ВСТУП

Покупки вже давно стали невід'ємною частиною нашого життя.   
З розвитком комп'ютерних технологій був знайдений вихід – створити інтелектуальну систему, яка виконувала всі необхідні завдання, систему, яка була проста в розумінні для споживачів. В даній курсовій роботі необхідно розробити програму «Довідник покупця». Метою даної курсової роботи є розробка емулятора, який буде візуально схожий на справжню базу данних, забезпечити взаємодія програми з користувачем(клієнтом). За основу взято об'єктно-орієнтований підхід.   
 Завдання виконання роботи:

1. дослідження предметної області з метою виявлення основних принципів даної сфери;
2. проектування ієрархії класів, інтерфейсів, взаємодії компонентів на основі виділених принципів і даних коштів;
3. використання вбудовані елементи середовища для структуризації класів та оптимізації коду;
4. застосування принципу інкапсуляції до класів.

Об'єктно-орієнтований підхід вимагає глибокого розуміння основних принципів, або, інакше, концепцій, на яких він базується. У даному підході основними концепціями є поняття об'єктів і класів.   
 Об'єктно-орієнтоване програмування в даний час є абсолютним лідером в області прикладного програмування. Використання цього підходу надає програмісту широкі можливості в функціональності і сопровождаемости проекту.  
 В якості основного інструменту розробки застосовується Microsoft Visual Studio 2013 Proffesional. Мова програмування C#.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Основні вимоги та функції програми

Метою курсового проекту є написання програми «Довідник покупця», а також написання документації супроводжує його.

Програма повинна мати функції, які надають користувачу можливості для перегляду, редагування, збереження і завантаження даних.

Стійкість програми повинна бути максимальною: програма не повинна втрачати працездатність ні за яких, навіть некоректні дії користувача. Всякі дії, які загрожують втратою інформації, повинні бути підтверджені користувачем.

Інтерфейс користувача повинен бути зрозумілим і приємним зовні, а так само надавати зручні можливості для роботи з даними.

На будь-якому етапі натискання будь-якої клавіші повинно ігноруватися або викликати передбачені дії, описані в засобах допомоги.

Функції програми:

1. надання даних для перегляду, а саме: для магазинів – назва магазинів, список телефонів, адрес, час роботи, телефони;
2. додавання в програму нових і зміна існуючих торгових точок;
3. видалення існуючих торгових точок. Слід зазначити, що видалення, як і зміна виконується щодо виділеного в цих списках елемента. При виборі функції видалення користувачеві надається можливість скасувати своє рішення або підтвердити його;
4. завантаження списку торгових точок з диску: передбачає очищення поточного списку і завантаження останнього збереженого списку відповідних торгових точок;
5. функція пошуку дозволяє виявити магазини з заданими критері ями.

1.2 Інтерфейс користувача

Дана програма має 4 вікна, кожне з яких надає користувачеві можливість виконання будь-якої з функцій програми.

При запуску програми відкривається основне вікно, в якому користувач має можливість переглядати інформацію торгові точки, а також виконувати функції видалення та пошуку видимих торгови точок. В основному вікні розташовані елементи для перегляду даних про торгові точки в яких відображаються списки торгови точок, а також інформація про магазин.

Так само в основному вікні реалізована можливість пошуку, що дозволяє знайти торгові точки.

Розглянемо меню в основному вікні формі.

Підменю «Editor» містить наступні пункти:

1. пункт «New firm» дозволяє користувачеві вносити в список нові магазини. При натискання на пункт «New firm» буде відкрито діалогове вікно, в яке необхідно заповнити інформацію про торгову точку;
2. пункт «Edit firm» дозволяє користувачеві вносити зміни в список магазинів. При натискання пункт «Edit firm» буде відкрито діалогове вікно, в полі якого буде записана вже наявна інформація про магазин, і яке можна буде зберегти після редагування;
3. пункт «Delete firm» дозволяє користувачеві повністю видалити магазин. Інформація про магазин так само буде повністю видалена. При виклику цього пункту користувачеві буде запропоновано підтвердити або скасувати своє рішення;
4. пункт «Directories» дозволяє користувачеві вносити зміни в список типів власності і спеціалізацій. При натискання пункт «Directories» буде відкрито діалогове вікно, в полі якого буде записана вже наявна інформація про типи власності і спеціалізації, і яке можна буде зберегти після редагування.

# 2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Об’єктна модель програми

Основним конструктивним елементом у мовах ООП є модуль (module), який представляє собою логічно пов'язану сукупність класів і об'єктів, а не підпрограм, як у попередніх мовами.

У графічному інтерфейсі цієї програми містяться декілька форм заповнення і редагування даних, отже в проекті присутні спеціальні допоміжні функції, які зчитують дані, заповнюють колекції зберігають дані. Зчитування даних відбувається при завантаженні вікон.

У даному проекті був реалізований зручний інтерфейс, який схожий на більшість Windows – програм. Проект можна представити у вигляді модулів, взаємодіючих один з одним (рис. 2.1).

Модуль збереження даних

Модуль редагування даних

Модуль зчитування даних

Модуль вводу даних

Модуль редагування данных

Модуль збереження даных

Модуль зчитувания даных

Модуль перегляду даних

Рисунок 2.1 – «Структура програми»

Модуль вводу даних, являє собою зручний інтерфейс для введення необхідної інформації.

Модуль редагування даних, що дозволяє швидко і зручно змінити вже існуючі дані.

Модуль зчитування даних, робить можливим відкрити текстовий файл і заповнити довідник.

Модуль перегляду даних, що надає користувачеві можливість зручно переглядати вміст довідника.

Модуль збереження даних, створює текстовий документ, що містить інформацію про магазини.

В даному курсовому проекті було реалізовано систему класів (рис 2.2), які тісно взаємодію між собою.

Program

static Класс

Specialization

Класс

Firm

Класс

TypeOfOwnerships

Класс

DataManager

static Класс

FileHelper

static Класс

Рисунок 2.2 – «Объектная спецификация программы»

Клас Specialization (рис 2.3) містить два відкритих поля – власний номер і назва, методи повертають ім'я екземпляра і те ж саме ім'я, але для збереження у файл, а ще 3 конструктора, що дозволяють створити екземпляр з певним ім'ям і id або з файлової інтерпретації раніше записаної туди спеціалізації.

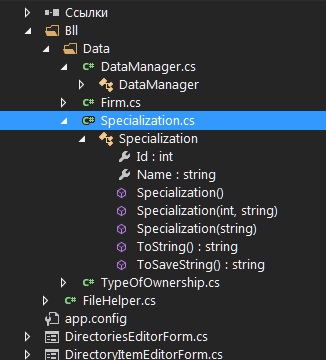


Рисунок 2.3 – «Specialization»

Клас DataManager (рис. 2.4) – містить колекції фірм спеціалізацій і типів власності, а також методи завантаження, збереження, додавання, редагування, видалення фірм, спеціалізацій, типів власності.

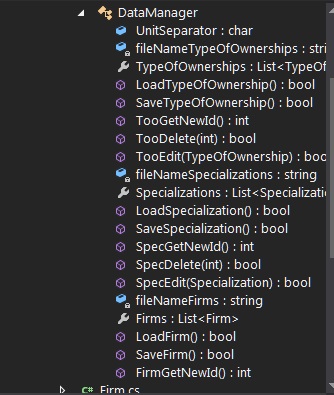


Рисунок 2.4 – «Класс DataManager»

Клас FileHelper (рис. 2.6) – Перевіряє на існування і отримує шлях до директорії з файлами і створює їх, якщо такі відсутні, безпосередньо завантажує і записує дані з файлів.

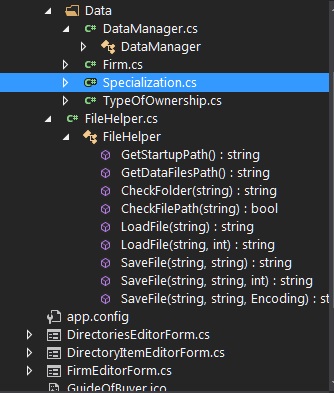


Рисунок 2.6 – «Клас FileHelper»

2.2 Реалізация функцій програми

Як згадувалося раніше, дана програма складається з декількох компонентів, які тісно взаємопов'язані один з одним. В цьому розділі будуть розглянуті більшість з цих функцій. Повний набір методів можна побачити в коді програми (додаток А).

Клас Specialization (рис 2.3)

Метод LoadSpecialization() (рис. 2.5) – спочатку створює шлях до потрібного файлу, потім робить колекцію спеціалізацій, потім перевіряє наявність потрібної директорії, використовує метод класу FilеHelper, щоб зчитати дані з файлу, потім розбиває рядок на масив рядків і додає назви спеціалізацій в колекцію

.

private const string fileNameSpecializations = "Specializations.txt";

public static List<Specialization> Specializations { get; set; }

public static bool LoadSpecialization()

{

var path = FileHelper.GetDataFilesPath() + fileNameSpecializations;

Specializations = new List<Specialization>();

if (!File.Exists(path))

{

return false;

}

var str = FileHelper.LoadFile(path, 1251);

var lines = str.Replace("\r\n", "\r").Split('\r');

foreach (var line in lines)

{

if (string.IsNullOrEmpty(line))

{

continue;

}

var obj = new Specialization(line);

Specializations.Add(obj);

}

return true;

}

Рисунок 2.5 – «Метод завантаження спеціалізації»

Клас FileHelper (рис. 2.6)

Метод LoadFile() (рис. 2.6) – Метод зчитує файл в масив байтів, перетворює масив байтів в текст з урахуванням кодової сторінки і повертає рядок.

public static string LoadFile(string fileName, int codePage)

{

try

{

// Считываем файл в массив байтов...

using (var fs = new FileStream(fileName, FileMode.Open, FileAccess.Read))

{

var bt = new byte[(int)fs.Length];

fs.Read(bt, 0, (int)fs.Length);

// Преобразуем массив байтов в текст с учетом

// кодовой страницы по умолчанию.

//CodePageEncoding cpe = new CodePageEncoding(0);

//tbTextForSearch.Text = cpe.GetString(bt);

return codePage <= 0 ? Encoding.UTF8.GetString(bt) : Encoding.GetEncoding(codePage).GetString(bt);

}

}

catch

{

return "";

}

}

Рисунок 2.6 – «Метод LoadFile()»

Метод SaveFile() (рис. 2.7) – приймає ім'я файлу з шляхом до нього, рядок для запису і кодировочную таблицю, перевіряє на наявність файлу, у вказаній директорії і якщо він там є то видаляє його і створює новий, записуючи в нього потрібну інформацію.

/// <summary>

/// Сохраняет текстовый файл /\*Взято с stackoverflow.com\*/

/// </summary>

/// <param name="fileName">Имя файла с путём</param>

/// <param name="text">Содержимое файла</param>

/// <param name="encoding">Кодировочная таблица для сохранения</param>

/// <returns></returns>

public static string SaveFile(string fileName, string text, Encoding encoding)

{

try

{

if (File.Exists(fileName))

{

//Если файл существует, удаляем его

File.Delete(fileName);

}

CheckFilePath(fileName);

using (var fs = new FileStream(fileName, FileMode.CreateNew))

{

using (var sw = new StreamWriter(fs, encoding))

{

sw.Write(text);

sw.Close();

}

}

}

catch (Exception ex)

{

return ex.Message;

}

return "";

}

Рисунок 2.7 – «Метод SaveFile()»

Форма MainForm

Метод InitData() (рис. 2.8) – послідовно завантажує на головну форму спеціалізації, типи власності і магазини.

private void InitData()

{

flagStartApplication = true;

cbSpect.Items.Clear();

cbSpect.Items.Add(new Specialization(0, "None")); //Служебная запись показывающая что фильтр не задан

if (DataManager.Specializations != null)

{

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

{

cbSpect.Items.Add(DataManager.Specializations[i]);

}

}

cbSpect.SelectedIndex = 0;

cbToo.Items.Clear();

cbToo.Items.Add(new TypeOfOwnership(0, "None")); //Служебная запись показывающая что фильтр не задан

if (DataManager.TypeOfOwnerships != null)

{

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

{

cbToo.Items.Add(DataManager.TypeOfOwnerships[i]);

}

}

cbToo.SelectedIndex = 0;

lvFirms.Items.Clear();

foreach (Firm data in DataManager.Firms)

{

var lvi = CreateListViewItem(data);

lvFirms.Items.Add(lvi);

}

flagStartApplication = false;

}

Рисунок 2.8 – «Метод InitData()»

Метод EditFirm() (рис. 2.9) – перевіряє наявність елементів в списку магазинів, потім привласнює змінної data обраний магазин, якщо він не обраний, то присвоює перший у списку, надсилає запит формі редагування і отримує нові дані з неї потім оновлює цей магазин у файлі, а потім і на головній формі.

private void EditFirm()

{

if (lvFirms.Items == null || lvFirms.Items.Count == 0)

{

return;

}

Firm data = null;

if (lvFirms.SelectedItems == null || lvFirms.SelectedItems.Count == 0)

{

lvFirms.Items[0].Selected = true;

data = (Firm)lvFirms.Items[0].Tag;

}

else

{

data = (Firm)lvFirms.SelectedItems[0].Tag;

}

var obj = FirmEditorForm.EditFirm(data);

if (obj != null)

{

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

{

if (DataManager.Firms[i].Id == obj.Id)

{

DataManager.Firms[i] = obj;

break;

}

}

DataManager.SaveFirm();

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

{

if (((Firm) lvFirms.Items[i].Tag).Id == obj.Id)

{

lvFirms.Items[i] = CreateListViewItem(obj);

break;

}

}

}

}

Рисунок 2.9 – «Метод EditFirm()»

Метод DoSearch() (рис. 2.10) – перевіряє на встановлені фільтри і записує їх у змінні, а також пошуковий запит, потім очищає поля на головній формі, потім проходиться по колекції завантажених магазинів і перевіряє їх на відповідність фільтрів і наявність пошукового запиту в будь-якому полі даного екземпляра класу Firm, якщо примірник відповідає фільтрам і містить в своїх полях рядок пошуку то він завантажується на головну форму.

private void DoSearch()

{

if (flagStartApplication || cbSpect.SelectedItem == null || cbToo.SelectedItem == null)

{

return;

}

var specId = ((Specialization) cbSpect.SelectedItem).Id;

var tooId = ((TypeOfOwnership)cbToo.SelectedItem).Id;

var searchTxt = tbSearch.Text.ToUpper();

lvFirms.Items.Clear();

tbAddress.Text = string.Empty;

tbPhone.Text = string.Empty;

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

{

var data = DataManager.Firms[i];

if (specId > 0 && specId != data.SpecId)

{

continue;

}

if (tooId > 0 && tooId != data.TooId)

{

continue;

}

if (!string.IsNullOrEmpty(searchTxt))

{

var flag = (!string.IsNullOrEmpty(data.Name) && data.Name. ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1) ||

(!string.IsNullOrEmpty(data.Address) && data.Address. ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1) ||

(!string.IsNullOrEmpty(data.Phones) && data.Phones. ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1) ||

(!string.IsNullOrEmpty(data.TimeWork) && data.TimeWork.ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1);

if (!flag)

{

continue;

}

}

lvFirms.Items.Add(CreateListViewItem(data));

}

}

Рисунок 2.10 – «Метод DoSearch()»

3 КЕРІВНИЦТВО КОРИСТУВАЧА

Для установки програми досить розпакувати архів в потрібну папку. Далі слід запустити додаток з отриманої папки.

При запуску ви опиняєтеся на основній формі (рис. 3.1). ви побачите кнопку Editor, яка відкриє підменю редагування фірми в цілому. Кнопка Directories відкриє форму редагування спеціалізацій і форм власності.

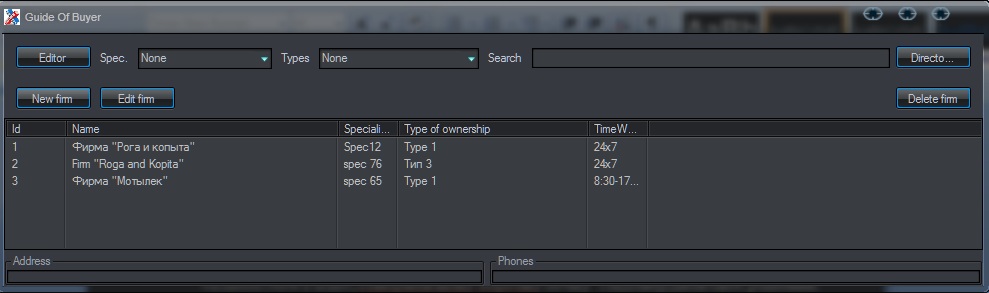


Рисунок 3.1 – «Головна форма»

Для створення нової торгової точки (рис. 3.2) необхідно вибрати пункт «Editor/New Firm», створити нову торгову точку і підтвердити своє рішення.

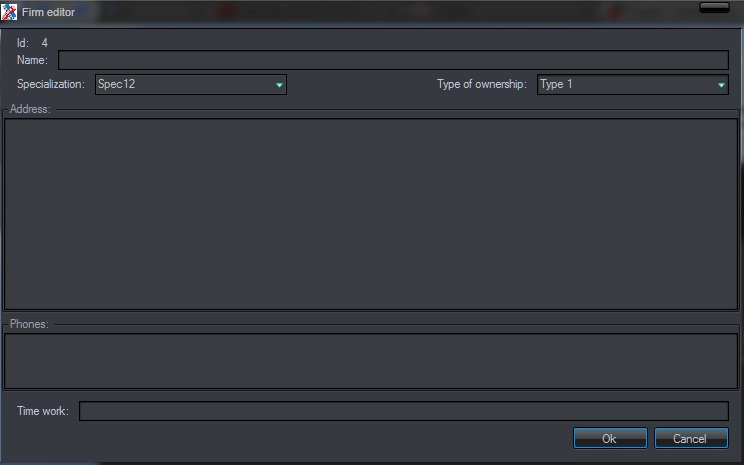


Рисунок 3.2 – «Форма створення нового магазину»

Для видалення магазину (рис. 3.1) натисніть кнопку Delete Firm

При виконанні дій, що загрожують втрати інформації буде відкрито попередження (рис. 3.3). У користувача є можливість скасувати свою дію натиснувши на кнопку «Нет».

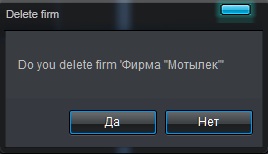


Рисунок 3.3 – «Видалення магазину»

Для додавання або редагування (рис. 3.4) спеціалізації чи форми власності натисніть кнопку «Directories», кнопка «Save» зберігає введені вами дані в файл, а кнопка «Reload» перезавантажує дані з файлу.

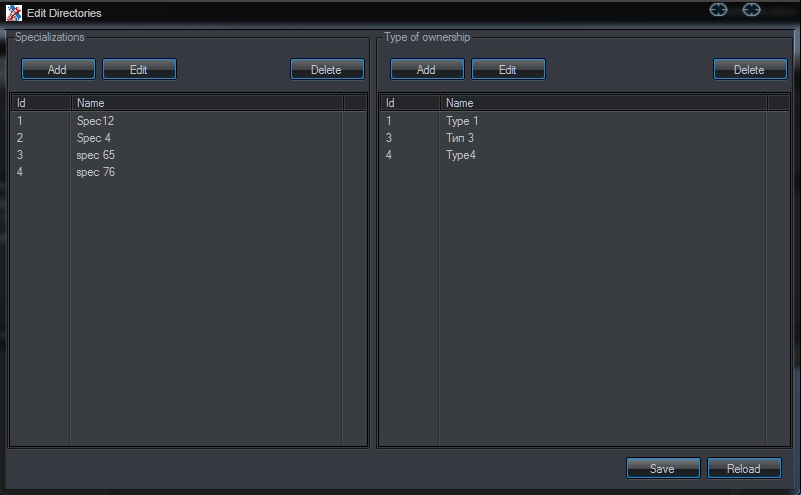


Рисунок 3.4 – «Форма додавання або редагування спеціалізації чи форми власності»

Для додавання (рис. 3.5) спеціалізації чи форми власності натисніть кнопку «Add» » (див. рис.3.4).

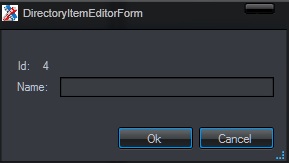


Рисунок 3.5 – «Форма додавання спеціалізації чи форми власності»

Для видалення (рис. 3.6) спеціалізації чи форми власності натисніть кнопку «Delete» (див. рис.3.4).

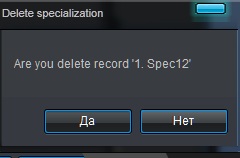


Рисунок 3.6 – «Форма видалення спеціалізації»

Для редагування (рис. 3.7) спеціалізації чи форми власності натисніть кнопку «Edit» (див. рис.3.4).

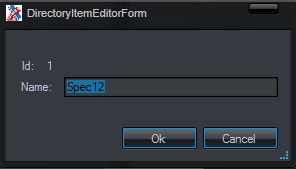


Рисунок 3.7 – «Форма редагування спеціалізації чи форми власності»

ВИСНОВКИ

В результаті виконання поставленої задачі був розроблений програмний продукт, що відповідає всім вимогам і цілям, поставленим при його плануванні. Був розроблений зручний інтерфейс. Реалізована функція пошуку торгових точок. Реалізована функція збереження файлу, що зберігає файл цілком в читабельному вигляді. Під час розробки програми велика увага приділялася дотриманню парадигми об'єктно-орієнтованого програмування. Значну увагу було приділено оптимізації і тестування програми.

Об'єктно-орієнтований підхід дає можливість на ранніх етапах розробки врахувати всі особливості створюються програм, необхідний набір функцій, структуру баз даних, що надалі максимально виключає необхідність переробки вже написаних компонентів програми.

Програма реалізована повноцінно зі зручним і зрозумілим інтерфейсом. Завдяки зрозумілою архітектурі і нескладному управління працювати з базою даних може і непідготовлений користувач.

Даний програмний продукт значно полегшує життя простим покупцям так само програма буде мати величезне практичне застосування. А вбудовані функції, що запобігають втраті даних, дозволить користувачеві не турбується про те що його праця може безслідно зникнути.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондарев, В.М. Объектно-ориентированное программирование на С# [Текст] : учеб. пособ. /В.М. Бондарев. – Х. : Компания СМИТ, 2009. – 224 с. - ISBN 978-966-2028-34-8.
2. ДСТУ 3008-95 "Документація. Звіти в сфері науки і техніки. Структура і правила оформлення" – Державний стандарт України.
3. ДСТУ ГОСТ 7.1:2006 "Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання".

Додаток А – Код програми

Клас Specialization

public class Specialization

{

public int Id { get; set; }

public string Name { get; set; }

public Specialization()

{

}

public Specialization(int id, string name)

{

Id = id;

Name = name;

}

public Specialization(string dataString)

{

if (dataString.IndexOf('|') == -1)

{

throw new Exception("Bad input data. Not found char '|'");

}

var mas = dataString.Split('|');

Id = Convert.ToInt32(mas[0]);

Name = mas[1];

}

public override string ToString()

{

return Name;

}

public string ToSaveString()

{

return string.Format("{0}|{1}", Id, Name);

}

}

Клас DataManager #region Specializations

private const string fileNameSpecializations = "Specializations.txt";

public static List<Specialization> Specializations { get; set; }

public static bool LoadSpecialization()

{

var path = FileHelper.GetDataFilesPath() + fileNameSpecializations;

Specializations = new List<Specialization>();

if (!File.Exists(path))

{

return false;

}

var str = FileHelper.LoadFile(path, 1251);

var lines = str.Replace("\r\n", "\r").Split('\r');

foreach (var line in lines)

{

if (string.IsNullOrEmpty(line))

{

continue;

}

var obj = new Specialization(line);

Specializations.Add(obj);

}

return true;

}

public static bool SaveSpecialization()

{

var path = FileHelper.GetDataFilesPath() + fileNameSpecializations;

var sb = new StringBuilder();

foreach (var specialization in Specializations)

{

sb.AppendLine(specialization.ToSaveString());

}

return string.IsNullOrEmpty(FileHelper.SaveFile(path, sb.ToString()));

}

public static int SpecGetNewId()

{

if (Specializations == null || Specializations.Count == 0)

{

return 1;

}

return Specializations.Select(specialization => specialization.Id).Concat(new[] { 0 }).Max() + 1;

}

public static bool SpecDelete(int id)

{

if (Specializations == null || Specializations.Count == 0)

{

return false;

}

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

{

if (Specializations[i].Id == id)

{

Specializations.RemoveAt(i);

return true;

}

}

return false;

}

public static bool SpecEdit(Specialization data)

{

if (Specializations == null)

{

Specializations = new List<Specialization>();

}

if (Specializations.Count == 0)

{

Specializations.Add(data);

return true;

}

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

{

if (Specializations[i].Id == data.Id)

{

Specializations[i] = data;

return true;

}

}

Specializations.Add(data);

return false;

}

Клас FileHelper

public static class FileHelper

{

#region GetStartupPath

/// <summary>

/// Return startup path application

/// </summary>

/// <returns></returns>

public static string GetStartupPath()

{

//Системная переменная возвращающая путь запуска исполняемого файла

string s = System.AppDomain.CurrentDomain.BaseDirectory;

#region "Выходим" на уровен корня проекта, что бы были одинаковые данные в debug и release режимах

int i;

i = s.IndexOf(@"\Debug", 1, s.Length - 1);

if (i > 0)

{

s = s.Substring(0, i);

i = s.IndexOf(@"\bin", 1, s.Length - 1);

if (i > 0)

{

s = s.Substring(0, i);

}

i = s.IndexOf(@"\obj", 1, s.Length - 1);

if (i > 0)

{

s = s.Substring(0, i);

}

}

#endregion

return s + @"\";

}

#endregion

#region GetDataFilesPath

/// <summary>

/// Return int files path

/// </summary>

/// <returns></returns>

public static string GetDataFilesPath()

{

string res = GetStartupPath() + @"Data\";

CheckFolder(res);

return res;

}

#endregion

#region CheckFolder

/// <summary>

/// Проверяет доступность папки, в случае отсутсвия папки создает папку

/// </summary>

/// <param name="path">Путь к папке</param>

/// <returns>Пустая строка или сообщение об ошибке</returns>

public static string CheckFolder(string path)

{

try

{

if (!Directory.Exists(path))

{

//Если папки не существует, создаем её

Directory.CreateDirectory(path);

}

}

catch (Exception e)

{

return e.Message;

}

return "";

}

#endregion

#region CheckFilePath

/// <summary>

/// Проверяет доступность папки куда нужно положить файл, в случае отсутсвия папки создает полный путь

/// </summary>

/// <param name="fileName">Полный путь к файлу</param>

/// <returns>Успешность операции</returns>

public static bool CheckFilePath(string fileName)

{

//Получаем имя папки отбрасывая имя файла

var path = Path.GetDirectoryName(fileName);

return string.IsNullOrEmpty(CheckFolder(path));

}

#endregion

#region LoadFile

/// <summary>

/// Загружает содержимое файла в текстовое поле

/// </summary>

/// <param name="fileName">Имя файла с путём</param>

/// <returns>Содержимое файла</returns>

public static string LoadFile(string fileName)

{

try

{

// Считываем файл в массив байтов...

using (var fs = new FileStream(fileName, FileMode.Open, FileAccess.Read))

{

var bt = new byte[(int)fs.Length];

fs.Read(bt, 0, (int)fs.Length);

// Преобразуем массив байтов в текст с учетом

// умолчальной кодовой страницы.

//CodePageEncoding cpe = new CodePageEncoding(0);

//tbTextForSearch.Text = cpe.GetString(bt);

return Encoding.UTF8.GetString(bt);

}

}

catch

{

return "";

}

}

/// <summary>

/// Загружает содержимое файла в текстовое поле /\*Взято с stackoverflow.com\*/

/// </summary>

/// <param name="fileName">Имя файла с путём</param>

/// <param name="codePage">Кодировочная таблица</param>

/// <returns>Содержимое файла</returns>

public static string LoadFile(string fileName, int codePage)

{

try

{

// Считываем файл в массив байтов...

using (var fs = new FileStream(fileName, FileMode.Open, FileAccess.Read))

{

var bt = new byte[(int)fs.Length];

fs.Read(bt, 0, (int)fs.Length);

// Преобразуем массив байтов в текст с учетом

// кодовой страницы по умолчанию.

//CodePageEncoding cpe = new CodePageEncoding(0);

//tbTextForSearch.Text = cpe.GetString(bt);

return codePage <= 0 ? Encoding.UTF8.GetString(bt) : Encoding.GetEncoding(codePage).GetString(bt);

}

}

catch

{

return "";

}

}

#endregion

#region SaveFile

/// <summary>

/// Сохраняет текстовый файл

/// </summary>

/// <param name="fileName">Имя файла с путём</param>

/// <param name="text">Содержимое файла</param>

/// <returns></returns>

public static string SaveFile(string fileName, string text)

{

return SaveFile(fileName, text, Encoding.GetEncoding(1251));

}

/// <summary>

/// Сохраняет текстовый файл

/// </summary>

/// <param name="fileName">Имя файла с путём</param>

/// <param name="text">Содержимое файла</param>

/// <param name="codePage">Номер кодировочной таблицы</param>

/// <returns></returns>

public static string SaveFile(string fileName, string text, int codePage)

{

return SaveFile(fileName, text, Encoding.GetEncoding(codePage));

}

/// <summary>

/// Сохраняет текстовый файл /\*Взято с stackoverflow.com\*/

/// </summary>

/// <param name="fileName">Имя файла с путём</param>

/// <param name="text">Содержимое файла</param>

/// <param name="encoding">Кодировочная таблица для сохранения</param>

/// <returns></returns>

public static string SaveFile(string fileName, string text, Encoding encoding)

{

try

{

if (File.Exists(fileName))

{

//Если файл существует, удаляем его

File.Delete(fileName);

}

CheckFilePath(fileName);

using (var fs = new FileStream(fileName, FileMode.CreateNew))

{

using (var sw = new StreamWriter(fs, encoding))

{

sw.Write(text);

sw.Close();

}

}

}

catch (Exception ex)

{

return ex.Message;

}

return "";

}

#endregion

}

Форма DirectoriesEditorForm #region Specialization

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

{

var data = DirectoryItemEditorForm.AddSpec();

if (data != null)

{

DataManager.Specializations.Add(data);

lvSpec.Items.Add(new ListViewItem(new string[] { data.Id.ToString(), data.Name }));

}

}

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

{

EditSpec();

}

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

{

if (lvSpec.Items == null || lvSpec.Items.Count == 0)

{

return;

}

string txtId;

string txt;

if (lvSpec.SelectedItems == null || lvSpec.SelectedItems.Count == 0)

{

lvSpec.Items[0].Selected = true;

txtId = lvSpec.Items[0].Text;

txt = string.Format("Are you delete record '{0}. {1}'", txtId, lvSpec.Items[0].SubItems[1].Text);

}

else

{

txtId = lvSpec.SelectedItems[0].Text;

txt = string.Format("Are you delete record '{0}. {1}'", txtId, lvSpec.SelectedItems[0].SubItems[1].Text);

}

if (MessageBox.Show(txt, "Delete specialization", MessageBoxButtons.YesNo) == DialogResult.Yes)

{

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

{

if (lvSpec.Items[i].Text == txtId)

{

DataManager.SpecDelete(Convert.ToInt32(txtId));

lvSpec.Items.RemoveAt(i);

break;

}

}

}

}

private void lvSpec\_MouseDoubleClick(object sender, MouseEventArgs e)

{

EditSpec();

}

private void EditSpec()

{

if (lvSpec.Items == null || lvSpec.Items.Count == 0)

{

return;

}

string txtId;

string txt;

if (lvSpec.SelectedItems == null || lvSpec.SelectedItems.Count == 0)

{

lvSpec.Items[0].Selected = true;

txtId = lvSpec.Items[0].Text;

txt = lvSpec.Items[0].SubItems[1].Text;

}

else

{

txtId = lvSpec.SelectedItems[0].Text;

txt = lvSpec.SelectedItems[0].SubItems[1].Text;

}

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

{

if (lvSpec.Items[i].Text == lvSpec.SelectedItems[0].Text)

{

var obj = new Specialization(Convert.ToInt32(txtId), txt);

var data = DirectoryItemEditorForm.EditSpec(obj);

DataManager.SpecEdit(data);

lvSpec.SelectedItems[0].SubItems[1].Text = data.Name;

break;

}

}

}

Форма DirectoryItemEditorForm

public partial class DirectoryItemEditorForm : Form

{

public DirectoryItemEditorForm()

{

InitializeComponent();

}

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

{

if (e.KeyCode == Keys.Enter)

{

DialogResult = DialogResult.OK;

}

if (e.KeyCode == Keys.Escape)

{

DialogResult = DialogResult.Cancel;

}

}

public static TypeOfOwnership AddToo()

{

var win = new DirectoryItemEditorForm();

var id = DataManager.TooGetNewId();

win.lbId.Text = id.ToString();

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

{

var obj = new TypeOfOwnership(Convert.ToInt32(win.lbId.Text), win.tbName.Text);

return obj;

}

return null;

}

public static TypeOfOwnership EditToo(TypeOfOwnership data)

{

var win = new DirectoryItemEditorForm

{

lbId = {Text = data.Id.ToString()},

tbName = {Text = data.Name}

};

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

{

var obj = new TypeOfOwnership(Convert.ToInt32(win.lbId.Text), win.tbName.Text);

return obj;

}

return data;

}

public static Specialization AddSpec()

{

var win = new DirectoryItemEditorForm();

var id = DataManager.SpecGetNewId();

win.lbId.Text = id.ToString();

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

{

var obj = new Specialization(Convert.ToInt32(win.lbId.Text), win.tbName.Text);

return obj;

}

return null;

}

public static Specialization EditSpec(Specialization data)

{

var win = new DirectoryItemEditorForm

{

lbId = {Text = data.Id.ToString()},

tbName = {Text = data.Name}

};

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

{

var obj = new Specialization(Convert.ToInt32(win.lbId.Text), win.tbName.Text);

return obj;

}

return data;

}

}

Форма MainForm

public partial class MainForm : Form

{

private bool flagStartApplication = true;

public MainForm()

{

InitializeComponent();

}

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

{

pnEditor.Visible = !pnEditor.Visible;

}

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

{

var win = new DirectoriesEditorForm();

win.ShowDialog();

InitData();

}

/// <summary>

/// Метод вызывается при первом показе формы

/// </summary>

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

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

private void MainForm\_Shown(object sender, EventArgs e)

{

DataManager.LoadTypeOfOwnership();

DataManager.LoadSpecialization();

DataManager.LoadFirm();

InitData();

}

private void InitData()

{

flagStartApplication = true;

cbSpect.Items.Clear();

cbSpect.Items.Add(new Specialization(0, "None")); //Служебная запись показывающая что фильтр не задан

if (DataManager.Specializations != null)

{

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

{

cbSpect.Items.Add(DataManager.Specializations[i]);

}

}

cbSpect.SelectedIndex = 0;

cbToo.Items.Clear();

cbToo.Items.Add(new TypeOfOwnership(0, "None")); //Служебная запись показывающая что фильтр не задан

if (DataManager.TypeOfOwnerships != null)

{

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

{

cbToo.Items.Add(DataManager.TypeOfOwnerships[i]);

}

}

cbToo.SelectedIndex = 0;

lvFirms.Items.Clear();

foreach (Firm data in DataManager.Firms)

{

var lvi = CreateListViewItem(data);

lvFirms.Items.Add(lvi);

}

flagStartApplication = false;

}

private static ListViewItem CreateListViewItem(Firm data)

{

var lvi = new ListViewItem(new string[]

{

data.Id.ToString(),

data.Name,

data.SpecName,

data.TooName,

data.TimeWork

});

lvi.Tag = data;

return lvi;

}

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

{

var obj = FirmEditorForm.AddFirm();

if (obj != null)

{

DataManager.Firms.Add(obj);

DataManager.SaveFirm();

lvFirms.Items.Add(CreateListViewItem(obj));

}

}

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

{

EditFirm();

}

private void lvFirms\_MouseDoubleClick(object sender, MouseEventArgs e)

{

EditFirm();

}

private void EditFirm()

{

if (lvFirms.Items == null || lvFirms.Items.Count == 0)

{

return;

}

Firm data = null;

if (lvFirms.SelectedItems == null || lvFirms.SelectedItems.Count == 0)

{

lvFirms.Items[0].Selected = true;

data = (Firm)lvFirms.Items[0].Tag;

}

else

{

data = (Firm)lvFirms.SelectedItems[0].Tag;

}

var obj = FirmEditorForm.EditFirm(data);

if (obj != null)

{

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

{

if (DataManager.Firms[i].Id == obj.Id)

{

DataManager.Firms[i] = obj;

break;

}

}

DataManager.SaveFirm();

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

{

if (((Firm) lvFirms.Items[i].Tag).Id == obj.Id)

{

lvFirms.Items[i] = CreateListViewItem(obj);

break;

}

}

}

}

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

{

Firm data = null;

if (lvFirms.SelectedItems == null || lvFirms.SelectedItems.Count == 0)

{

lvFirms.Items[0].Selected = true;

data = (Firm)lvFirms.Items[0].Tag;

}

else

{

data = (Firm)lvFirms.SelectedItems[0].Tag;

}

if (MessageBox.Show(string.Format("Do you delete firm '{0}'", data.Name), "Delete firm", MessageBoxButtons.YesNo) == DialogResult.Yes)

{

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

{

if (DataManager.Firms[i].Id == data.Id)

{

DataManager.Firms.RemoveAt(i);

break;

}

}

DataManager.SaveFirm();

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

{

if (((Firm)lvFirms.Items[i].Tag).Id == data.Id)

{

lvFirms.Items.RemoveAt(i);

break;

}

}

}

}

private void lvFirms\_SelectedIndexChanged(object sender, EventArgs e)

{

if (lvFirms.SelectedItems == null || lvFirms.SelectedItems.Count == 0)

{

return;

}

var data = (Firm)lvFirms.SelectedItems[0].Tag;

tbAddress.Text = data.Address;

tbPhone.Text = data.Phones;

}

private void cbFiler\_SelectedIndexChanged(object sender, EventArgs e)

{

DoSearch();

}

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

{

if (e.KeyCode == Keys.Enter)

{

DoSearch();

}

}

private void DoSearch()

{

if (flagStartApplication || cbSpect.SelectedItem == null || cbToo.SelectedItem == null)

{

return;

}

var specId = ((Specialization) cbSpect.SelectedItem).Id;

var tooId = ((TypeOfOwnership)cbToo.SelectedItem).Id;

var searchTxt = tbSearch.Text.ToUpper();

lvFirms.Items.Clear();

tbAddress.Text = string.Empty;

tbPhone.Text = string.Empty;

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

{

var data = DataManager.Firms[i];

if (specId > 0 && specId != data.SpecId)

{

continue;

}

if (tooId > 0 && tooId != data.TooId)

{

continue;

}

if (!string.IsNullOrEmpty(searchTxt))

{

var flag = (!string.IsNullOrEmpty(data.Name) && data.Name. ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1) ||

(!string.IsNullOrEmpty(data.Address) && data.Address. ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1) ||

(!string.IsNullOrEmpty(data.Phones) && data.Phones. ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1) ||

(!string.IsNullOrEmpty(data.TimeWork) && data.TimeWork.ToUpper().IndexOf(searchTxt, StringComparison.Ordinal) > -1);

if (!flag)

{

continue;

}

}

lvFirms.Items.Add(CreateListViewItem(data));

}

}

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

{

tbSearch.Text = string.Empty;

}

}