НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Программирование»

**Контрольное домашнее задание №2**

**Вариант 13**

Выполнил: Ярных Роман,

студент группы 101ПИ.

Преподаватель: Подбельский Вадим Валериевич,

доктор технических наук,

профессор программной инженерии

факультета компьютерных наук

Дата сдачи: 18.03.2015

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва 2015

Оглавление

Условие задачи 3

Функции разрабатываемого приложения 4

Варианты использования 4

Описание интерфейса пользователя 4

Структура приложения 8

Диаграмма классов 8

Описание классов, их полей и методов 8

Распределение исходного кода по файлам проекта 10

Контрольный пример и описание результатов 11

Приложение 13

Список литературы 19

# Условие задачи – Вариант 13

Программа Контрольного домашнего задания представляет собой **WindowsForms** приложение и предназначена для просмотра, обработки и сохранения результатов обработки данных из файла.

**reestr-advokatov.csv**

1. Требования к основным классам приложения
   1. Основная информация о Членах коллегии адвокатов хранится в объектах класса «Адвокат». Набор полей класса задаётся полями **csv** файла, кроме полей, содержащих адрес и телефон. Реквизиты решения о присвоении статуса адвоката представлены полем типа «Реквизиты». Класс «Адвокат» находится в отношении композиции с классом «Реквизиты».
   2. Класс «Реквизиты» представляет номер постановления и дату.
   3. Дополнительные классы, необходимые для решения задачи.
2. Требования к функциям приложения
   1. Программа должна позволять
      1. Открыть **\*.csv** файл с исходными данными. Использовать **OpenFileDialog**.
         1. *Ограничение.* В случае ошибок открывания файла или некорректных данных программа должна выводить сообщение.
      2. Отсортировать данные по полям: **Фамилия**, **Имя**
      3. Отфильтровать данные по полям: **Приостановление статуса адвоката**, **Возобновление статуса адвоката**
      4. Сохранять результаты сортировок и фильтраций в файл формата **csv**. Использовать **SaveFileDialog**.
         1. *Ограничение*. В случае ошибок открывания файла или некорректных данных, программа должна выводить сообщение.
         2. *Режимы* сохранения в файл
            1. создание нового файла.
            2. замена содержимого уже существующего файла.
            3. добавить сохраняемые данные к содержимому уже существующего файла.
3. Требования к интерфейсу
   1. Для отображения данных использовать сетку **DataGridView**
   2. Количество отображаемых в сетке элементов (***N***) выбирается пользователем, ***N* > 1** и не превышает количества записей в **csv** файле.
4. Требования к устойчивости приложения
   1. Аварийные ситуации должны обрабатываться, пользователю должны выводиться информативные сообщения.

# Функции разрабатываемого приложения

## Варианты использования

Программа имеет практическое значение в области обработки, хранения и передачи данных. С ее помощью пользователь может открывать файлы реестров адвокатов, фильтровать и сортировать поля по ключам, добавлять, изменять и удалять записи, а также сохранять их в файл. В качество входных и выходных данных используются текстовые файлы формата “.csv”.

## Описание интерфейса пользователя

Интерфейс приложения состоит из главного окна, где пользователь может открывать файл, отображать из него в таблице записи об адвокатах, изменять и удалять их и сохранять все изменения в файл, а также из вспомогательных диалоговых окон. Главное окно приложения (Рис. 1) содержит следующие пользовательские элементы управления:

* пункты главного меню “Файл”, “Помощь”;
* кнопки “Отобразить”, “Сортировать”, “Отфильтровать”, “Добавить запись”, “Удалит запись”, “Очистить” и “Изменить”;
* текстовое поле для ввода “Число строк”;
* выпадающий список “Фильтрация по полю”;
* флажки “Фамилия”, “Имя”.

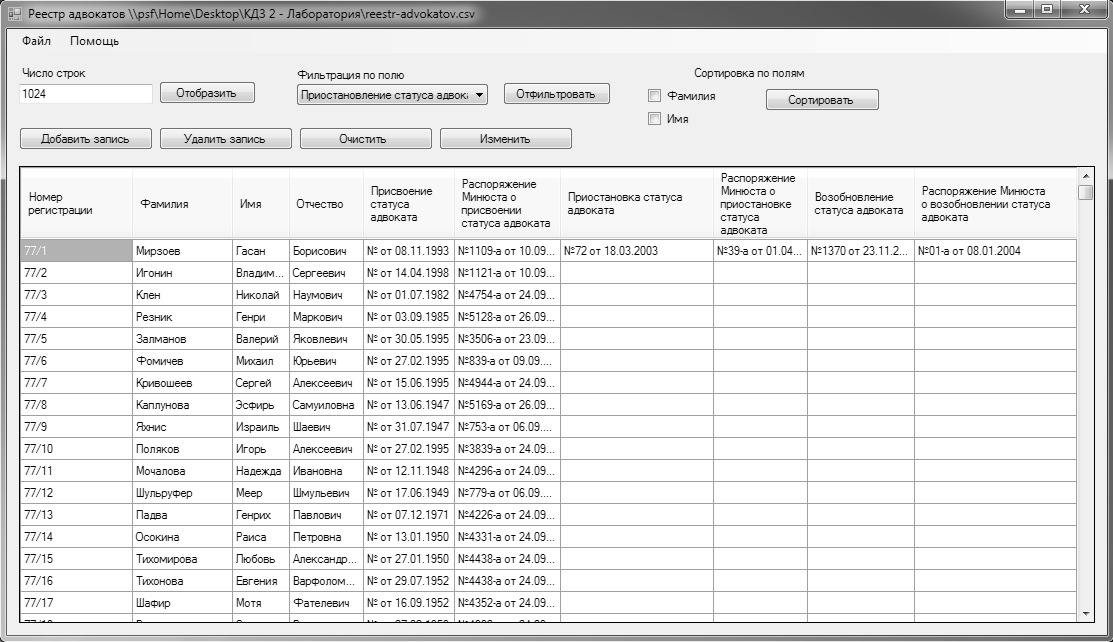


Рис. 1

Работа с программой состоит из следующих шагов:

1) нажать на пункт “Открыть” в меню “Файл” (Рис. 1) или перенести иконку файла в таблицу (Рис. 2);

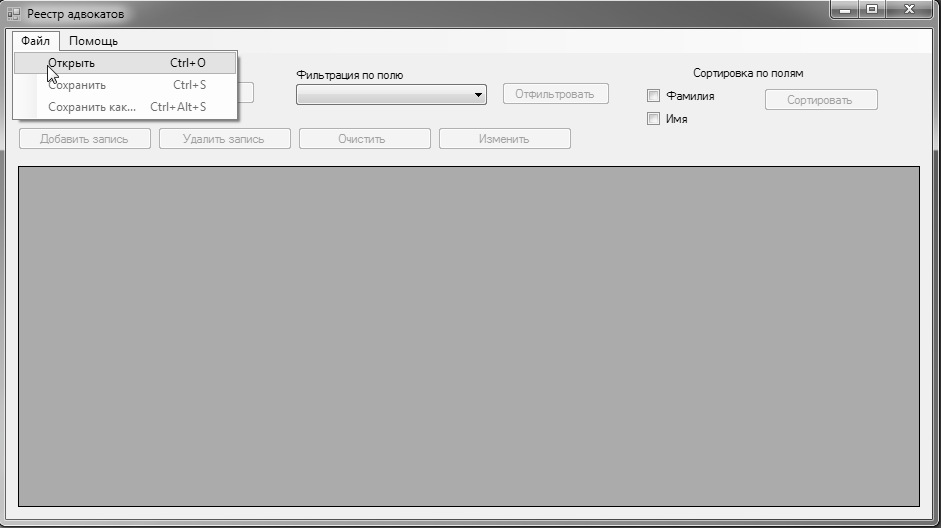


Рис. 2

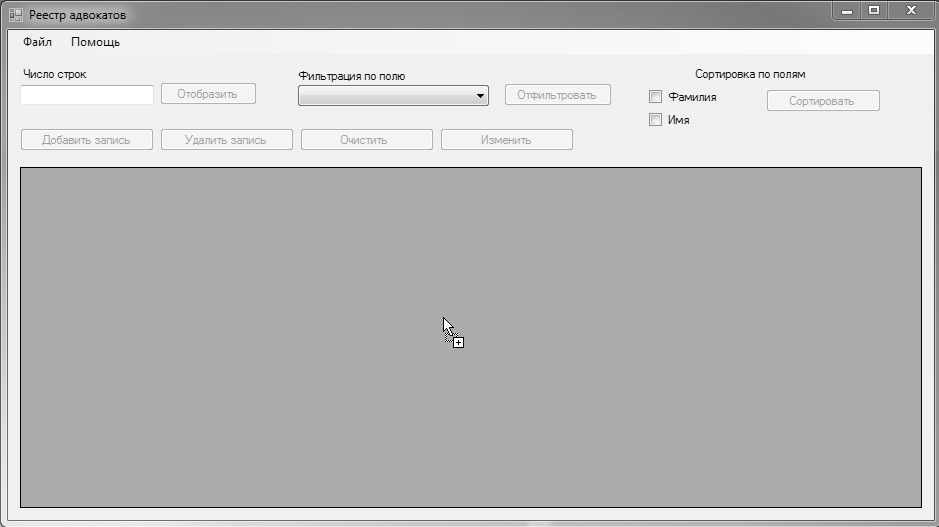


Рис. 3

2) ввести число строк для отображения и нажать на кнопку “Отобразить” (Рис. 4), после чего данные из файла будут загружены в таблицу (Рис. 5);



Рис. 4

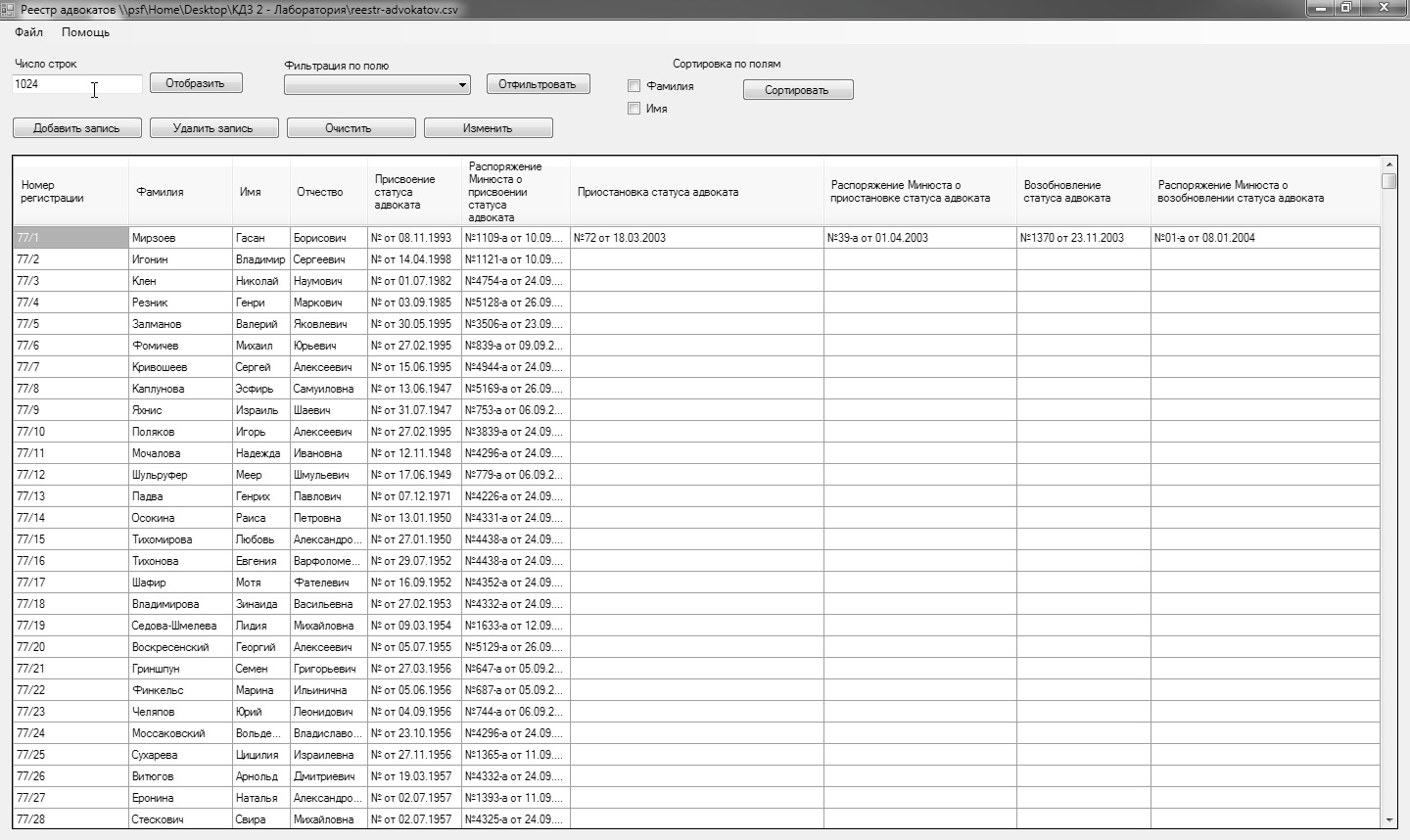


Рис. 5

3) для фильтрации по полям “Приостановление статуса адвоката” или “Возобновление статуса адвоката” нужно нажать кнопку “Отфильтровать” (Рис. 6);

OS X:Users:egdeveloper:Documents:ShareX:Screenshots:2015-03:2015-03-17_15-57-24.png

Рис. 6

# 4) чтобы отсортировать записи об адвокатах, необходимо выбрать поля, по которым будет проходить сортировка, и нажать на кнопку “Сортировать” (Рис. 7);

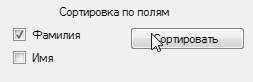


Рис. 7

5) чтобы сохранить все изменения в файл, нужно найти в меню “Файл” пункты “Сохранить”, “Сохранить как…” и нажать на один из них (Рис. 8).

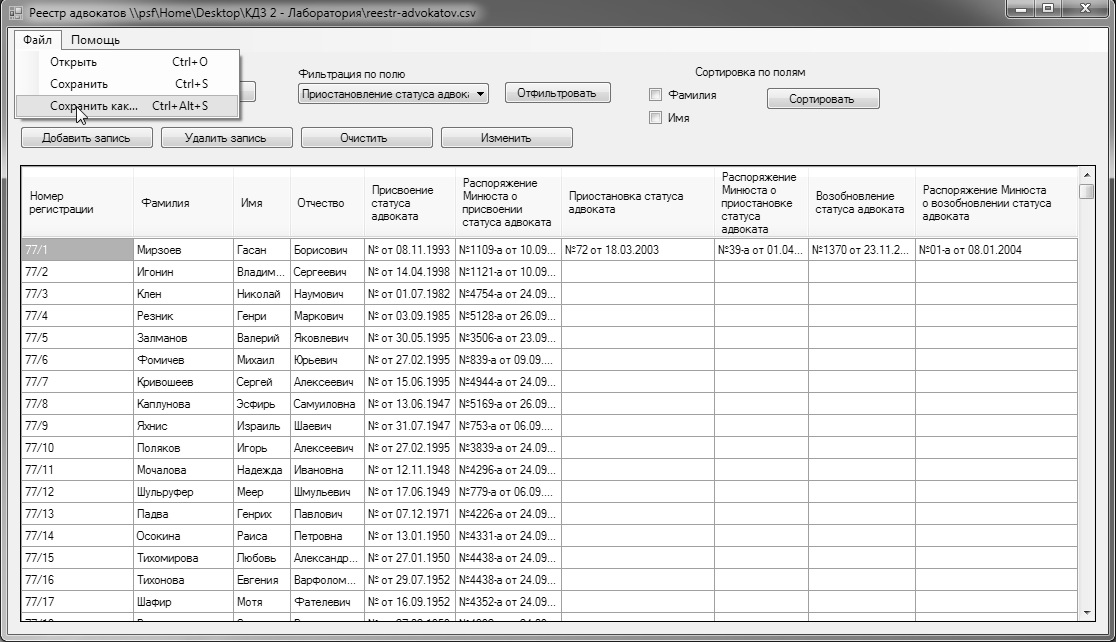


Рис. 8

# 6) чтобы добавить новые записи в реестр, необходимо нажать на кнопку “Добавить запись” (Рис. 9) и в диалоговом окне ввести информацию об адвокате (Рис. 10).

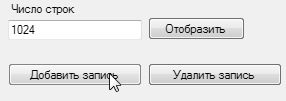


Рис. 9

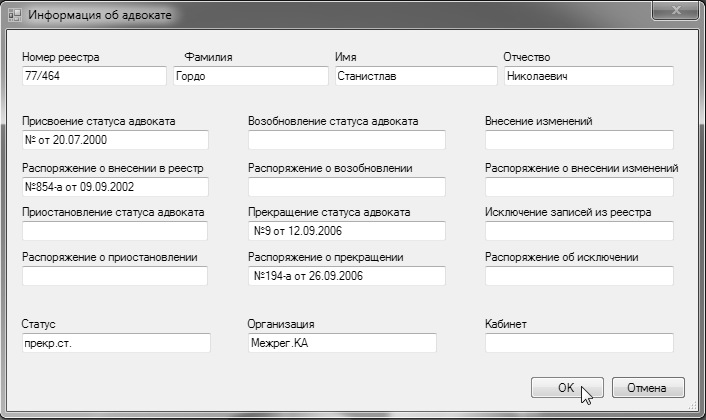


Рис. 10

# Структура приложения

## Диаграмма классов

На рис. 9 показан граф связей между классами в проекте.

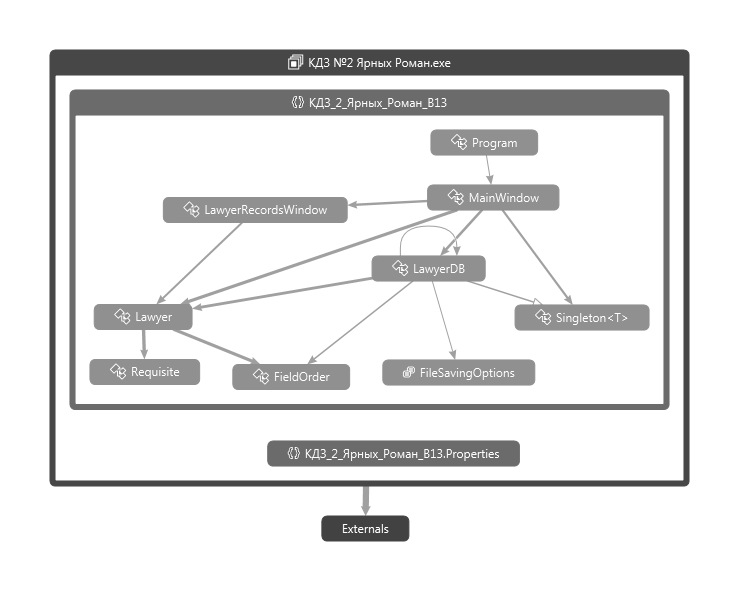


Рис. 9

## Описание классов, их полей и методов

Весь проект состоит из 5 вручную написанных классов: Lawyer, Requisite, Singleton, LawyerDB, FieldOrder.

Класс **Lawyer** предоставляет клиентам удобный контейнер для хранения информации об адвокатах. **Lawyer** находится в композиции с классом **Requisite**, который хранит реквизиты адвокатов и делегирует первому обращения клиентов к своим полям. Класс **Singleton** реализует шаблон проектирования “Одиночка” и наделяет своих потомков механизмом создания одиночного экземпляра. Он состоит из следующих членов:

* внутренний класс **SingletonFactory**, обеспечивающий динамическое создание экземпляра потомка **Singleton**;
* свойство **Instance**, позволяющее клиентам безопасно обращаться к членам потомка класса **Singleton**.

Класс **LawyerDB** обрабатывает запросы клиентов к файлу реестра адвокатов и инкапсулирует все необходимые операции над ним (открытие, редактирование полей, сохранение). Он состоит из следующих членов:

* метод **InitDB**, инициализирующий базу данных и подготавливающий ее к дальнейшей работе;
* метод **ReadDB**, читающий данные из файла и передающий их в виде списка экземпляров **Lawyer**;
* метод **WriteDBToFile**, преобразующий список экземпляров **Lawyer** в данные реестра адвокатов и сохраняющий их в файл по указанному пути;
* закрытый метод **ParseLine**, обрабатывающий строки файла реестра адвокатов и возвращающий их в виде экземпляра **Lawyer**.

# Распределение исходного кода по файлам проекта

**Lawyer.cs**

Файл содержит определение класса-контейнера **Lawyer** для хранения и передачи данных адвокатов.

**Singleton.cs**

Файл содержит определение класса **Singleton**, служащего для обеспечения контроля за созданием единственного экземпляра и глобального доступа к нему в пределах пространства имен.

**LawyerModel.cs**

Файл содержит определение класса **LawyerDB**, служащего для обработки запросов клиентов к базе данных, редактирования полей реестра адвокатов и сохранения их в локальных файл по указанному пути.

**MainWindow.cs**

Файл содержит определение класса главного окна **MainWindow**, служащего для отображения данных, хранящихся в файле и обработки запросов клиентов к ним.

**LawyerRecordsWindow.cs**

Файл содержит определение класса диалогового окна **LawyerRecordsWindow**, служащего для просмотра, изменения и сохранения записей об адвокате.

# Контрольный пример и описание результатов

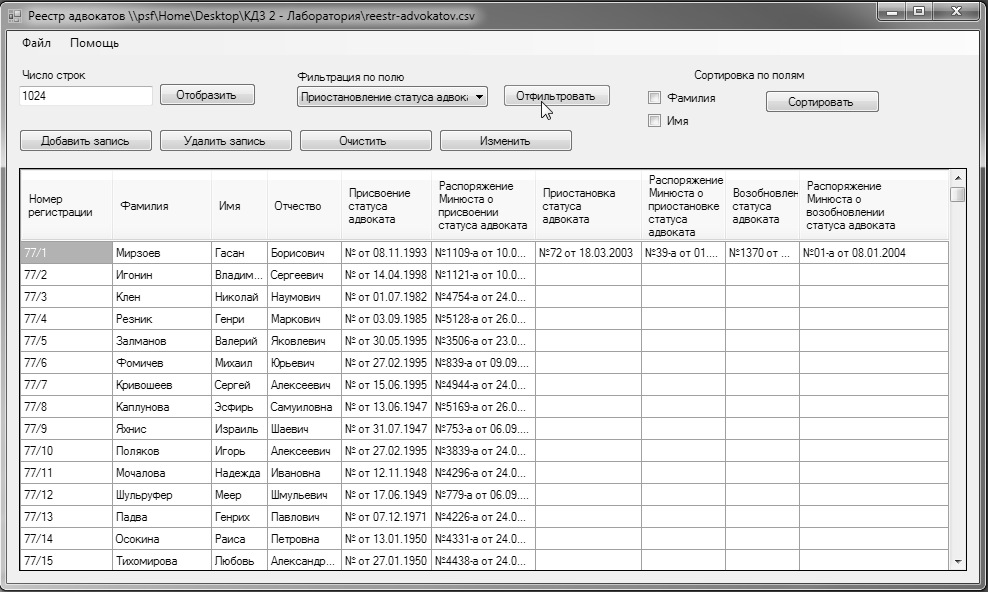


Рис. 10

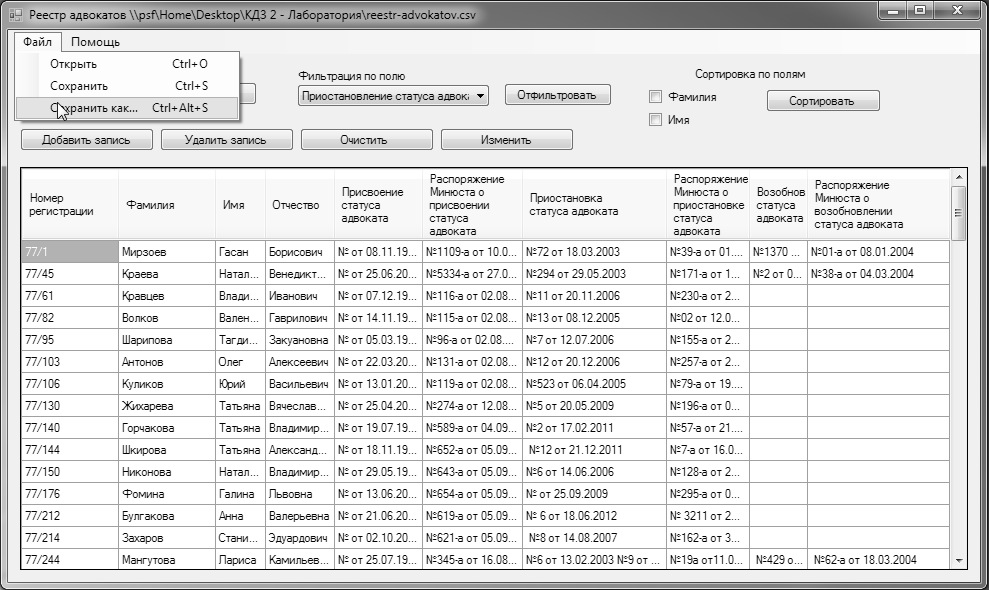


Рис. 11

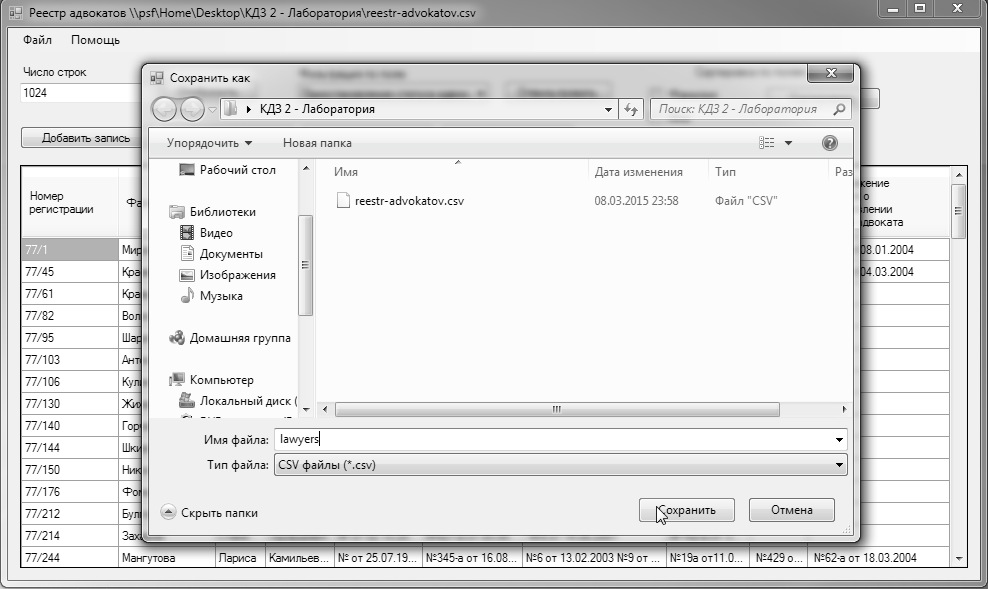


Рис. 12



Рис. 13

Как видно на рис. 10, программа успешно открыла и загрузила данные из тестового файла в таблицу. Затем при нажатии кнопки “Отфильтровать” таблица обновила поля и отобразила только те, в которых есть запись о приостановке статуса адвоката (Рис. 11). После этого по запросу пользователя программа вывела диалоговое окно для сохранения файла. (Рис. 12). Все изменения, которые мы внесли в реестр, были успешно сохранены (Рис. 13)

# Приложение

**Lawyer.cs**

using System;

using System.ComponentModel;

namespace КДЗ\_2\_Ярных\_Роман\_В13

{

public class Requisite

{

public string Assignment { get; set; }

public string AssignmentEntry { get; set; }

public string Suspension { get; set; }

public string SuspensionEntry { get; set; }

public string Resumption { get; set; }

public string ResumptionEntry { get; set; }

public string Termination { get; set; }

public string TerminationEntry { get; set; }

public string ChangeLog { get; set; }

public string ChangeLogEntry { get; set; }

public string ClearRecord { get; set; }

public string ClearRecordEntry { get; set; }

public string Status { get; set; }

public string Org { get; set; }

public string Cab { get; set; }

}

[Serializable]

public class Lawyer

{

private Requisite requisite\_ = new Requisite();

[DisplayName("Номер регистрации")]

[FieldOrder(1)]

public string RegistryNumber { get; set; }

[DisplayName("Фамилия")]

[FieldOrder(2)]

public string LastName { get; set; }

[DisplayName("Имя")]

[FieldOrder(3)]

public string FirstName { get; set; }

[DisplayName("Отчество")]

[FieldOrder(4)]

public string Patronym { get; set; }

[DisplayName("Присвоение статуса адвоката")]

[FieldOrder(5)]

public string Assignment {

set { requisite\_.Assignment = value; }

get { return requisite\_.Assignment; }

}

[DisplayName("Распоряжение Минюста о присвоении статуса адвоката")]

[FieldOrder(6)]

public string AssignmentEntry {

set { requisite\_.AssignmentEntry = value; }

get { return requisite\_.AssignmentEntry; }

}

[DisplayName("Приостановка статуса адвоката")]

[FieldOrder(7)]

public string Suspension {

set { requisite\_.Suspension = value; }

get { return requisite\_.Suspension; }

}

[DisplayName("Распоряжение Минюста о приостановке статуса адвоката")]

[FieldOrder(8)]

public string SuspensionEntry {

set { requisite\_.SuspensionEntry = value; }

get { return requisite\_.SuspensionEntry; }

}

[DisplayName("Возобновление статуса адвоката")]

[FieldOrder(9)]

public string Resumption {

set { requisite\_.Resumption = value; }

get { return requisite\_.Resumption; }

}

[DisplayName("Распоряжение Минюста о возобновлении статуса адвоката")]

[FieldOrder(10)]

public string ResumptionEntry {

set { requisite\_.ResumptionEntry = value; }

get { return requisite\_.ResumptionEntry; }

}

[Browsable(false)]

[FieldOrder(11)]

public string Termination {

set { requisite\_.Termination = value; }

get { return requisite\_.Termination; }

}

[Browsable(false)]

[FieldOrder(12)]

public string TerminationEntry {

set { requisite\_.TerminationEntry = value; }

get { return requisite\_.TerminationEntry; }

}

[Browsable(false)]

[FieldOrder(13)]

public string ChangeLog {

set { requisite\_.ChangeLog = value; }

get { return requisite\_.ChangeLog; }

}

[Browsable(false)]

[FieldOrder(14)]

public string ChangeLogEntry {

set { requisite\_.ChangeLogEntry = value; }

get { return requisite\_.ChangeLogEntry; }

}

[Browsable(false)]

[FieldOrder(15)]

public string ClearRecord {

set { requisite\_.ClearRecord = value; }

get { return requisite\_.ClearRecord; }

}

[Browsable(false)]

[FieldOrder(16)]

public string ClearRecordEntry {

set { requisite\_.ClearRecordEntry = value; }

get { return requisite\_.ClearRecordEntry; } }

[Browsable(false)]

[FieldOrder(17)]

public string Status { get; set; }

[Browsable(false)]

[FieldOrder(18)]

public string Org { get; set; }

[Browsable(false)]

[FieldOrder(19)]

public string Cab { get; set; }

[Browsable(false)]

[FieldOrder(20)]

public Requisite Requisite { set { requisite\_ = value; } }

}

}

**Singleton.cs**

using System;

using System.Reflection;

namespace КДЗ\_2\_Ярных\_Роман\_В13

{

/// <summary>

/// Базовый класс, гарантирующий своим потомкам создание единственного экземпляра

/// </summary>

/// <typeparam name="T"></typeparam>

abstract class Singleton<T> where T : class

{

private class SingletonFactory<S> where S : class

{

private static readonly S instance\_ = (S)typeof(S).GetConstructor(BindingFlags.NonPublic | BindingFlags.Instance, null, new Type[0], new ParameterModifier[0]).Invoke(new object[0]);

public static S GetInstance()

{

return instance\_;

}

}

protected Singleton() { }

public static T Instance

{

get { return SingletonFactory<T>.GetInstance(); }

}

}

}

**LawyerDB.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

using System.ComponentModel;

using System.Reflection;

namespace КДЗ\_2\_Ярных\_Роман\_В13

{

enum FileSavingOptions { Rewrite, AddToEnd}

class LawyerDB : Singleton<LawyerDB> //делаем класс синглтоном

{

private string header\_;

private string path\_;

private List<Lawyer> lawyer\_list\_;

private const int COLUMNS\_LEN = 19;

private LawyerDB() { }

/// <summary>

/// Инициализация БД

/// </summary>

/// <param name="path"></param>

public void InitDB(string path)

{

if (!File.Exists(path)) //если не существует файла, выбрасываем исключение

throw new IOException(String.Format("Файл {0} не найден", path));

path\_ = path;

lawyer\_list\_ = new List<Lawyer>();

}

/// <summary>

/// Чтение полей БД

/// </summary>

/// <param name="n"></param>

/// <returns></returns>

public IEnumerable<Lawyer> ReadDB(int n)

{

string[] lines = File.ReadAllLines(path\_);

header\_ = lines[0];

int k = 0;

string line = "";

int len = 0;

bool context\_ = false;

for (int i = 1; k < n; i++)

{

lines[i].Trim('\n', ' ');

Array.ForEach(lines[i].ToArray(), x =>

{

context\_ ^= x == '\"';

if (!context\_ && x == ',')

len++;

});

line += lines[i];

if (len == COLUMNS\_LEN - 1)

{

len = 0;

yield return ParseLine(line);

line = "";

k++;

}

}

}

/// <summary>

/// Запись данных БД в файл

/// </summary>

/// <param name="lawyers"></param>

/// <param name="path"></param>

public void WriteDBToFile(IEnumerable<Lawyer> lawyers, string path, FileSavingOptions options)

{

using (FileStream fileStream = new FileStream(path ?? path\_, options == FileSavingOptions.Rewrite ? FileMode.Create : FileMode.Append))

{

using (StreamWriter streamWriter = new StreamWriter(fileStream))

{

if(options == FileSavingOptions.Rewrite)

streamWriter.WriteLine(header\_);

foreach (var lawyer in lawyers)

{

string line = "";

int k = 0;

PropertyInfo[] properties = typeof(Lawyer).GetProperties(BindingFlags.Public | BindingFlags.Instance).Where(x => x.PropertyType == typeof(string)).ToArray();

foreach (var property in properties.OrderBy(x => ((FieldOrder)x.GetCustomAttributes(typeof(FieldOrder), true)[0]).Order))

{

string property\_value = (string)property.GetValue(lawyer, null);

if (property\_value.Contains(',') || property\_value.Contains('"'))

line += "\"" + property\_value.Replace("\"", "\"\"") + "\"";

else line += property\_value;

if (++k < COLUMNS\_LEN)

line += ",";

}

streamWriter.WriteLine(line);

}

}

}

}

/// <summary>

/// Обработка строкового представления данных

/// </summary>

/// <param name="line"></param>

/// <returns></returns>

private Lawyer ParseLine(string line)

{

Lawyer lawyer = new Lawyer();

string[] columns\_ = line.Split(',');

PropertyInfo[] properties = typeof(Lawyer).GetProperties(BindingFlags.Public | BindingFlags.Instance).Where(x => x.PropertyType == typeof(string)).ToArray();

int k = -1;

foreach (var property in properties.OrderBy(x => ((FieldOrder)x.GetCustomAttributes(typeof(FieldOrder), true)[0]).Order))

{

property.SetValue(lawyer, columns\_[++k].Replace("\"", "\"\"").Replace("\"\"", ""), null);

}

return lawyer;

}

}

}

# Список литературы

1. Герберт Шилдт. C# 4.0. Полный справочник.

2. Эндрю Троелсен. C# 4.0 и платформа .NET 4.0.

3. MSDN (Microsoft Developer Network) – электронный справочный ресурс для разработчиков.