

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

Дисциплина: «Разработка программных систем»

Отчёт по лабораторной работе № 4

**Задание:** Написать приложение с использованием технологии WinForms для хранения списка должников.

**Выполнил студент группы №485:**

Зобнин Илья Михайлович

**Проверили:**

Иван Григорьевич Корниенко

Алексей Константинович Федин

Санкт-Петербург

2020

# Постановка задачи

Необходимо написать приложение с использованием технологии WinForms, реализующее хранение списка должников. Программа должна позволять добавлять новые сущности с использованием интерфейса и редактировать существующие. Сущности, добавленные в программу, должны сохраняться между запусками приложения. Для хранения данных необходимо использовать СУБД SqLite. Необходимо предусмотреть возможность сохранения списка существующих сущностей в файл.

# Особые ситуации

* При добавлении или изменении записи пользователь не сможет внести изменения в таблицу, пока не заполнит все поля данными верного формата, а также все обязательные поля.
* При сохранении данных в файл программа берёт их не из базы данных, а из таблицы в программе.

# Математические методы и алгоритмы решения задач

Согласно постановке задачи, для реализации взаимодействия пользовательского интерфейса с базой данных был использован паттерн MVP, блок-схема которого представлена на Рисунке 1.

Слева представлена модель IModel. Это некий шаблон, от которого наследуются Debtor и Debt. Они обязаны реализовать в себе свойства для возврата имени таблицы, первичного ключа, а также ключа для связки таблиц.

Далее представлен репозиторий. У него есть обобщённый параметр T, который, однако, ограничен наследниками IModel. Репозиторий связывается с базой данных, с помощью TableName модели берёт данные из нужной таблицы, назначает первичный ключ, беря его название из той же модели, а также при необходимости связывает две таблицы с помощью свойств RelationKey IModel.

Правее расположен интерфейс IPresenter, содержащий метод для запуска приложения, а также его имплементация Presenter. Presenter служит связующим между View и репозиторием. При запросе от View на получение данных Presenter вызывает метод репозитория GetAllRecords сначала для таблицы должников, подставляя в обобщённый тип Debtor, а потом для таблицы долгов, подставляя уже Debt. Потом с помощью метода репозитория ConnectTwoTables связывает эти таблицы и передаёт во View. При запросе от View на сохранение в базу данных для каждой из таблиц вызывается уже метод UpdateDB репозитория.

Далее расположен IView и View. Во View содержатся множество классов для добавления, редактирования и удаления записей в таблице.

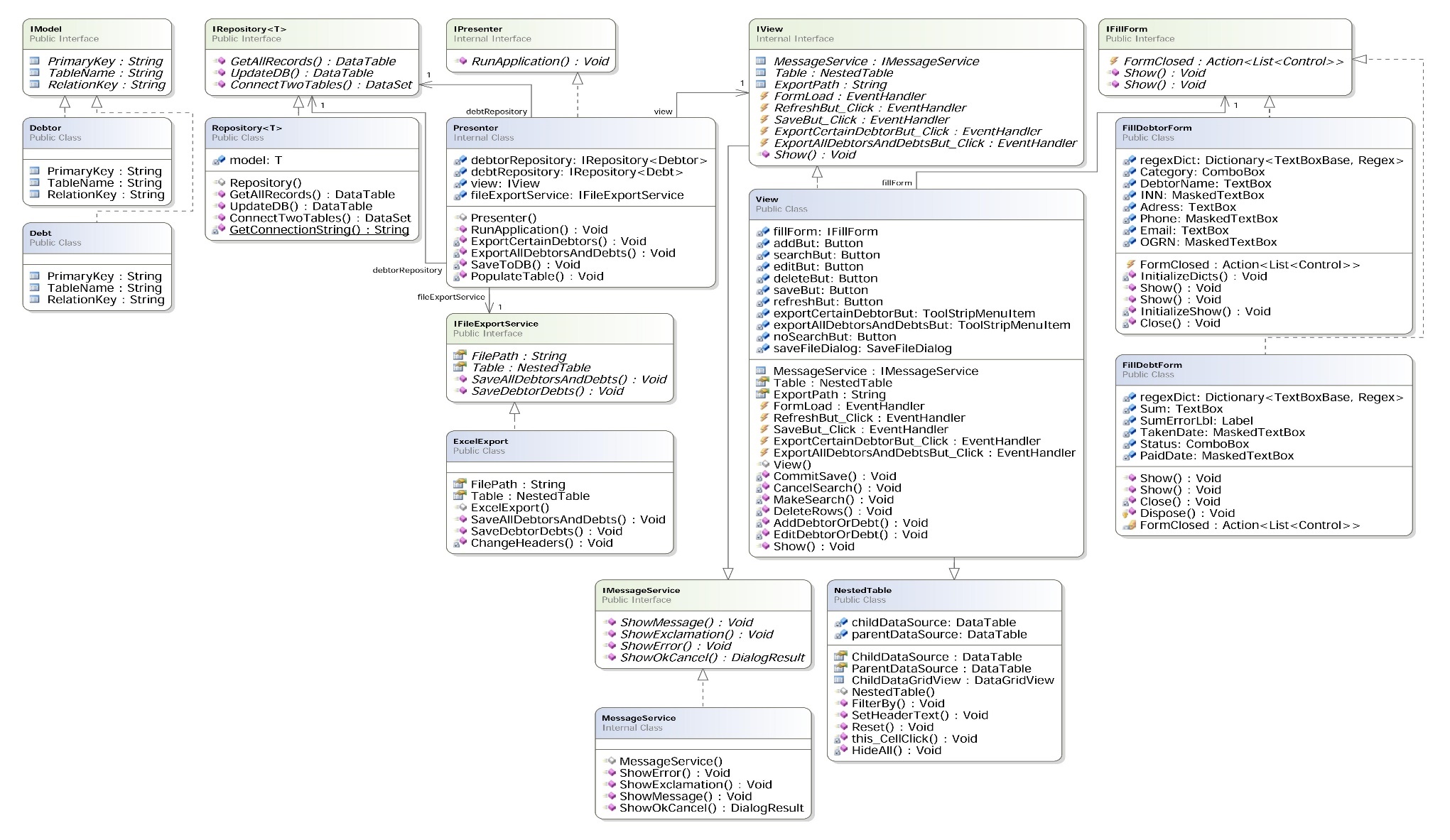


Рисунок 1 – Блок-схема реализации паттерна MVP

# Форматы представления данных

Программа использует следующие переменные:

Таблица 1 – Переменные, используемы в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| debtorRepository | IRepository<Debtor> | Содержит методы управления базой данных для модели Debtor |
| debtRepository | IRepository<Debt> | Содержит методы управления базой данных для модели Debt |
| view | IView | Содержит члены и методы для управления пользовательским интерфейсом |
| messageService | IMessageService | Сервис уведомлений пользователя |
| fileExportService | IFileExportService | Сервис экспорта в файл |
| table | NestedTable | Вложенная таблица должников и их долгов |
| regexDict | Dictionary<string, Regex> | Словарь проверки формата введённых данных |

# Структура программы

Программа разделена на двадцать четыре класса. Program отвечает за запуск программы, ThisApplication за внедрение зависимостей и вызов метода для показа основной формы, IRepository, IModel, IView, IPresenter и их имплементации, а также вспомогательные классы служат для реализации паттерна и общей работоспособности программы.

Класс Program.cs:

Таблица 2 – Функции, составляющие Класс Program.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Main | Запуск программы |

Класс ThisApplication.cs:

Таблица 3 – Функции, составляющие класс ThisApplication.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Run | Внедрение зависимостей и запуск приложения |

Класс View.cs:

Таблица 4 – Функции, составляющие класс View.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| CommitSave | Запрос пути и сохранение в файл |
| MakeSearch | Поиск по таблице |
| CancelSearch | Сброс критериев поиска |
| DeleteRows | Удаление из таблицы выбранных строк |
| AddDebtorOrDebt | Добавление записи о должнике или долге |
| EditDebtorOrDebt | Изменение записи о должнике или долге |

Класс IMessageService.cs:

Таблица 5 – Функции, составляющие класс IMessageService.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| ShowMessage | Уведомление |
| ShowExclamation | Предупреждение |
| ShowError | Уведомление об ошибке |
| ShowOkCancel | Диалог с выбором «Ок» или «Отмена» |

Класс NestedTable.cs:

Таблица 6 – Функции, составляющие класс NestedTable.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| FilterBy | Фильтрация записей таблицы по заданному критерию |
| Reset | Сброс критериев поиска |
| this\_CellClick | Отображение или скрытие дочерней таблицы |

Класс IFileExportService.cs:

Таблица 7 – Функции, составляющие класс IFileExportService.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| SaveAllDebtorsAndDebts | Сохранение в файл полностью обоих таблиц |
| SaveDebtorDebts | Сохранение выделенного должника и его долгов |

Класс IFillForm.cs

Таблица 8 – Функции, составляющие класс IFillForm.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Show | Показ формы добавления/редактирования записи |
| FormClosed | Передача списка значений для вставки в таблицу при закрытии формы |

Класс IRepository.cs

Таблица 9 – Функции, составляющие класс IRepository.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| GetAllRecords | Получение из базы данных все записи таблицы |
| UpdateDB | Обновление таблицы в базе данных из заданного источника |
| ConnectTwoTables | Связь двух заданных таблиц |

Класс Presenter.cs

Таблица 10 – Функции, составляющие класс Presenter.cs

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Show | Вызов метода View для показа формы |
| PopulateTable | Заполнение таблицы View данными, полученными из Repository |
| SaveToDB | Сохранение таблицы View с базу данных |
| ExportCertainDebtors | Экспорт в файл выделенные записи о должниках и их долгах |
| ExportAllDebtorsAndDebts | Экспорт в файл всех записей о должниках и их долгах |

# Описание хода выполнения лабораторной работы

* При получении пользовательского ввода необходимо было добавить проверку, на его корректность.
* При выполнении лабораторной работы были созданы тесты для проверки правильности ввода и вывода данных в/из базы данных.

# Результат работы программы

В результате работы программа выводит вложенную таблицу должников и их долгов

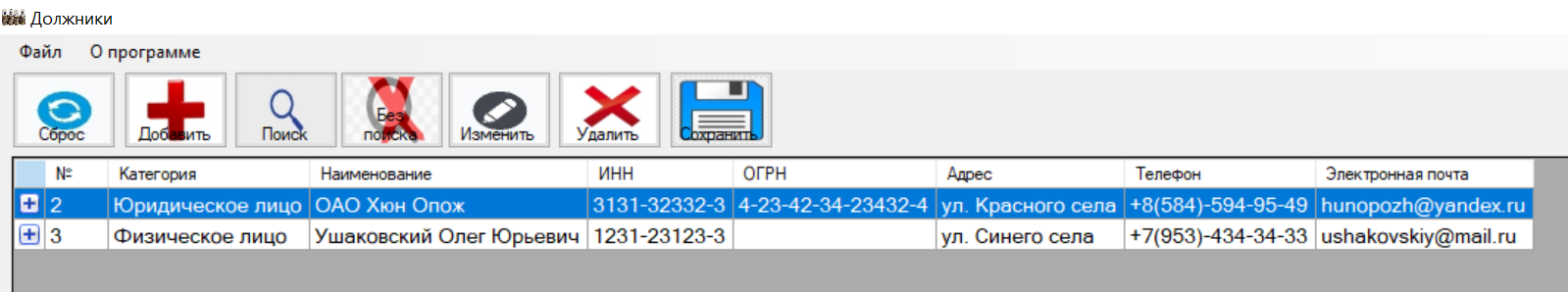


Рисунок 2 – Основное окно программы.

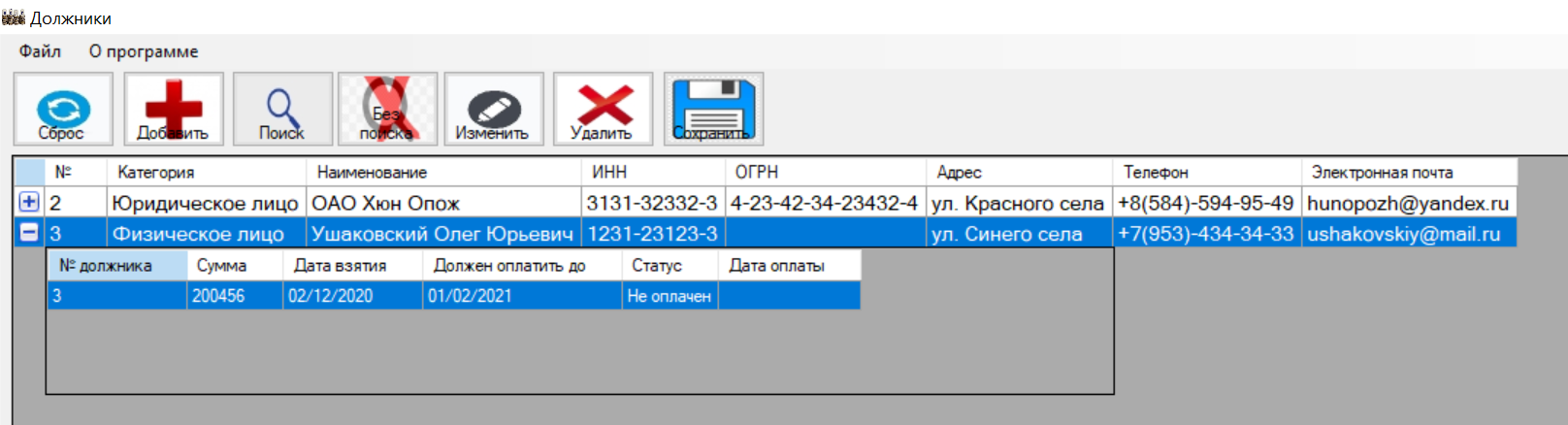


Рисунок 3 – Вложенная запись о долгах.

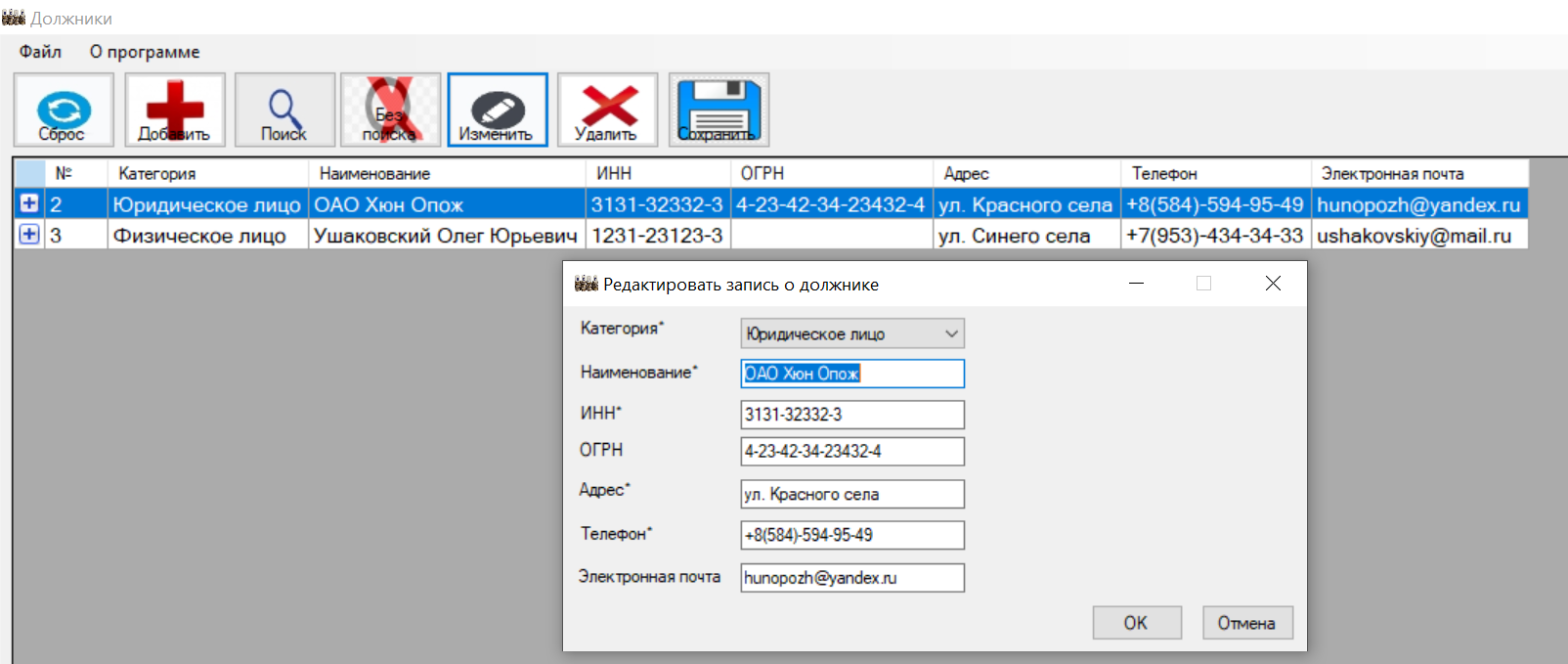


Рисунок 4 – Изменение записи о должнике.

# Текст программы

**[--- Начало программы ---]**

**// Program.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

namespace Lab4

{

static class Program

{

[STAThread]

static void Main()

{

ThisApplication.Run();

}

}

}

**// ThisApplication.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Autofac;

using Lab4.Model;

using Lab4.Presenter;

using Lab4.Repository;

using Lab4.View;

using Lab4.View.MessageService;

using System.Windows.Forms;

namespace Lab4

{

public static class ThisApplication

{

public static void Run()

{

var builder = new ContainerBuilder();

builder.RegisterType<Debtor>();

builder.RegisterType<Debt>();

builder.RegisterType<Repository<Debtor>>().As<IRepository<Debtor>>();

builder.RegisterType<Repository<Debt>>().As<IRepository<Debt>>();

builder.RegisterType<ExcelExport>().As<IFileExportService>();

builder.RegisterType<Presenter.Presenter>().As<IPresenter>();

builder.RegisterType<MessageService>().As<IMessageService>();

builder.RegisterType<View.View>().As<IView>();

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

var container = builder.Build();

container.Resolve<IPresenter>().RunApplication();

}

}

}

**// IView.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Lab4.View.MessageService;

using System;

using System.Data;

namespace Lab4.View

{

interface IView

{

IMessageService MessageService { get; }

NestedTable.NestedTable Table { get; }

DataSet TableDataSet { get; set; }

string ExportPath { get; }

event EventHandler FormLoad;

event EventHandler RefreshBut\_Click;

event EventHandler SaveBut\_Click;

event EventHandler ExportCertainDebtorBut\_Click;

event EventHandler ExportAllDebtorsAndDebtsBut\_Click;

void Show();

}

}

**// View.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Lab4.View.FillingForms;

using Lab4.View.MessageService;

using System;

using System.Data;

using System.Reflection;

using System.Windows.Forms;

namespace Lab4.View

{

public partial class View : Form, IView

{

public IMessageService MessageService { get; }

public NestedTable.NestedTable Table { get; private set; }

public DataSet TableDataSet

{

get => Table.ParentDataSource.DataSet;

set

{

Table.ParentDataSource = value.Tables[0];

Table.ChildDataSource = value.Tables[1];

Table.ChildDataGridView.Columns[0].Visible = false;

Table.SetHeaderText();

}

}

public string ExportPath { get; private set; }

IFillForm fillForm;

public View(IMessageService messageService)

{

InitializeComponent();

this.Text = this.ProgramName;

this.MessageService = messageService;

SetEventHandlers();

}

#region Events

public event EventHandler FormLoad;

public event EventHandler RefreshBut\_Click;

public event EventHandler SaveBut\_Click;

public event EventHandler ExportCertainDebtorBut\_Click;

public event EventHandler ExportAllDebtorsAndDebtsBut\_Click;

#endregion

private void SetEventHandlers()

{

this.Load += (sender, e) => FormLoad?.Invoke(sender, e);

refreshBut.Click += (sender, e) => RefreshBut\_Click?.Invoke(sender, e);

addBut.Click += AddDebtorOrDebt;

searchBut.Click += MakeSearch;

noSearchBut.Click += CancelSearch;

editBut.Click += EditDebtorOrDebt;

deleteBut.Click += DeleteRows;

saveBut.Click += (sender, e) => SaveBut\_Click?.Invoke(sender, e);

exportCertainDebtorBut.Click += (sender, e) => CommitSave(ExportCertainDebtorBut\_Click, sender, e);

exportAllDebtorsAndDebtsBut.Click += (sender, e) => CommitSave(ExportAllDebtorsAndDebtsBut\_Click, sender, e);

}

public new void Show()

{

Application.Run(this);

}

public string ProgramName

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyProductAttribute)attributes[0]).Product;

}

}

}

}

**// AboutBox.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Reflection;

using System.Windows.Forms;

namespace Lab4.View

{

partial class AboutBox : Form

{

public AboutBox()

{

InitializeComponent();

this.Text = String.Format("О программе {0}", AssemblyTitle);

this.labelProductName.Text = AssemblyProduct;

this.labelVersion.Text = String.Format("Версия {0}", AssemblyVersion);

this.labelCopyright.Text = AssemblyCopyright;

this.labelCompanyName.Text = AssemblyCompany;

this.textBoxDescription.Text = AssemblyDescription;

}

#region Методы доступа к атрибутам сборки

public string AssemblyTitle

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

if (attributes.Length > 0)

{

AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];

if (titleAttribute.Title != "")

{

return titleAttribute.Title;

}

}

return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);

}

}

public string AssemblyVersion

{

get

{

return Assembly.GetExecutingAssembly().GetName().Version.ToString();

}

}

public string AssemblyDescription

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyDescriptionAttribute)attributes[0]).Description;

}

}

public string AssemblyProduct

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyProductAttribute)attributes[0]).Product;

}

}

public string AssemblyCopyright

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;

}

}

public string AssemblyCompany

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCompanyAttribute)attributes[0]).Company;

}

}

#endregion

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

{

this.Close();

}

}

}

**// TranslateToHeaderText.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System.Collections.Generic;

using System.Linq;

namespace Lab4.View.TranslateToHeaderText

{

public static class TranslateToHeaderText

{

static readonly Dictionary<string, string> headerTextDict = new Dictionary<string, string>

{

//debtors

{ "Id", "№" },

{ "Category", "Категория" },

{ "Name", "Наименование" },

{ "INN", "ИНН" },

{ "OGRN", "ОГРН" },

{ "Adress", "Адрес" },

{ "Phone", "Телефон" },

{ "Email", "Электронная почта" },

//debts

{ "DebtId", "№ долга" },

{ "DebtorId", "№ должника" },

{ "Sum", "Сумма" },

{ "TakenDate", "Дата взятия" },

{ "LimitDate", "Должен оплатить до" },

{ "Status", "Статус" },

{ "PaidDate", "Дата оплаты" }

};

public static string HeaderText(this string toTranslate)

{

return headerTextDict[toTranslate];

}

public static string ColumnName(this string toTranslate)

{

return headerTextDict.Where(pair => pair.Value == toTranslate)

.Select(pair => pair.Key)

.FirstOrDefault();

}

}

}

**// SearchForm.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

namespace Lab4.View.SearchForm

{

public partial class SearchForm : Form

{

readonly string placeHolder = "Введите запрос и нажмите Enter";

public DataTable DataForAutoComplete { get; set; }

public new event Action<string, string> FormClosed;

public SearchForm(DataTable dataForAutoComplete)

{

InitializeComponent();

DataForAutoComplete = dataForAutoComplete;

SearchByComboBox.SelectedIndex = 0;

}

void SearchTextBox\_Enter(object sender, EventArgs e)

{

SearchTextBoxPlaceHolderHide();

}

void SearchTextBox\_Leave(object sender, EventArgs e)

{

SearchTextBoxPlaceHolderShow();

}

void SearchTextBoxPlaceHolderShow()

{

if (string.IsNullOrWhiteSpace(SearchTextBox.Text))

{

SearchTextBox.Text = placeHolder;

SearchTextBox.ForeColor = Color.Gray;

}

}

void SearchTextBoxPlaceHolderHide()

{

if (SearchTextBox.Text == placeHolder)

{

SearchTextBox.Text = "";

SearchTextBox.ForeColor = Color.Black;

}

}

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

{

if (e.KeyCode == Keys.Enter)

{

this.Close(SearchByComboBox.SelectedItem as string, SearchTextBox.Text);

}

}

private void Close(string searchColumn, string searchValue)

{

this.Close();

this.FormClosed(searchColumn, searchValue);

}

}

}

**// IMessageService.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System.Windows.Forms;

namespace Lab4.View.MessageService

{

public interface IMessageService

{

void ShowMessage(string message);

void ShowExclamation(string exclamation);

void ShowError(string error);

DialogResult ShowOkCancel(string message);

}

}

**// MessageService.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System.Windows.Forms;

namespace Lab4.View.MessageService

{

class MessageService : IMessageService

{

public void ShowError(string error)

{

MessageBox.Show(error, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

public void ShowExclamation(string exclamation)

{

MessageBox.Show(exclamation, "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

public void ShowMessage(string message)

{

MessageBox.Show(message, "Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

public DialogResult ShowOkCancel(string message)

{

return MessageBox.Show(message, "Предупреждение", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

}

}

}

**// FillDebtorForm.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Collections.Generic;

using System.Data;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab4.View.FillingForms

{

public partial class FillDebtorForm : Form, IFillForm

{

public new event Action<List<Control>> FormClosed;

readonly Dictionary<TextBoxBase, Regex> regexDict = new Dictionary<TextBoxBase, Regex>();

readonly Dictionary<TextBoxBase, Label> warningLabels = new Dictionary<TextBoxBase, Label>();

void InitializeDicts()

{

regexDict.Add(DebtorName, new Regex(@"(^.{3,}$)"));

regexDict.Add(Adress, new Regex(@"(^.{5,}$)"));

regexDict.Add(Email, new Regex(@"^([\w\-]+\.)\*[\w\-]+@([\w\-]+\.)+([\w\-]{2,3})|(^$)"));

warningLabels.Add(DebtorName, NameErrorLbl);

warningLabels.Add(INN, INNErrorLbl);

warningLabels.Add(OGRN, OGRNErrorLbl);

warningLabels.Add(Phone, PhoneErrorLbl);

warningLabels.Add(Adress, AdressErrorLbl);

warningLabels.Add(Email, EmailErrorLbl);

}

public FillDebtorForm()

{

InitializeComponent();

InitializeDicts();

this.DebtorName.Name = "Name";

}

public new void Show()

{

Category.SelectedIndex = 0;

InitializeShow("Добавить запись о должнике");

}

public void Show(DataRow dataSource)

{

foreach (Control control in Controls)

try

{

control.Text = dataSource[control.Name].ToString();

}

catch { }

InitializeShow("Редактировать запись о должнике");

}

private void InitializeShow(string formTitle)

{

this.Text = formTitle;

foreach (Control control in this.Controls)

{

if (control is MaskedTextBox maskedTextBox)

{

maskedTextBox.Enter += MoveCaretToStart;

maskedTextBox.KeyDown += EnterPressed;

}

if (control is TextBox textBox)

{

textBox.KeyDown += EnterPressed;

}

if (control is ComboBox comboBox)

{

comboBox.KeyDown += EnterPressed;

}

}

base.Show();

DebtorName.Focus();

}

private void EnterPressed(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

okBut.PerformClick();

}

private void MoveCaretToStart(object sender, EventArgs e)

{

if (sender is MaskedTextBox textBox)

{

MaskFormat oldFormat = textBox.TextMaskFormat;

textBox.TextMaskFormat = MaskFormat.IncludePromptAndLiterals;

string fullText = textBox.Text;

textBox.TextMaskFormat = oldFormat;

int index = fullText.IndexOf(textBox.PromptChar);

if (index > -1)

{

BeginInvoke(new Action(() => textBox.Select(index, 0)));

}

}

}

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

{

bool noErrors = true;

var resultTextBoxes = new List<Control>();

var category = new Control { Name = "Category", Text = Category.SelectedItem.ToString() };

resultTextBoxes.Add(category);

foreach (Control control in this.Controls)

{

if (control is MaskedTextBox maskedTextBox)

{

if (maskedTextBox.MaskCompleted)

{

resultTextBoxes.Add(maskedTextBox);

warningLabels[maskedTextBox].Visible = false;

}

else

{

if (maskedTextBox.Equals(OGRN) && maskedTextBox.Text == " - - - - -")

{

resultTextBoxes.Add(new Control { Name = "OGRN", Text = "" });

warningLabels[maskedTextBox].Visible = false;

continue;

}

warningLabels[maskedTextBox].Visible = true;

noErrors = false;

}

}

if (control is TextBox textBox)

{

if (regexDict[textBox].IsMatch(textBox.Text))

{

resultTextBoxes.Add(new Control { Name = textBox.Name, Text = textBox.Text });

warningLabels[textBox].Visible = false;

}

else

{

warningLabels[textBox].Visible = true;

noErrors = false;

}

}

}

if (noErrors)

this.Close(resultTextBoxes);

}

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

{

this.Close();

}

private void Close(List<Control> resultTextBoxes)

{

this.Close();

this.FormClosed(resultTextBoxes);

}

}

}

**// FillDebtForm.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Collections.Generic;

using System.Data;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab4.View.FillingForms

{

public partial class FillDebtForm : Form, IFillForm

{

public new event Action<List<Control>> FormClosed;

readonly Dictionary<TextBoxBase, Regex> regexDict = new Dictionary<TextBoxBase, Regex>();

readonly Dictionary<TextBoxBase, Label> warningLabels = new Dictionary<TextBoxBase, Label>();

void InitializeDicts()

{

regexDict.Add(Sum, new Regex(@"^\d+[\.|\,]?\d\*$"));

warningLabels.Add(Sum, SumErrorLbl);

warningLabels.Add(TakenDate, TakenDateErrorLbl);

warningLabels.Add(LimitDate, LimitDateErrorLbl);

warningLabels.Add(PaidDate, PaidDateErrorLbl);

}

public FillDebtForm()

{

InitializeComponent();

InitializeDicts();

}

public new void Show()

{

Status.SelectedIndex = 0;

InitializeShow("Добавить запись о долге");

}

public void Show(DataRow dataSource)

{

foreach (Control control in Controls)

try

{

control.Text = dataSource[control.Name].ToString();

}

catch { }

InitializeShow("Редактировать запись о долге");

}

void InitializeShow(string formTitle)

{

this.Text = formTitle;

foreach (Control control in this.Controls)

{

if (control is MaskedTextBox maskedTextBox)

{

maskedTextBox.Enter += MoveCaretToStart;

maskedTextBox.KeyDown += EnterPressed;

}

if (control is TextBox textBox)

{

textBox.KeyDown += EnterPressed;

}

if (control is ComboBox comboBox)

{

comboBox.KeyDown += EnterPressed;

}

}

base.Show();

Sum.Focus();

}

private void EnterPressed(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

okBut.PerformClick();

}

private void MoveCaretToStart(object sender, EventArgs e)

{

if (sender is MaskedTextBox textBox)

{

MaskFormat oldFormat = textBox.TextMaskFormat;

textBox.TextMaskFormat = MaskFormat.IncludePromptAndLiterals;

string fullText = textBox.Text;

textBox.TextMaskFormat = oldFormat;

int index = fullText.IndexOf(textBox.PromptChar);

if (index > -1)

{

BeginInvoke(new Action(() => textBox.Select(index, 0)));

}

}

}

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

{

bool noErrors = true;

var resultTextBoxes = new List<Control>();

var status = new Control { Name = "Status", Text = Status.SelectedItem.ToString() };

resultTextBoxes.Add(status);

foreach (Control control in this.Controls)

{

if (control is MaskedTextBox maskedTextBox)

{

if (maskedTextBox.MaskCompleted)

{

resultTextBoxes.Add(maskedTextBox);

warningLabels[maskedTextBox].Visible = false;

}

else

{

if (maskedTextBox.Equals(PaidDate) && maskedTextBox.Text == " / /")

{

resultTextBoxes.Add(new Control { Name = "PaidDate", Text = "" });

warningLabels[maskedTextBox].Visible = false;

continue;

}

warningLabels[maskedTextBox].Visible = true;

noErrors = false;

}

}

if (control is TextBox textBox)

{

if (regexDict[textBox].IsMatch(textBox.Text))

{

resultTextBoxes.Add(new Control { Name = textBox.Name, Text = textBox.Text });

warningLabels[textBox].Visible = false;

}

else

{

warningLabels[textBox].Visible = true;

noErrors = false;

}

}

}

if (noErrors)

this.Close(resultTextBoxes);

}

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

{

this.Close();

}

private void Close(List<Control> resultTextBoxes)

{

this.Close();

this.FormClosed(resultTextBoxes);

}

}

}

**// IPresenter.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

namespace Lab4.Presenter

{

interface IPresenter

{

void RunApplication();

}

}

**// Presenter.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Lab4.Model;

using Lab4.Repository;

using Lab4.View;

using System;

namespace Lab4.Presenter

{

class Presenter : IPresenter

{

readonly IRepository<Debtor> debtorRepository;

readonly IRepository<Debt> debtRepository;

readonly IView view;

readonly IFileExportService fileExportService;

public void RunApplication() => view.Show();

public Presenter(IRepository<Debtor> debtorRepository, IRepository<Debt> debtRepository,

IView view, IFileExportService fileExportService)

{

this.debtorRepository = debtorRepository;

this.debtRepository = debtRepository;

this.view = view;

this.fileExportService = fileExportService;

this.view.FormLoad += PopulateTable;

this.view.RefreshBut\_Click += PopulateTable;

this.view.SaveBut\_Click += SaveToDB;

this.view.ExportAllDebtorsAndDebtsBut\_Click += ExportAllDebtorsAndDebts;

this.view.ExportCertainDebtorBut\_Click += ExportCertainDebtors;

}

private void ExportCertainDebtors(object sender, EventArgs e)

{

fileExportService.Table = view.Table;

fileExportService.FilePath = view.ExportPath;

fileExportService.SaveDebtorDebts();

view.MessageService.ShowMessage("Успешно сохранено!");

}

private void ExportAllDebtorsAndDebts(object sender, EventArgs e)

{

fileExportService.Table = view.Table;

fileExportService.FilePath = view.ExportPath;

fileExportService.SaveAllDebtorsAndDebts();

view.MessageService.ShowMessage("Успешно сохранено!");

}

private void SaveToDB(object sender, EventArgs e)

{

var updatedDebtors = debtorRepository.UpdateDB(view.TableDataSet.Tables[0]);

var updatedDebts = debtRepository.UpdateDB(view.TableDataSet.Tables[0]);

view.TableDataSet = debtorRepository.ConnectTwoTables<Debtor, Debt>(updatedDebtors, updatedDebts);

view.MessageService.ShowMessage("Успешно сохранено!");

}

void PopulateTable(object sender, EventArgs e)

{

var debtors = debtorRepository.GetAllRecords();

var debts = debtRepository.GetAllRecords();

view.TableDataSet = debtorRepository.ConnectTwoTables<Debtor, Debt>(debtors, debts);

}

}

}

**// ExitAboutButs.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

namespace Lab4.View

{

partial class View

{

void ExitBut\_Click(object sender, EventArgs e)

{

this.Close();

}

void AboutProgBut\_Click(object sender, EventArgs e)

{

new AboutBox().Show();

}

}

}

**// TableChangingMethods.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Lab4.View.DataTableExtension;

using Lab4.View.FillingForms;

using Lab4.View.TranslateToHeaderText;

using System;

using System.Collections.Generic;

using System.Data;

using System.Windows.Forms;

namespace Lab4.View

{

partial class View

{

private void CommitSave(EventHandler eventHandler, object sender, EventArgs e)

{

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

{

this.ExportPath = saveFileDialog.FileName;

eventHandler?.Invoke(sender, e);

}

}

private void CancelSearch(object sender, EventArgs e)

{

Table.Reset();

}

private void MakeSearch(object sender, EventArgs e)

{

var searchForm = new SearchForm.SearchForm(Table.ParentDataSource);

searchForm.FormClosed += Search;

searchForm.Show();

}

private void Search(string searchColumn, string searchValue)

{

var translatedSearchColumn = searchColumn.ColumnName();

Table.FilterBy(translatedSearchColumn, searchValue);

}

private void DeleteRows(object sender, EventArgs e)

{

if (MessageService.ShowOkCancel("Вы уверены, что хотите удалить выбранную(ые) запись(и)?") == DialogResult.OK)

{

if (Table.ChildDataGridView.Visible)

for (int i = 0; i < Table.ChildDataGridView.SelectedRows.Count; ++i)

{

var rowToDelete = Table.ChildDataGridView.SelectedRows[i];

var id = rowToDelete.Cells[0].Value;

Table.ChildDataSource.Rows.Find(id).Delete();

}

else

{

for (int i = 0; i < Table.SelectedRows.Count; ++i)

{

var rowToDelete = Table.SelectedRows[i];

var id = rowToDelete.Cells[1].Value;

Table.ParentDataSource.Rows.Find(id).Delete();

}

}

}

MessageService.ShowMessage("Запись(и) успешно удалена(ы)");

}

private void AddDebtorOrDebt(object sender, EventArgs e)

{

if (Table.ChildDataGridView.Visible)

{

fillForm = new FillDebtForm();

fillForm.FormClosed += (record) => AddNewDebt(record, Table.ChildDataSource);

debtorId = Table.SelectedRows[0].Cells[1].Value;

}

else

{

fillForm = new FillDebtorForm();

fillForm.FormClosed += (record) => AddNewDebtor(record, Table.ParentDataSource);

}

fillForm.Show();

}

private object debtorId;

private void AddNewDebt(List<Control> recordToInsert, DataTable dataTable)

{

var record = new Dictionary<string, string>

{

{ "DebtorId", debtorId.ToString() }

};

foreach (var element in recordToInsert)

record.Add(element.Name, element.Text);

dataTable.AddRow(record);

MessageService.ShowMessage("Запись успешно добавлена!");

}

private void AddNewDebtor(List<Control> recordToInsert, DataTable dataTable)

{

var record = new Dictionary<string, string>();

foreach (var element in recordToInsert)

record.Add(element.Name, element.Text);

dataTable.AddRow(record);

MessageService.ShowMessage("Запись успешно добавлена!");

}

private object editingId;

private void EditDebtorOrDebt(object sender, EventArgs e)

{

if (Table.ChildDataGridView.Visible)

{

if (Table.ChildDataGridView.SelectedRows.Count > 1)

{

MessageService.ShowError("Нельзя редактировать более одной строки! Выберите одну!");

return;

}

fillForm = new FillDebtForm();

var selectedId = Table.ChildDataGridView.SelectedRows[0].Cells[0].Value;

editingId = selectedId;

fillForm.Show(Table.ChildDataSource.Rows.Find(selectedId));

fillForm.FormClosed += (record) => EditRecord(record, Table.ChildDataSource);

}

else

{

if (Table.SelectedRows.Count > 1)

{

MessageService.ShowError("Нельзя редактировать более одной строки! Выберите одну!");

return;

}

fillForm = new FillDebtorForm();

var selectedId = Table.SelectedRows[0].Cells[1].Value;

editingId = selectedId;

fillForm.Show(Table.ParentDataSource.Rows.Find(selectedId));

fillForm.FormClosed += (record) => EditRecord(record, Table.ParentDataSource);

}

}

private void EditRecord(List<Control> recordToInsert, DataTable dataTable)

{

for (int i = 0; i < recordToInsert.Count; ++i)

{

var editingRow = dataTable.Rows.Find(editingId);

editingRow[recordToInsert[i].Name] = recordToInsert[i].Text;

}

MessageService.ShowMessage("Запись успешно изменена!");

}

}

}

**// DataTableExtension.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System.Collections.Generic;

using System.Data;

namespace Lab4.View.DataTableExtension

{

public static class AddNewRowExtension

{

private static long CalculateNewId(DataTable dataTable)

{

var ids = new List<long>();

foreach (DataRow row in dataTable.Rows)

{

if (row.RowState != DataRowState.Deleted)

ids.Add(long.Parse(row[dataTable.PrimaryKey[0]]?.ToString()));

}

long id = 1;

while (true)

{

if (!ids.Contains(id))

return id;

++id;

}

}

public static void AddRow(this DataTable dataTable, IDictionary<string, string> rowToInsert)

{

var idColumn = dataTable.PrimaryKey[0];

var newId = CalculateNewId(dataTable);

var newRow = dataTable.NewRow();

newRow[idColumn] = newId;

foreach (var element in rowToInsert)

{

newRow[element.Key] = element.Value;

}

dataTable.Rows.Add(newRow);

}

}

}

**// IFileExportService.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

namespace Lab4.Presenter

{

public interface IFileExportService

{

string FilePath { get; set; }

View.NestedTable.NestedTable Table { get; set; }

void SaveAllDebtorsAndDebts();

void SaveDebtorDebts();

}

}

**// ExcelExportPrivateMethods.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using System;

using System.Data;

namespace Lab4.Presenter

{

partial class ExcelExport

{

void ChangeHeaders(DataSet debtorsAndDebts, Func<string, string> headerTextOrColumnName)

{

DataTable debtors = debtorsAndDebts.Tables[0];

foreach (DataColumn column in debtors.Columns)

column.ColumnName = headerTextOrColumnName(column.ColumnName);

DataTable debts = debtorsAndDebts.Tables[1];

foreach (DataColumn column in debts.Columns)

column.ColumnName = headerTextOrColumnName(column.ColumnName);

}

}

}

**// ExcelExport.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using ClosedXML.Excel;

using Lab4.View.TranslateToHeaderText;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace Lab4.Presenter

{

public partial class ExcelExport : IFileExportService

{

public string FilePath { get; set; }

public View.NestedTable.NestedTable Table { get; set; }

public void SaveAllDebtorsAndDebts()

{

var debtors = new DataTable();

var debts = new DataTable();

var toMerge = new DataView(Table.ParentDataSource);

debtors.Merge(toMerge.ToTable());

toMerge = new DataView(Table.ChildDataSource);

debts.Merge(toMerge.ToTable());

var debtorsAndDebts = new DataSet();

debtorsAndDebts.Tables.AddRange(new[] { debtors, debts });

ChangeHeaders(debtorsAndDebts, TranslateToHeaderText.HeaderText);

debtorsAndDebts.Tables[0].TableName = "Должники";

debtorsAndDebts.Tables[1].TableName = "Долги";

using (var wb = new XLWorkbook())

{

wb.AddWorksheet(debtorsAndDebts);

wb.Worksheets.ToList().ForEach(ws => { ws.Rows().AdjustToContents(); ws.Columns().AdjustToContents(); });

wb.SaveAs(FilePath);

}

ChangeHeaders(debtorsAndDebts, TranslateToHeaderText.ColumnName);

}

public void SaveDebtorDebts()

{

var debtors = new DataTable();

var debts = new DataTable();

foreach (DataGridViewRow row in Table.SelectedRows)

{

var id = row.Cells[1].Value;

var toMerge = new DataView(Table.ParentDataSource) { RowFilter = "Id = '" + id + "'" };

debtors.Merge(toMerge.ToTable());

toMerge = new DataView(Table.ChildDataSource) { RowFilter = "DebtorId = '" + id + "'" };

debts.Merge(toMerge.ToTable());

}

var debtorsAndDebts = new DataSet();

debtorsAndDebts.Tables.AddRange(new[] { debtors, debts });

ChangeHeaders(debtorsAndDebts, TranslateToHeaderText.HeaderText);

debtorsAndDebts.Tables[0].TableName = "Должники";

debtorsAndDebts.Tables[1].TableName = "Долги";

using (var wb = new XLWorkbook())

{

wb.AddWorksheet(debtorsAndDebts);

wb.Worksheets.ToList().ForEach(ws => { ws.Rows().AdjustToContents(); ws.Columns().AdjustToContents(); });

wb.SaveAs(FilePath);

}

ChangeHeaders(debtorsAndDebts, TranslateToHeaderText.ColumnName);

}

}

}

**// IModel.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

namespace Lab4.Model

{

public interface IModel

{

string PrimaryKey { get; }

string TableName { get; }

string RelationKey { get; }

}

}

**// Debt.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

namespace Lab4.Model

{

public class Debt : IModel

{

public string PrimaryKey { get => "DebtId"; }

public string TableName { get => "Debts"; }

public string RelationKey { get => "DebtorId"; }

}

}

**// Debtor.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

namespace Lab4.Model

{

public class Debtor : IModel

{

public string PrimaryKey { get => "Id"; }

public string TableName { get => "Debtors"; }

public string RelationKey { get => "Id"; }

}

}

**// IRepository.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Lab4.Model;

using System.Data;

namespace Lab4.Repository

{

public interface IRepository<T> where T : IModel, new()

{

DataTable GetAllRecords();

DataTable UpdateDB(DataTable toUpdate);

DataSet ConnectTwoTables<ParentModel, ChildModel>(DataTable parentTable, DataTable childTable)

where ParentModel : IModel, new() where ChildModel : IModel, new();

}

}

**// Repository.cs**

**// Лабораторная работа №4.**

**// Студент группы 485, Зобнин Илья Михайлович. 2020 год**

using Lab4.Model;

using System.Configuration;

using System.Data;

using System.Data.SQLite;

using System.Linq;

namespace Lab4.Repository

{

public class Repository<T> : IRepository<T> where T : IModel, new()

{

T model;

public Repository(T model)

{

this.model = model;

}

public DataTable GetAllRecords()

{

var table = new DataTable();

string selectionStatement = "SELECT \* FROM " + model.TableName;

string connectionString = GetConnectionString();

using (var connection = new SQLiteConnection(connectionString))

{

connection.Open();

var dataAdapter = new SQLiteDataAdapter(selectionStatement, connection);

dataAdapter.Fill(table);

table.PrimaryKey = table.PrimaryKey.Append(table.Columns[model.PrimaryKey]).ToArray();

}

return table;

}

public DataTable UpdateDB(DataTable toUpdate)

{

string connectionString = GetConnectionString();

string selectionStatement = "SELECT \* FROM " + model.TableName;

using (var connection = new SQLiteConnection(connectionString))

{

connection.Open();

var dataAdapter = new SQLiteDataAdapter(selectionStatement, connection);

var commandBuilderForDebtors = new SQLiteCommandBuilder(dataAdapter);

dataAdapter.Update(toUpdate);

toUpdate.AcceptChanges();

}

return GetAllRecords();

}

public DataSet ConnectTwoTables<ParentModel, ChildModel>(DataTable parentTable, DataTable childTable)

where ParentModel : IModel, new() where ChildModel : IModel, new()

{

ParentModel parentModel = new ParentModel();

ChildModel childModel = new ChildModel();

var dataSet = new DataSet();

dataSet.Tables.AddRange(new[] { parentTable, childTable });

dataSet.Relations.Add

(parentTable.Columns[parentModel.RelationKey], childTable.Columns[childModel.RelationKey]);

return dataSet;

}

private static string GetConnectionString(string id = "Default")

{

return ConfigurationManager.ConnectionStrings[id].ConnectionString;

}

}

}

**[--- Конец программы ---]**