Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Департамент
Программной инженерии

Контрольное домашнее задание
по дисциплине
«Программирование»

Тема работы: Поликлиническая помощь взрослым версия 2.4 от 20.10.2014

Выполнил: студент группы БПИ176 (2)
_____ Ракитин А. А.

тел. +79778123289
e-mail адрес: aarakitin@edu.hse.ru

Преподаватель: Чуйкин Н. К.

Москва, 2018 год. Модуль 3

ОГЛАВЛЕНИЕ

Ракитин Александр Александрович
Вариант №10

1. УСЛОВИЕ ЗАДАЧИ

Вариант 10.

1. Требования к основным классам приложения

1.1. Основная информация о Взрослых поликлиниках хранится в объектах класса **Поликлиника**. Набор полей класса задаётся полями файла *Поликлиническая помощь взрослым версия 2.4 от 20.10.2014.csv*, кроме полей, содержащих информацию о его адресе, представленном полем типа **Адрес**. Класс Поликлиника находится в *отношении композиции* с классом Адрес.

1.2. Класс Адрес представляет адрес, заданный полями CSV-файла: **AdmArea, District, Address, Point X, Point Y**. Один из методов класса возвращает координаты ближайшие (расстояние измеряется по прямой) к координатам, переданным в качестве параметров.

1.3. Дополнительные классы, необходимые для решения задачи.

2. Приложение должно поддерживать следующие функции

2.1. Открыть CSV-файл (*.csv) с исходными данными и проверить корректность данных в нём.

2.2. Загрузить данные из CSV-файла в объекты классов Поликлиника, Адрес (объект Адрес является уникальным для каждого объекта Поликлиника) и др.

2.3. Отобразить данные из объектов в оконной форме.

2.4. Создать новую запись о Поликлинике.

2.5. Удалить уже существующую запись о Поликлинике.

2.6. Отредактировать существующую запись о Поликлинике.

2.7. Отсортировать данные по полям: **PostalCode, ShortName**

2.8. Отфильтровать данные по полям: **PaidServicesInfo, District**

2.9. Найти по выбранной в списке Поликлинике другую ближайшую к ней поликлинику (использовать метод класса Адрес)

2.10. Сохранять результаты редактирований, сортировок и фильтраций в CSV файл. Режимы сохранения в файл: создание нового файла, замена содержимого уже существующего файла, добавление сохраняемых данных к содержимому существующего файла.

3. Требования к интерфейсу

3.1. При управлении файлом (загрузка, сохранение) использовать **OpenFileDialog** и **SaveFileDialog**.

3.2. Для отображения данных использовать сетку **DataGridView**. Количество отображаемых в сетке элементов (N) выбирается пользователем, $N > 1$ и не превышает количества записей в файле *Поликлиническая помощь взрослым версия 2.4 от 20.10.2014.csv*.

4. Требования к устойчивости приложения

4.1. В случае ошибок открывания/сохранения файла или некорректных данных программа должна выводить сообщение.

4.2. Аварийные ситуации должны обрабатываться, пользователю должны выводиться информативные сообщения.

2. ФУНКЦИИ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ

2.1. Варианты использования

Программа должна быть использована для создания, просмотра и редактирования файла Поликлиническая помощь взрослым версия 2.4 от 20.10.2014.csv или другого файла, имеющего аналогичное строение.

2.2. Описание интерфейса программы

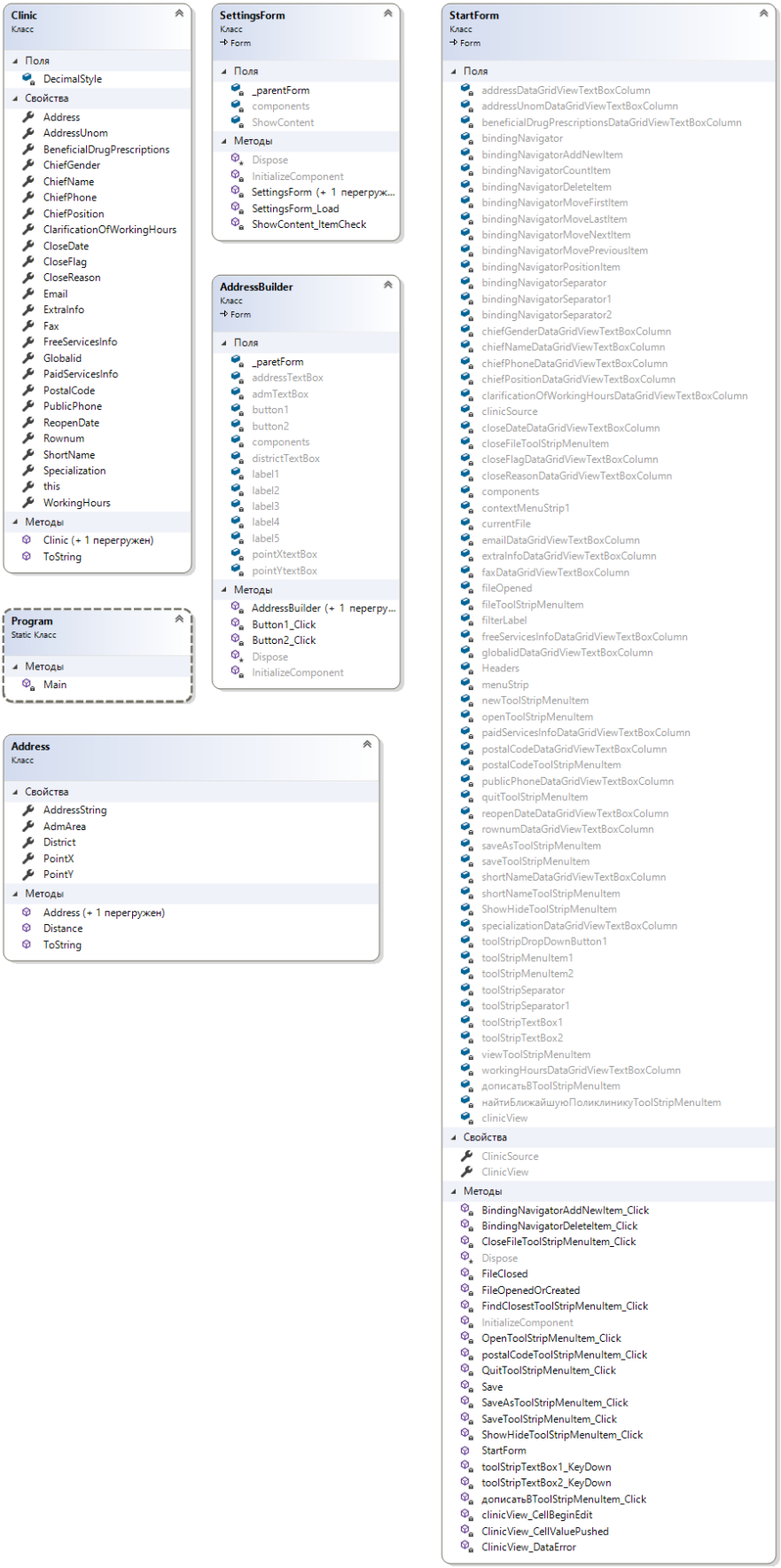
При первом запуске программа не показывает сетку. Чтобы ее увидеть необходимо открыть существующий или создать новый файл через меню программы. После открытия или создания становится доступной форма для показа и сокрытия столбцов сетки. Форму можно вызвать через меню «Вид» – «Показать/скрыть столбцы».

Для навигации, добавления и удаления записей в таблице служит элемент управления BindingNavigator, который становится доступным после открытия или создания файла записей о поликлиниках. Через это же меню возможна фильтрация таблицы по указанным в задании столбцам. Для применения фильтра необходимо использовать меню «Фильтр» и вписать критерий в TextBox, расположенный под название соответствующего столбца. Для удаления фильтра достаточно оставить TextBox пустым.

Для избегания ошибок, ячейки столбца «Address» возможно отредактировать только при помощи помощника, появляющегося при попытке изменить значения ячейки данного столбца.

3. СТРУКТУРА ПРИЛОЖЕНИЯ

3.1. Диаграмма классов



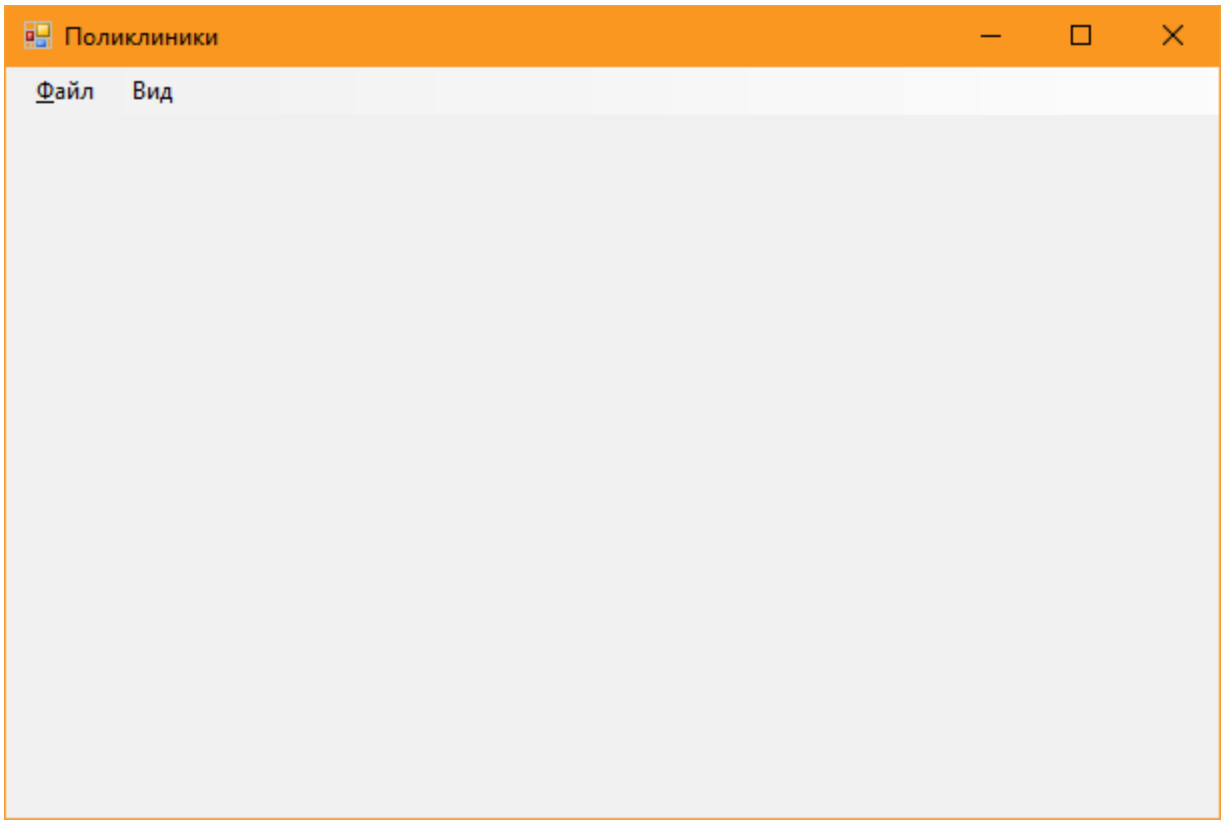
3.2. Описание полей и методов

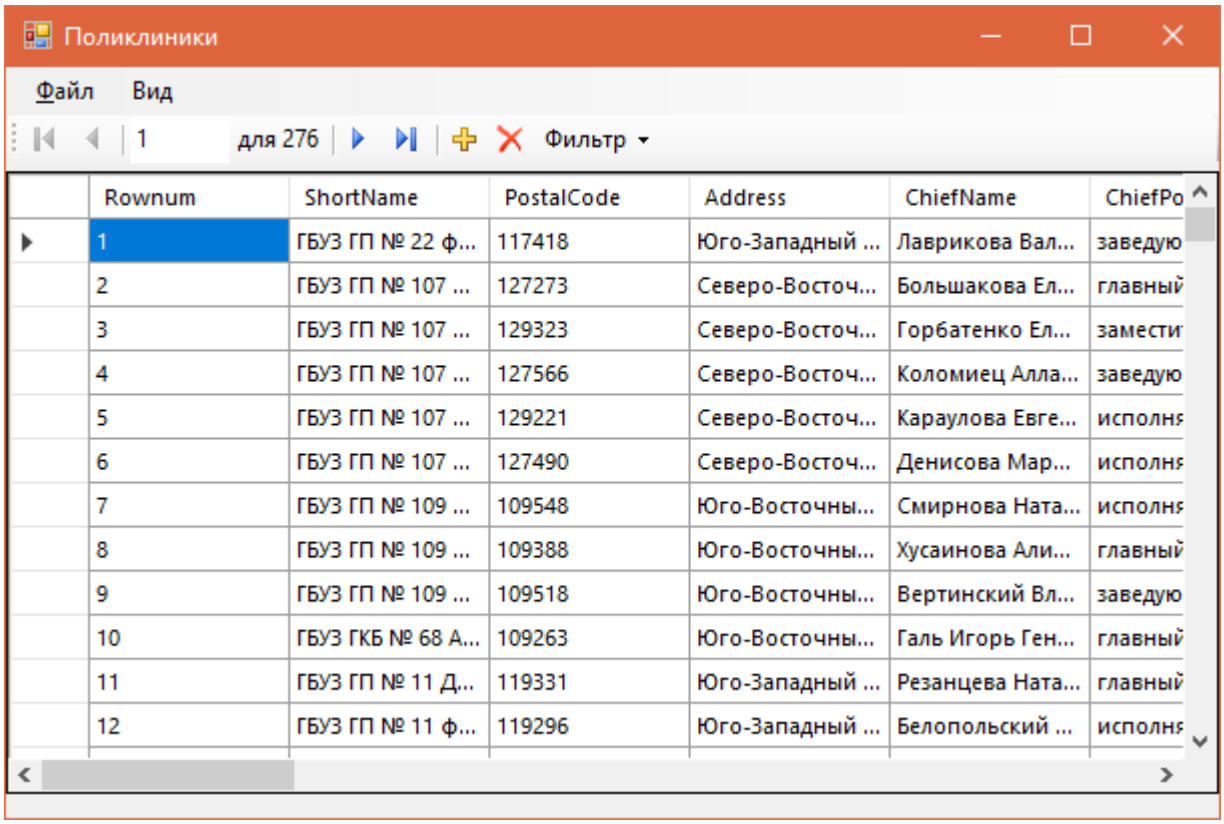
КЛАСС	ОПИСАНИЕ	
StartForm	МЕТОДЫ	*ToolStipMenuItem_Click – обеспечивают работоспособность кнопок и прочих элементов управления в форме. Например, OpenToolStipMenuItem_Click открывает диалоговое окно выбора файла и парсит выбранный файл, если он корректный.
	ПОЛЯ	Поля сгенерированы Visual Studio при работе в дизайнере форм.
Clinic Представляет запись о поликлинике	ПОЛЯ	Поля класса задаются полями файла Поликлиническая помощь взрослым версия 2.4 от 20.10.2014.csv, кроме полей, содержащих информацию о его адресе, представленном полем типа Address.
	МЕТОДЫ	Clinic – конструктор класса. По умолчанию (без аргументов) присваивает строковым полям значение пустой строки, а числовым – нули. Перегрузка Clinic принимает любой обьект, реализующий интерфейс IReadOnlyList<string> и на основе его элементов строит объект данного класса.
AddressBuilder	МЕТОДЫ	Address – конструктор класса.
		Distance – вычисляет евклидово расстояние до точки переданной по координатам в параметрах.
	ПОЛЯ	_parentForm – ссылка на вызвавшую форму.
		Остальные поля сгенерированы IDE Visual Studio 2017 автоматически при работе дизайнера форм.

4. РАСПРЕДЕЛЕНИЕ ИСХОДНОГО КОДА ПО ФАЙЛАМ ПРОЕКТА

AddressBuilder.cs	Форма для упрощения ввода адреса
Program.cs	Запуск программы
StartForm.cs	Содержит меню, сетку данных и навигатор по данным. Также в файле реализовано открытия и сохранение файла таблицы
SettingForm.cs	Форма для сокрытия и показа столбцов таблицы

5. КОНТРОЛЬНЫЙ ПРИМЕР И ОПИСАНИЕ РЕЗУЛЬТАТОВ





	Rownum	ShortName	PostalCode	Address	ChiefName	ChiefPo
▶	1	ГБУЗ ГП № 22 ф...	117418	Юго-Западный ...	Лаврикова Вал...	заведую
	2	ГБУЗ ГП № 107 ...	127273	Северо-Восточ...	Большакова Ел...	главный
	3	ГБУЗ ГП № 107 ...	129323	Северо-Восточ...	Горбатенко Ел...	замести
	4	ГБУЗ ГП № 107 ...	127566	Северо-Восточ...	Коломиец Алла...	заведую
	5	ГБУЗ ГП № 107 ...	129221	Северо-Восточ...	Караулова Евге...	исполня
	6	ГБУЗ ГП № 107 ...	127490	Северо-Восточ...	Денисова Мар...	исполня
	7	ГБУЗ ГП № 109 ...	109548	Юго-Восточны...	Смирнова Ната...	исполня
	8	ГБУЗ ГП № 109 ...	109388	Юго-Восточны...	Хусаинова Али...	главный
	9	ГБУЗ ГП № 109 ...	109518	Юго-Восточны...	Вертинский Вл...	заведую
	10	ГБУЗ ГКБ № 68 А...	109263	Юго-Восточны...	Галь Игорь Ген...	главный
	11	ГБУЗ ГП № 11 Д...	119331	Юго-Западный ...	Резанцева Ната...	главный
	12	ГБУЗ ГП № 11 ф...	119296	Юго-Западный ...	Белопольский ...	исполня

6. ТЕКСТ ПРОГРАММЫ

```
#region ToolStripMenuItem_Clicks

        /// <summary>
        /// Открывает файл и добавляет элементы в источник данных
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void OpenToolStripMenuItem_Click(object sender, EventArgs e)
        {
            using (var openDialog = new OpenFileDialog
            {
                Filter = Resources.StartForm_OpenToolStripMenuItem_Click_CSV_____csv
            })
            {
                if (openDialog.ShowDialog() != DialogResult.OK)
                    return;

                FileOpenedOrCreated(openDialog.FileName);

                using (var fieldParser = new
Microsoft.VisualBasic.FileIO.TextFieldParser(openDialog.FileName))
                {
                    if (fieldParser.TextFieldType ==
Microsoft.VisualBasic.FileIO.FieldType.Delimited)
                        fieldParser.SetDelimiters(";");
                    else
                    {
                        currentFile = null;

MessageBox.Show(Resources.StartForm_OpenToolStripMenuItem_Click_Invalid_File,
Resources.StartForm_OpenToolStripMenuItem_Click_Error_Reading_File, MessageBoxButtons.OK,
MessageBoxIcon.Error);
                        return;
                    }

                    if (fieldParser.EndOfData)
                    {
                        currentFile = null;

MessageBox.Show(Resources.StartForm_OpenToolStripMenuItem_Click_Empty_File,
Resources.StartForm_OpenToolStripMenuItem_Click_Error_Reading_File, MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                    else
                    {
                        // Пропустить строку названий столбцов
                        fieldParser.ReadFields();

                        while (!fieldParser.EndOfData)
                        {
                            try
                            {
                                clinicSource.Add(new Clinic(fieldParser.ReadFields()));
                            }
                            catch (Microsoft.VisualBasic.FileIO.MalformedLineException)
                            {
                                FileClosed();

                                if
(MessageBox.Show(Resources.StartForm_OpenToolStripMenuItem_Click_Invalid_File__Open_Anoth
er, Resources.StartForm_OpenToolStripMenuItem_Click_Error_Reading_File,
MessageBoxButtons.YesNo, MessageBoxIcon.Error) != DialogResult.Yes)
```

```
                                return;

                            OpenToolStripMenuItem_Click(this, EventArgs.Empty);
                            return;
                        }
                        catch (Exception ex)
                        {
                            FileClosed();

                            MessageBox.Show(ex.Message,
Resources.StartForm_OpenToolStripMenuItem_Click_Error_Reading_File, MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            return;
                        }
                    }
                }
            }
        }

        /// <summary>
        /// Закрывает форму
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void QuitToolStripMenuItem_Click(object sender, EventArgs e) => Close();

        /// <summary>
        /// Откывает форму показа столбцов
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ShowHideToolStripMenuItem_Click(object sender, EventArgs e) => new
SettingsForm(this).Show();

        private async void SaveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (currentFile != null)
                await Save(currentFile);
            else
            {
                SaveAsToolStripMenuItem_Click(sender, e);
            }
        }

        private async void SaveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            using (var saveDialog = new SaveFileDialog { Filter =
Resources.StartForm_SaveAsToolStripMenuItem_Click_Таблица_CSV_____csv })
            {
                if (saveDialog.ShowDialog() != DialogResult.OK)
                    return;

                await Save(saveDialog.FileName);
            }

        }

        private async void дописатьВToolStripMenuItem_Click(object sender, EventArgs e)
        {
            using (var save = new SaveFileDialog { Filter =
Resources.StartForm_OpenToolStripMenuItem_Click_CSV_____csv })
            {
                if (save.ShowDialog() == DialogResult.OK)
```

```
                await Save(currentFile = save.FileName, true, false);
            }

            OpenToolStripMenuItem_Click(this, EventArgs.Empty);
        }

        /// <summary>
        /// Очищает источник данных и прячет сетку и некоторые другие элементы формы
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CloseFileToolStripMenuItem_Click(object sender, EventArgs e)
        {
            clinicSource.Clear();
            FileClosed();
        }
        #endregion

        #region BindingNavigator_Clicks

        private void BindingNavigatorAddNewItem_Click(object sender, EventArgs e)
        {
            clinicSource.Add(new Clinic { Rownum = clinicSource.List.Count + 1,
PostalCode = 123456 });
        }

        private void BindingNavigatorDeleteItem_Click(object sender, EventArgs e)
        {
            try
            {
                if (ClinicView.CurrentRow != null && ClinicView.CurrentRow.Index != 0)
                    clinicSource.RemoveAt(ClinicView.CurrentRow.Index);
            }
            catch (InvalidOperationException)
            {
                MessageBox.Show("Невозможно удалить запись", "Ошибка выполнения",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка выполнения", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        #endregion

/// <summary>
        /// Подготавливает форму для отображения данных
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void FileOpenedOrCreated(object sender = null, EventArgs e = null)
        {
            ShowHideToolStripMenuItem.Enabled = true;
            fileOpened = true;
            bindingNavigator.Visible = true;
            ClinicView.Visible = true;
            currentFile = sender as string;
            clinicSource.Clear();
        }

        /// <summary>
        /// Подготавливает форму для очистки данных.
```

```
        /// </summary>
        private void FileClosed()
        {
            currentFile = null;
            ShowHideToolStripMenuItem.Enabled = false;
            fileOpened = false;
            bindingNavigator.Visible = false;
            ClinicView.Visible = false;
        }

        /// <summary>
        /// Обеспечивает сохранение в файл тремя способами.
        /// </summary>
        /// <param name="path">Полный путь до файла для записи.</param>
        /// <param name="append">true для добавления данных в файл, false для
перезаписи.</param>
        /// <param name="writeHeaders"></param>
        /// <returns>Задача, представляющая асинхронную операцию записи в файл.</returns>
        private async System.Threading.Tasks.Task Save(string path, bool append = false,
bool writeHeaders = true)
        {
            if (fileOpened)
            {
                Cursor.Current = Cursors.WaitCursor;
                using (var writer = new System.IO.StreamWriter(path, append,
System.Text.Encoding.Unicode))
                {
                    if (writeHeaders)
                    {
                        foreach (var header in Headers)
                            await writer.WriteAsync($"{header};");

                        await writer.WriteLineAsync();
                    }

                    foreach (var item in clinicSource)
                    {
                        await writer.WriteLineAsync(item.ToString());
                    }
                }
                Cursor.Current = Cursors.Default;
            }
            else
            {
                if (MessageBox.Show(Resources.StartForm_Save_File_Not_Opened__Open_,
Resources.StartForm_Save_Saving_Error, MessageBoxButtons.YesNo, MessageBoxIcon.Error) !=
DialogResult.Yes)
                    return;

                OpenToolStripMenuItem_Click(this, EventArgs.Empty);
            }
        }

        private void ClinicView_CellValuePushed(object sender,
DataGridViewCellValueEventArgs e)
        {
            ((Clinic)clinicSource.Current)[e.ColumnIndex] = e.Value;
        }



        private void ClinicView_DataError(object sender, DataGridViewDataErrorEventArgs
e)
        {
```

```
            // If the data source raises an exception when a cell value is
            // commited, display an error message.
            if (e.Exception != null)
            {
                MessageBox.Show(e.Exception.Message);
            }
        }

        private void FindClosestToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (clinicSource.Current is Clinic current)
            {
                var min = double.MaxValue;
                var index = 0;
                for (var i = 0; i < clinicSource.Count; i++)
                {
                    var clinic = (Clinic)clinicSource[i];
                    var distance = current.Address.Distance(clinic.Address.PointX,
clinic.Address.PointY);
                    if (clinic != current && distance < min)
                    {
                        min = distance;
                        index = i;
                    }
                }

                ClinicView.FirstDisplayedScrollingRowIndex = index;
                ClinicView.Rows[index].Selected = true;
            }
        }

        private void clinicView_CellBeginEdit(object sender,
DataGridViewCellCancelEventArgs e)
        {
            if (ClinicView.Columns[e.ColumnIndex].HeaderText == "Address")
            {
                ClinicView.CancelEdit();
                new AddressBuilder(this).ShowDialog();
            }
        }

        private void toolStripTextBox1_KeyDown(object sender, KeyEventArgs e)
        {

            if (e.KeyCode == Keys.Enter)
            {
                for (var i = 1; i < ClinicView.RowCount; i++)
                {
                    if (((string)ClinicView.Rows[i].Cells[15].Value).Contains(
                        ((ToolStripTextBox)sender).Text))
                        ClinicView.Rows[i].Visible = true;
                    else
                        ClinicView.Rows[i].Visible = false;
                }
            }
        }

        private void toolStripTextBox2_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                for (var i = 1; i < ClinicView.RowCount; i++)
                {
                    if (((Address)ClinicView.Rows[i].Cells[3].Value).District.Contains(
```

```
                        ((ToolStripTextBox)sender).Text))
                        ClinicView.Rows[i].Visible = true;
                    else
                        ClinicView.Rows[i].Visible = false;
                }
            }
        }
    }
public class Address
    {
        /// <summary>
        /// Административный округ
        /// </summary>
        public string AdmArea { get; }

        /// <summary>
        /// Район
        /// </summary>
        public string District { get; }

        /// <summary>
        /// Адрес
        /// </summary>
        public string AddressString { get; }

        /// <summary>
        /// Координата X
        /// </summary>
        public double PointX { get; }

        /// <summary>
        /// Координата Y
        /// </summary>
        public double PointY { get; }

        public Address()
        {
            AdmArea = District = AddressString = string.Empty;
        }

        public Address(string admArea, string district, string addressString, double
pointX, double pointY)
        {
            AdmArea = admArea;
            District = district;
            AddressString = addressString;
            PointX = pointX;
            PointY = pointY;
        }

        /// <summary>
        /// Вычисление расстояния до места по координатам
        /// </summary>
        /// <param name="x">Координата x</param>
        /// <param name="y">Координата y</param>
        /// <returns></returns>
        public double Distance(double x, double y)
        {
            double dx = PointX - x, dy = PointY - y;
            return Math.Sqrt(dx * dx + dy * dy);
        }

        public override string ToString() =>
$"{AdmArea},{District},{AddressString},{PointX},{PointY}";
```

```
    }
public partial class AddressBuilder : Form
    {
        private readonly StartForm _paretForm;

        private AddressBuilder()
        {
            InitializeComponent();
        }

        public AddressBuilder(Form form) : this()
        {
            _paretForm = form as StartForm;
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            if (double.TryParse(pointXtextBox.Text, out var x) &&
double.TryParse(pointYtextBox.Text, out var y))
            {
                ((Clinic) _paretForm.ClinicSource.Current).Address = new
Address(admTextBox.Text, districtTextBox.Text, addressTextBox.Text, x, y);
                DialogResult = DialogResult.OK;
            }
            else
            {
                MessageBox.Show("Введенные координаты не являются числом. Попробуйте еще
раз!");
                pointXtextBox.Text = pointYtextBox.Text = string.Empty;
            }
        }

        private void Button2_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.Abort;
        }
    }
public class Clinic
    {
        /// <summary>
        /// Формат числа с плавающей точкой в файле
        /// </summary>
        private static readonly NumberFormatInfo DecimalStyle = new NumberFormatInfo {
NumberDecimalSeparator = "." };

        public Clinic()
        {
            ShortName =
                ChiefName =
                    ChiefPosition =
                        ChiefGender =
                            ChiefPhone =
                                PublicPhone =
                                    Fax =
                                        Email =
                                            CloseFlag =
                                                CloseReason =
                                                    CloseDate =
                                                        ReopenDate =
                                                            PaidServicesInfo =
                                                                FreeServicesInfo =
                                                                    WorkingHours =
```

```
ClarificationOfWorkingHours =

Specialization =

BeneficialDrugPrescriptions =

ExtraInfo =

AddressUnom = string.Empty;
            Address = new Address();
        }

        public Clinic(IReadOnlyList<string> row)
        {
            if (row.Count == 29)
            {
                Rownum = int.Parse(row[0]);
                ShortName = row[1];
                PostalCode = int.Parse(row[4]);
                Address = new Address(row[2], row[3], row[5], double.Parse(row[25],
DecimalStyle), double.Parse(row[26], DecimalStyle));
                ChiefName = row[6];
                ChiefPosition = row[7];
                ChiefGender = row[8];
                ChiefPhone = row[9];
                PublicPhone = row[10];
                Fax = row[11];
                Email = row[12];
                CloseFlag = row[13];
                CloseReason = row[14];
                CloseDate = row[15];
                ReopenDate = row[16];
                PaidServicesInfo = row[17];
                FreeServicesInfo = row[18];
                WorkingHours = row[19];
                ClarificationOfWorkingHours = row[20];
                Specialization = row[21];
                BeneficialDrugPrescriptions = row[22];
                ExtraInfo = row[23];
                AddressUnom = row[24];
                Globalid = int.Parse(row[27]);
            }
            else
            {
                throw new Microsoft.VisualBasic.FileIO.MalformedLineException();
            }
        }

        public int Rownum { get; set; }
        public string ShortName { get; set; }
        public int PostalCode { get; set; }
        public Address Address { get; set; }
        public string ChiefName { get; set; }
        public string ChiefPosition { get; set; }
        public string ChiefGender { get; set; }
        public string ChiefPhone { get; set; }
        public string PublicPhone { get; set; }
        public string Fax { get; set; }
        public string Email { get; set; }
        public string CloseFlag { get; set; }
        public string CloseReason { get; set; }
        public string CloseDate { get; set; }
        public string ReopenDate { get; set; }
```

```
        public string PaidServicesInfo { get; set; }
        public string FreeServicesInfo { get; set; }
        public string WorkingHours { get; set; }
        public string ClarificationOfWorkingHours { get; set; }
        public string Specialization { get; set; }
        public string BeneficialDrugPrescriptions { get; set; }
        public string ExtraInfo { get; set; }
        public string AddressUnom { get; set; }
        public int Globalid { get; set; }

        public object this[int index]
        {
            set
            {
                switch (index)
                {
                    case 0:
                        Rownum = (int)value;
                        break;
                    case 1:
                        ShortName = (string)value;
                        break;
                    case 2:
                        PostalCode = (int)value;
                        break;
                    case 3:
                        // Не дожно никогда запускаться
                        var t = ((Address)value).ToString().Split(',');
                        Address = new Address(t[0], t[1], t[2], double.Parse(t[3]),
double.Parse(t[3]));
                        break;
                    case 4:
                        ChiefName = (string)value;
                        break;
                    case 5:
                        ChiefPosition = (string)value;
                        break;
                    case 6:
                        ChiefGender = (string)value;
                        break;
                    case 7:
                        ChiefPhone = (string)value;
                        break;
                    case 8:
                        PublicPhone = (string)value;
                        break;
                    case 9:
                        Fax = (string)value;
                        break;
                    case 10:
                        Email = (string)value;
                        break;
                    case 11:
                        CloseFlag = (string)value;
                        break;
                    case 12:
                        CloseReason = (string)value;
                        break;
                    case 13:
                        CloseDate = (string)value;
                        break;
                    case 14:
                        ReopenDate = (string)value;
                        break;
```

```
                    case 15:
                        PaidServicesInfo = (string)value;
                        break;
                    case 16:
                        FreeServicesInfo = (string)value;
                        break;
                    case 17:
                        WorkingHours = (string)value;
                        break;
                    case 18:
                        ClarificationOfWorkingHours = (string)value;
                        break;
                    case 19:
                        Specialization = (string)value;
                        break;
                    case 20:
                        BeneficialDrugPrescriptions = (string)value;
                        break;
                    case 21:
                        ExtraInfo = (string)value;
                        break;
                    case 22:
                        ShortName = (string)value;
                        break;
                    case 23:
                        Globalid = (int)value;
                        break;
                    case 24:
                        AddressUnom = (string) value;
                        break;
                    default:
                        // Не дожно никогда запускаться
                        break;
                }
            }
        }

        public override string ToString()
        {
            return $"{Rownum};\"{ShortName ??
string.Empty}\";\"{Address.AdmArea}\";\"{Address.District}\";{PostalCode};\"{Address.Addr
essString}\";\"{ChiefName}\";\"{ChiefPosition}\";\"{ChiefGender}\";\"{ChiefPhone}\";\"{Pu
blicPhone}\";\"{Fax}\";\"{Email}\";\"{CloseFlag}\";\"{CloseReason}\";\"{CloseDate}\";\"{R
eopenDate}\";\"{PaidServicesInfo}\";\"{FreeServicesInfo}\";\"{WorkingHours}\";\"{Clarific
ationOfWorkingHours}\";\"{Specialization}\";\"{BeneficialDrugPrescriptions}\";\"{ExtraInf
o}\";\"{AddressUnom}\";{Address.PointX};{Address.PointY};{Globalid};";
        }
    }
public partial class SettingsForm : Form
    {
        private SettingsForm()
        {
            InitializeComponent();
        }

        private readonly StartForm _parentForm;

        public SettingsForm(StartForm parentForm) : this() => _parentForm = parentForm;

        private void SettingsForm_Load(object sender, EventArgs e)
        {
            for (var i = 0; i < _parentForm.ClinicView.ColumnCount; i++)
                ShowContent.Items.Add(item: _parentForm.ClinicView.Columns[i].HeaderText,
isChecked: _parentForm.ClinicView.Columns[i].Visible);
```

```
        }

        private void ShowContent_ItemCheck(object sender, ItemCheckEventArgs e)
        {
            if (sender is CheckedListBox checkedList && checkedList.SelectedItem != null
&& _parentForm?
                    .ClinicView?
                    .Columns != null)
                // ReSharper disable once PossibleNullReferenceException
                try
                {
                    _parentForm.ClinicView.Columns[checkedList.SelectedIndex].Visible =
                    e.NewValue == CheckState.Checked;
                }
                catch (NullReferenceException)
                {
                    MessageBox.Show("Открыт ли файл?");
                }
        }
    }
```