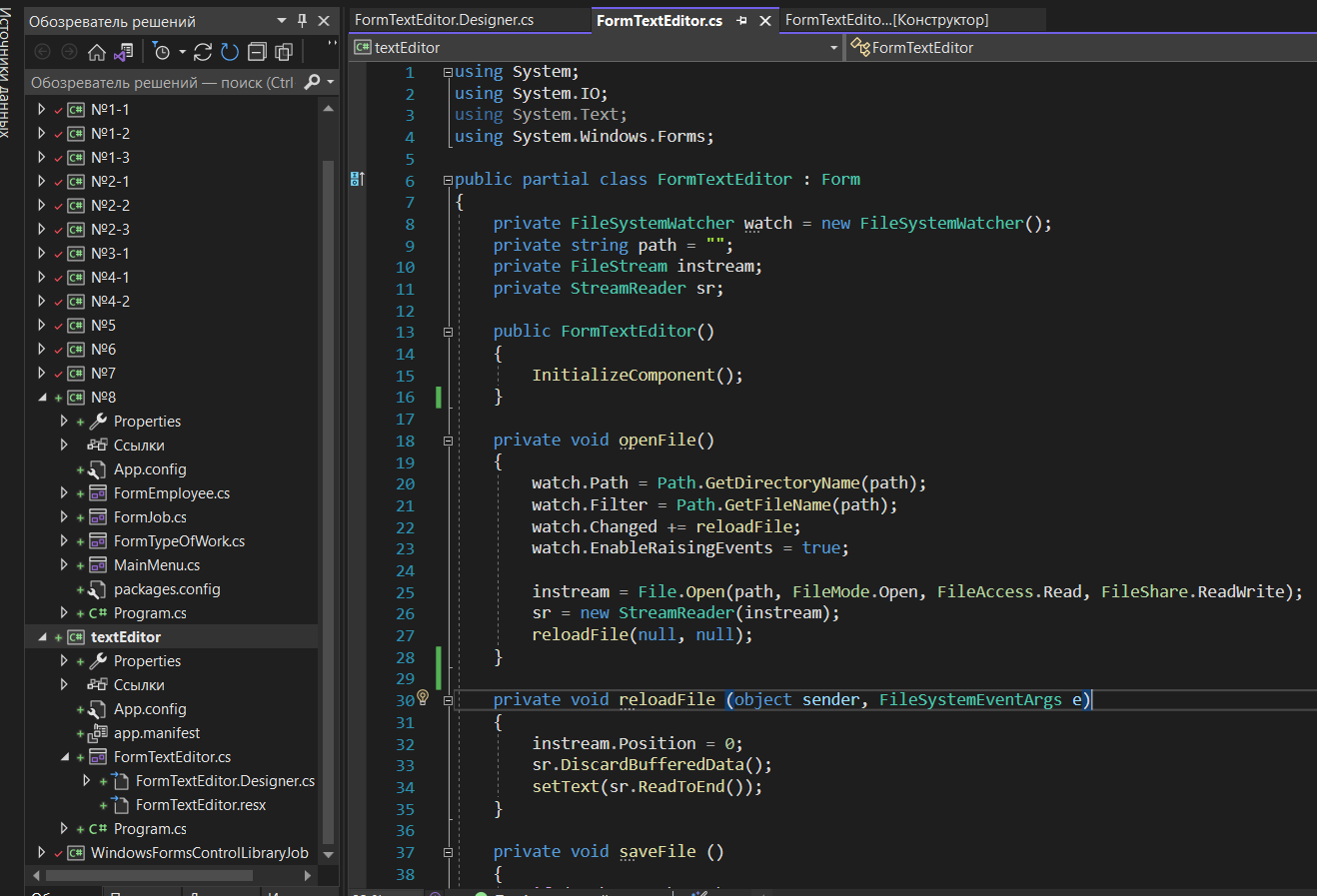
## Практическая часть

Фотоотчёт

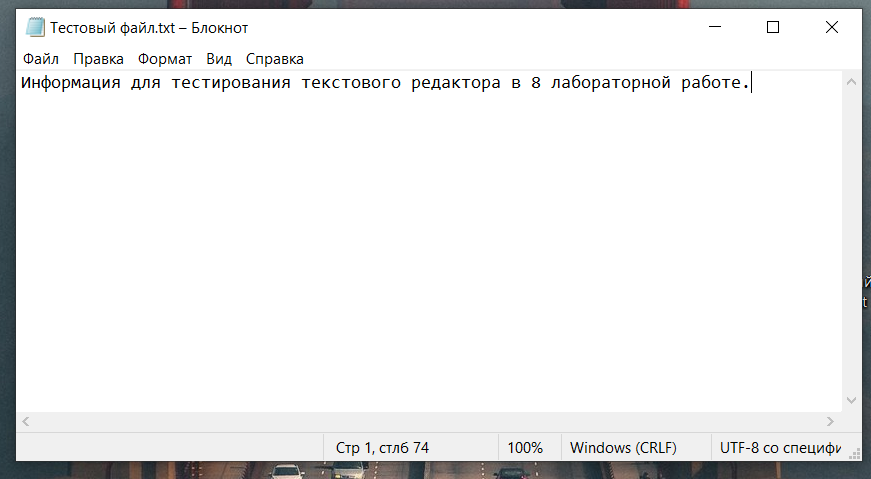
*Разработать Windows Forms приложение для просмотра, редактирования и сохранения текстового файла. Программа должна автоматически перезагружать содержимое загруженного файла, если оно изменилось другим процессом:*



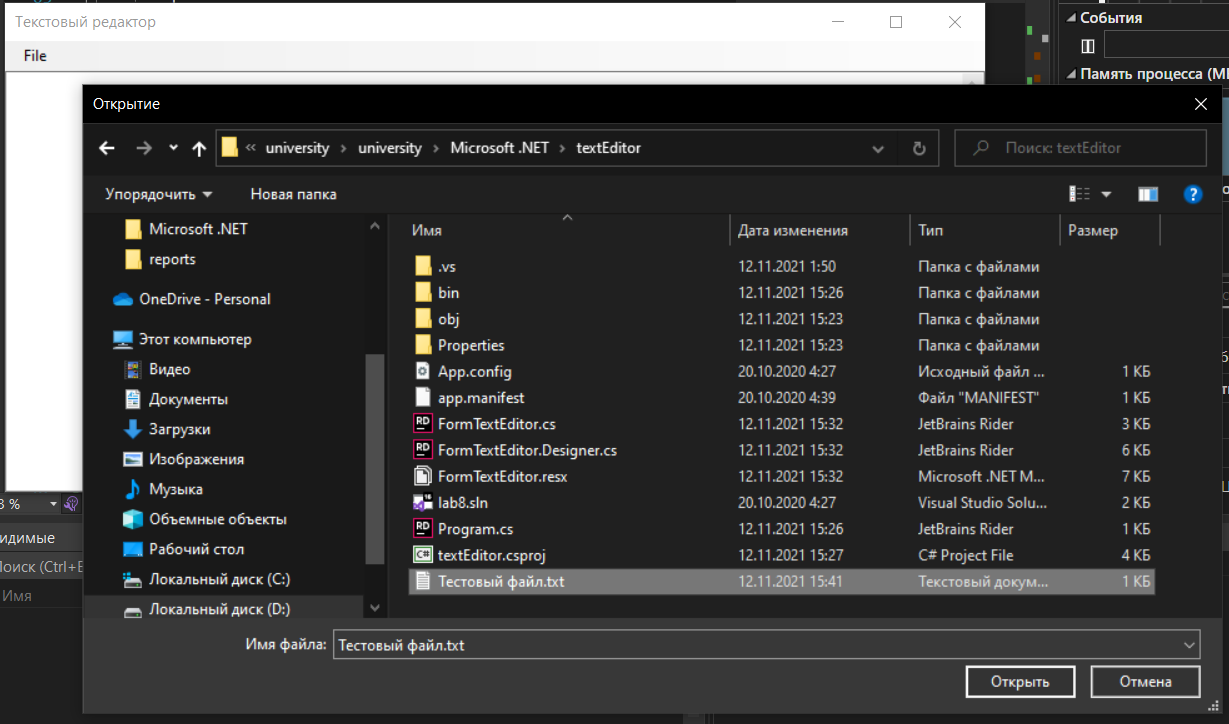
*Рис 1. Внешний вид формы*



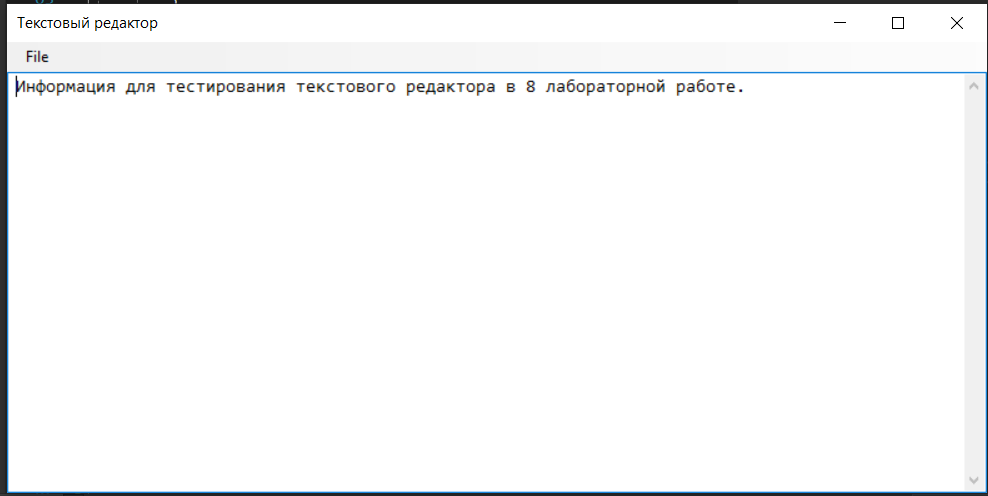
*Рис 2. Определение класса, описание методов открытия файла*

*. *

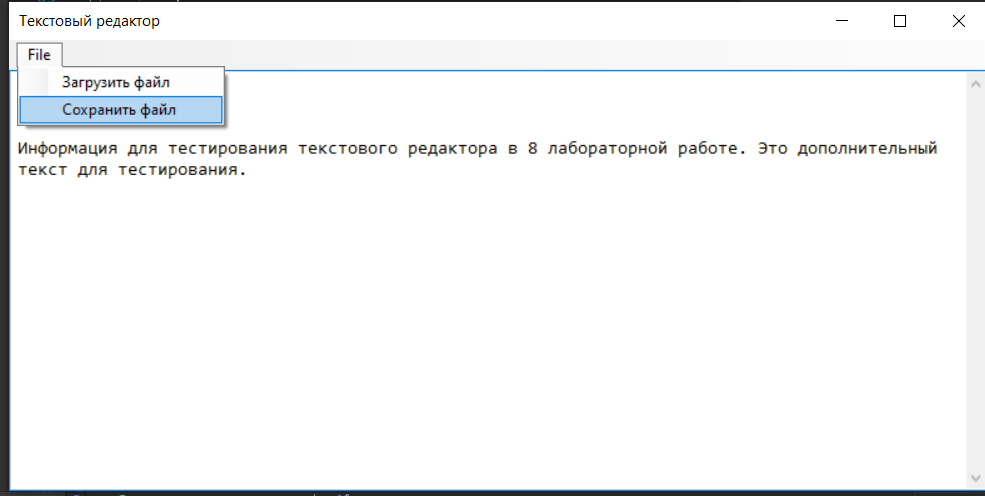
*Рис 3. Создадим тестовый файл через текстовый редактор windows 10.*

**

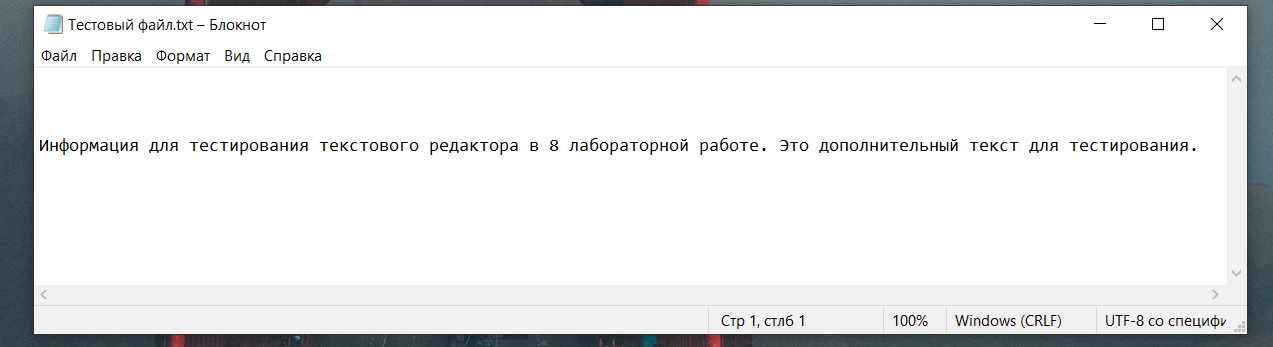
*Рис 4. Откроем файл через пользовательский текстовый редактор*

**

*Рис 5. Результат загрузки*

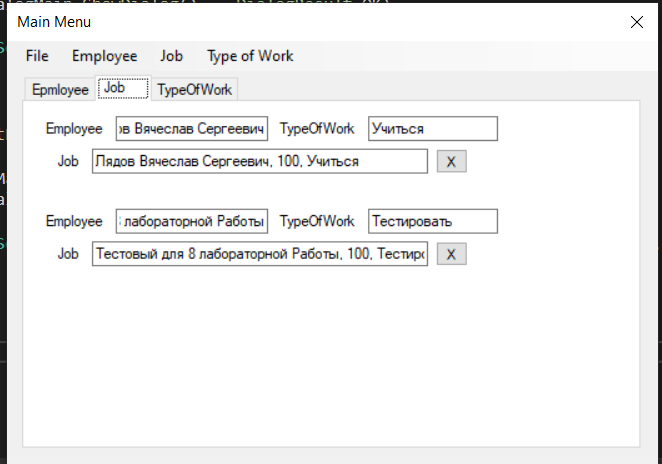
**

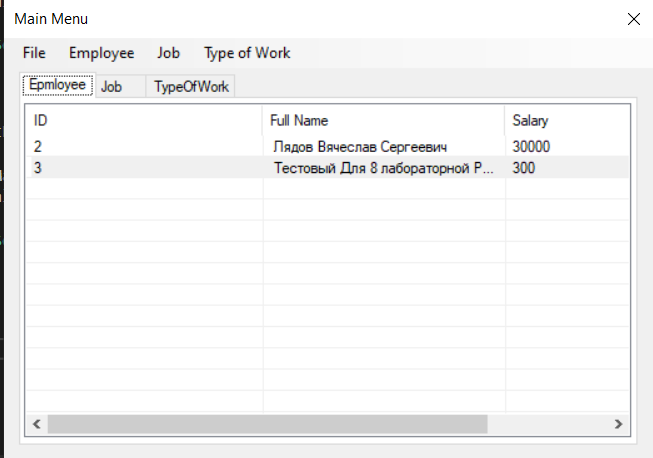
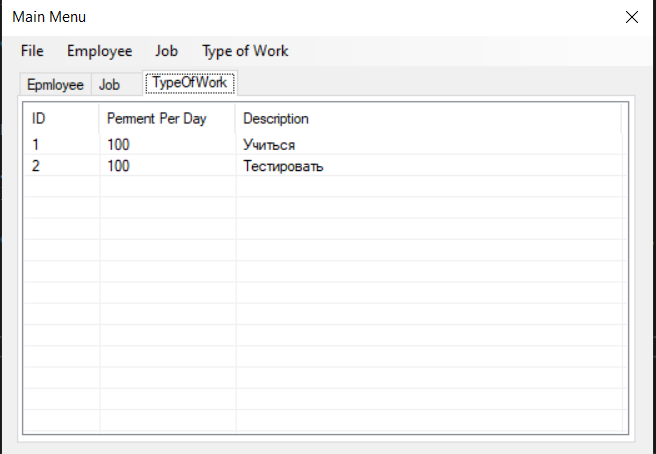
*Рис 6. Добавим изменения и сохраним файл.*

**

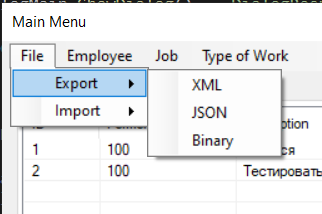
*Рис 7. Проверим, что все изменения сохранились.*

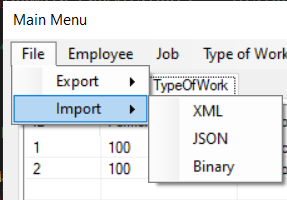
Добавьте в предыдущую лабораторную работу возможность загружать и сохранять коллекции в файлы, использую сериализацию и десериализацию в 3-х форматах: XML, JSON и двоичный:



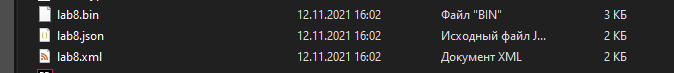
  


*Рис 8-10. Заполним тестовыми данными.*

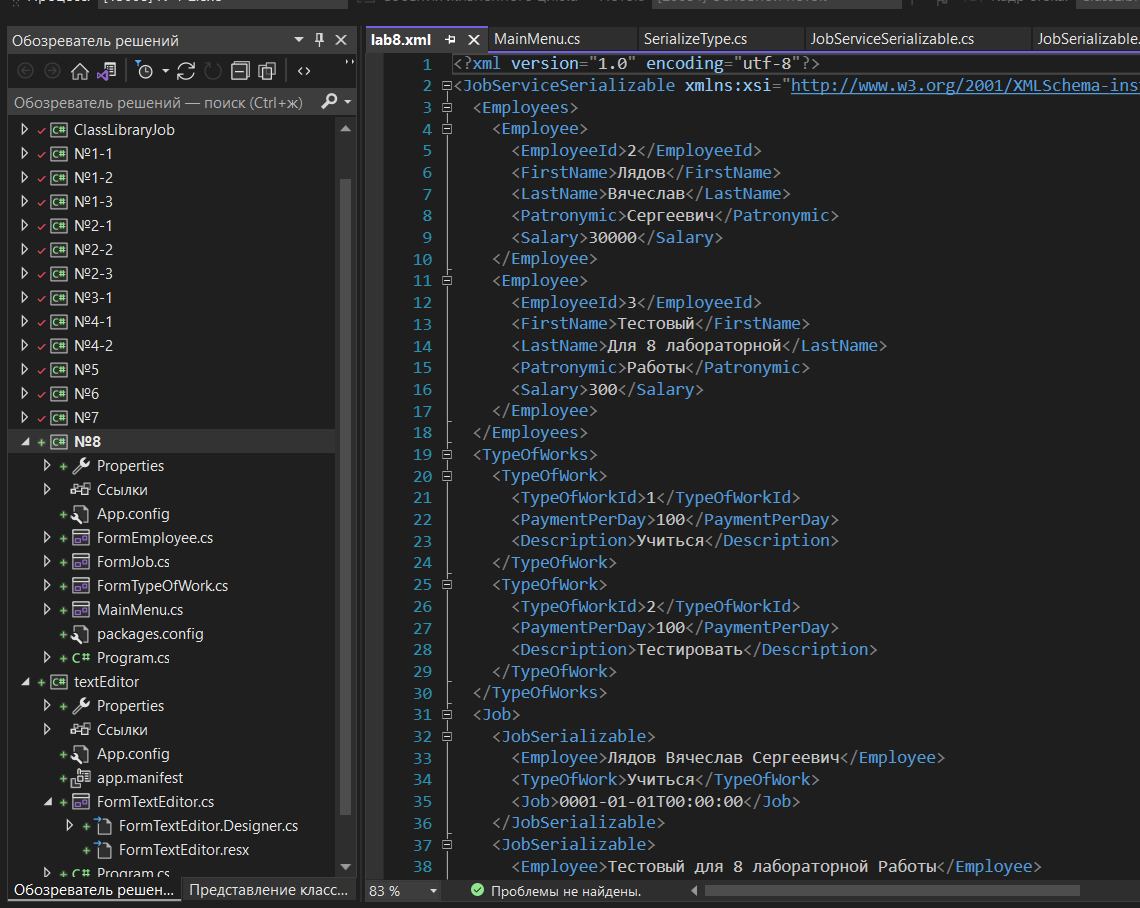
**

**

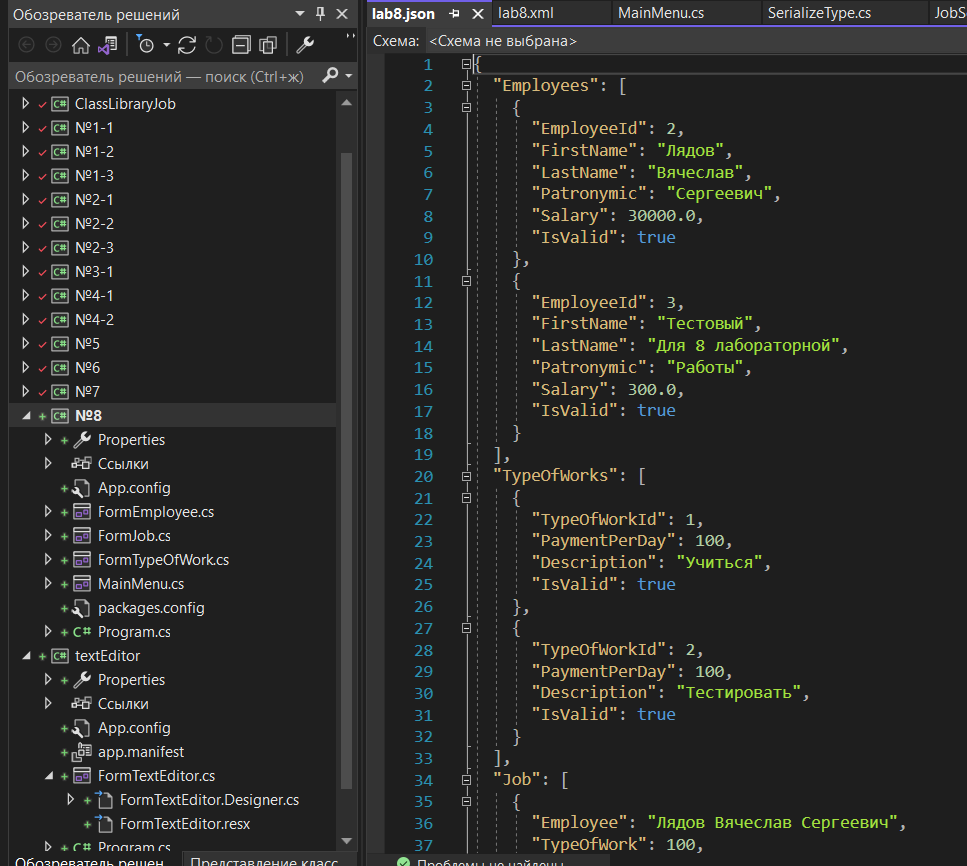
*Рис 9. Визуальный интерфейс для сохранения и загрузки файлов.*



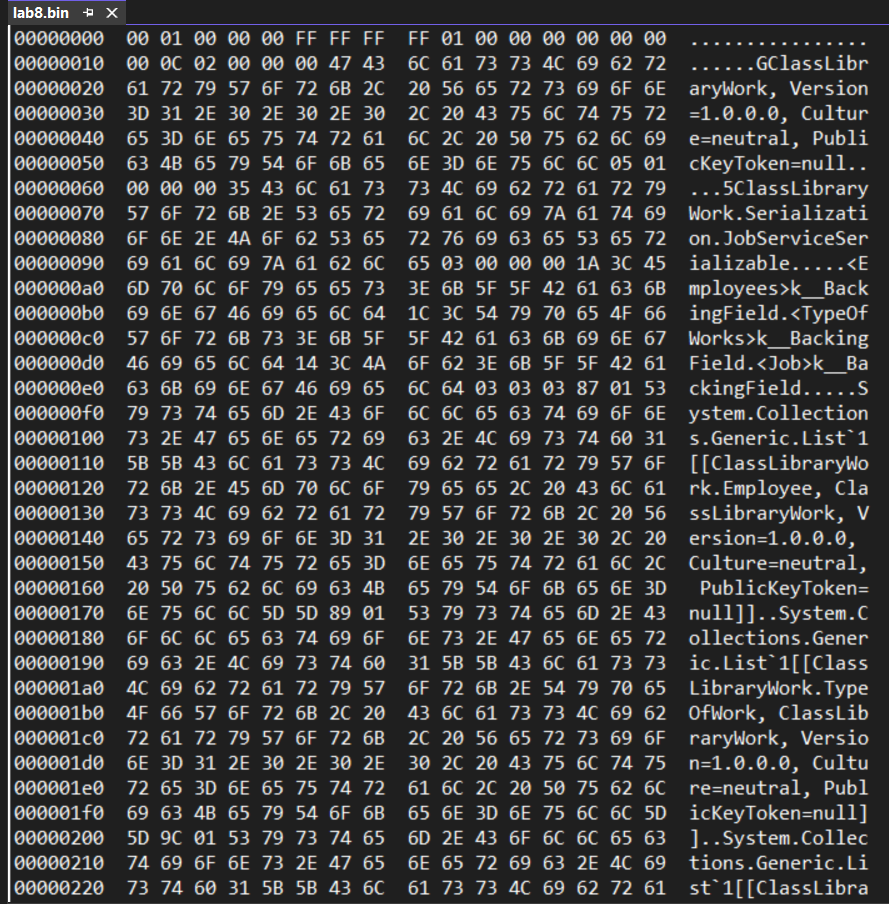
*Рис 10. Выгрузим в файлы.*



*Рис 11. Содержимое XML после выгрузки.*



*Рис 12. Содержимое JSON после выгрузки.*

**

*Рис 13. Содержимое BIN после выгрузки. ☺*

Листинг 1

Код главной формы текстового редактора, содержит описание работы загрузки, сохранения файла.

using System;

using System.IO;

using System.Text;

using System.Windows.Forms;

public partial class FormTextEditor : Form

{

private FileSystemWatcher watch = new FileSystemWatcher();

private string path = "";

private FileStream instream;

private StreamReader sr;

public FormTextEditor()

{

InitializeComponent();

}

private void openFile()

{

watch.Path = Path.GetDirectoryName(path);

watch.Filter = Path.GetFileName(path);

watch.Changed += reloadFile;

watch.EnableRaisingEvents = true;

instream = File.Open(path, FileMode.Open, FileAccess.Read, FileShare.ReadWrite);

sr = new StreamReader(instream);

reloadFile(null, null);

}

private void reloadFile (object sender, FileSystemEventArgs e)

{

instream.Position = 0;

sr.DiscardBufferedData();

setText(sr.ReadToEnd());

}

private void saveFile ()

{

if (path.Length > 0)

{

File.WriteAllText(path, editor.Text, sr.CurrentEncoding);

}

}

delegate void SetTextCallback(string text);

private void setText(string text)

{

if (editor.InvokeRequired)

{

SetTextCallback d = new SetTextCallback(setText);

Invoke(d, new object[] { text });

}

else

{

editor.Text = text;

}

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

if (path.Length > 0)

{

sr.Close();

instream.Dispose();

}

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Control && e.KeyCode == Keys.S)

{

saveFile();

}

}

private void uploadToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (fileDialog.ShowDialog() == DialogResult.OK)

{

path = fileDialog.FileName;

openFile();

}

}

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

saveFile();

}

}

Листинг 2.

Содержимое JobServiceSerializable

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Xml.Serialization;

using Newtonsoft.Json;

namespace ClassLibraryWork.Serialization

{

[Serializable]

public class JobServiceSerializable

{

public List<Employee> Employees { get; set; } = new List<Employee>();

public List<TypeOfWork> TypeOfWorks { get; set; } = new List<TypeOfWork>();

public List<JobSerializable> Job { get; set; } = new List<JobSerializable>();

public static void Save(string fileName, SerializeType type)

{

var jobServiceSerializable = new JobServiceSerializable();

var jobService = Human.Instance;

foreach (var employee in jobService.Employees)

{

jobServiceSerializable.Employees.Add(employee);

}

foreach (var typeOfWork in jobService.TypeOfWorks)

{

jobServiceSerializable.TypeOfWorks.Add(typeOfWork);

}

foreach (var Job in jobService.Jobs)

{

jobServiceSerializable.Job.Add(new JobSerializable

{

Employee = Job.Worker.EmployeeId,

TypeOfWork = Job.Position.PaymentPerDay,

StartDate = Job.StartDate,

EndDate = Job.EndDate

});

}

switch (type)

{

case SerializeType.XML:

XmlSerializer xmlSerializer = new XmlSerializer(typeof(JobServiceSerializable));

using (StreamWriter xmlStreamWriter = new StreamWriter(fileName))

{

xmlSerializer.Serialize(xmlStreamWriter, jobServiceSerializable);

}

break;

case SerializeType.JSON:

using (StreamWriter jsonStreamWriter = File.CreateText(fileName))

{

JsonSerializer jsonSerializer = new JsonSerializer { Formatting = Formatting.Indented };

jsonSerializer.Serialize(jsonStreamWriter, jobServiceSerializable);

}

break;

case SerializeType.Binary:

BinaryFormatter formatter = new BinaryFormatter();

using (FileStream binaryFileStream = new FileStream(fileName, FileMode.OpenOrCreate))

{

formatter.Serialize(binaryFileStream, jobServiceSerializable);

}

break;

default:

throw new ArgumentOutOfRangeException(nameof(type), type, null);

}

}

public static void Load(string fileName, SerializeType type)

{

JobServiceSerializable jobServiceSerializable;

switch (type)

{

case SerializeType.XML:

XmlSerializer xmlSerializer = new XmlSerializer(typeof(JobServiceSerializable));

StreamReader streamReader = new StreamReader(fileName);

jobServiceSerializable = (JobServiceSerializable)xmlSerializer.Deserialize(streamReader);

break;

case SerializeType.JSON:

StreamReader jsonStreamReader = File.OpenText(fileName);

JsonSerializer jsonSerializer = new JsonSerializer();

jobServiceSerializable = (JobServiceSerializable)jsonSerializer.Deserialize(jsonStreamReader, typeof(JobServiceSerializable));

break;

case SerializeType.Binary:

BinaryFormatter formatter = new BinaryFormatter();

FileStream binaryFileStream = new FileStream(fileName, FileMode.Open);

jobServiceSerializable = (JobServiceSerializable)formatter.Deserialize(binaryFileStream);

break;

default:

throw new ArgumentOutOfRangeException(nameof(type), type, null);

}

var jobService = Human.Instance;

var jobServiceEmployee = jobService.Employees.ToList();

var jobServiceTypeOfWork = jobService.TypeOfWorks.ToList();

var jobServiceJobs = jobService.Jobs.ToList();

foreach (var jobServiceEmployees in jobServiceEmployee)

{

jobService.RemoveEmployee(jobServiceEmployees.EmployeeId);

}

foreach (var jobServiceTypeOfWorks in jobServiceTypeOfWork)

{

jobService.RemoveTypeOfWork(jobServiceTypeOfWorks.PaymentPerDay);

}

foreach (var jobServiceJobss in jobServiceJobs)

{

jobService.RemoveJob(jobServiceJobss);

}

var employes = new Dictionary<int, Employee>();

var typeOfWorkes = new Dictionary<int, TypeOfWork>();

int maxClientId = 0;

foreach (var employee in jobServiceSerializable.Employees)

{

if (employee.EmployeeId > maxClientId) maxClientId = employee.EmployeeId;

employes.Add(employee.EmployeeId, employee);

jobService.AddEmployee(employee);

}

foreach (var typeofwork in jobServiceSerializable.TypeOfWorks)

{

typeOfWorkes.Add(typeofwork.PaymentPerDay, typeofwork);

jobService.AddTypeOfWork(typeofwork);

}

foreach (var job in jobServiceSerializable.Job)

{

jobService.AddJob(new Job

{

Worker = employes[job.Employee],

Position = typeOfWorkes[job.TypeOfWork],

StartDate = job.StartDate,

EndDate = job.EndDate

});

}

Employee.NewEmployeeId = maxClientId;

}

}

}

Листинг 3.

Содержимое *JobSerializable*

using System;

namespace ClassLibraryWork.Serialization

{

[Serializable]

public class JobSerializable

{

public int Employee { get; set; }

public int TypeOfWork { get; set; }

/// <summary>

/// Дата начала проживания

/// </summary>

public DateTime StartDate { get; set; } = DateTime.Now;

/// <summary>

/// Дата окончания проживания

/// </summary>

public DateTime EndDate { get; set; } = DateTime.Now;

}

}

Листинг 4.

Содержимое *SerializeType*

namespace ClassLibraryWork.Serialization

{

public enum SerializeType

{

XML,

JSON,

Binary

}

}

Листинг 5.

Методы обработки события нажатия ЛКМ для сохранения/загрузки XML/JSON/Binary.

private void exportXMLToolStripMenuItem\_Click(object sender, EventArgs e)

{

saveFileDialogMain.Filter = "XML-файлы|\*.xml|Все файлы|\*.\*";

if (saveFileDialogMain.ShowDialog() == DialogResult.OK)

{

JobServiceSerializable.Save(saveFileDialogMain.FileName, SerializeType.XML);

}

}

private void exportJSONToolStripMenuItem\_Click(object sender, EventArgs e)

{

saveFileDialogMain.Filter = "JSON-файлы|\*.json|Все файлы|\*.\*";

if (saveFileDialogMain.ShowDialog() == DialogResult.OK)

{

JobServiceSerializable.Save(saveFileDialogMain.FileName, SerializeType.JSON);

}

}

private void exportBinaryToolStripMenuItem\_Click(object sender, EventArgs e)

{

saveFileDialogMain.Filter = "Двоичные файлы|\*.bin|Все файлы|\*.\*";

if (saveFileDialogMain.ShowDialog() == DialogResult.OK)

{

JobServiceSerializable.Save(saveFileDialogMain.FileName, SerializeType.Binary);

}

}

private void importXMLToolStripMenuItem1\_Click(object sender, EventArgs e)

{

openFileDialogMain.Filter = "XML-файлы|\*.xml|Все файлы|\*.\*";

if (openFileDialogMain.ShowDialog() == DialogResult.OK)

{

JobServiceSerializable.Load(openFileDialogMain.FileName, SerializeType.XML);

}

}

private void importJSONToolStripMenuItem1\_Click(object sender, EventArgs e)

{

openFileDialogMain.Filter = "JSON-файлы|\*.json|Все файлы|\*.\*";

if (openFileDialogMain.ShowDialog() == DialogResult.OK)

{

JobServiceSerializable.Load(openFileDialogMain.FileName, SerializeType.JSON);

}

}

private void importBinaryToolStripMenuItem1\_Click(object sender, EventArgs e)

{

openFileDialogMain.Filter = "Двоичные файлы|\*.bin|Все файлы|\*.\*";

if (openFileDialogMain.ShowDialog() == DialogResult.OK)

{

JobServiceSerializable.Load(openFileDialogMain.FileName, SerializeType.Binary);

}

}

Ссылка на GitHub на проект: