Учреждение высшего образования

«Университет управления «ТИСБИ»

Факультет информационных технологий

Кафедра информационных технологий

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

на тему: «Разработка объектной программы для задачи «Юридическая фирма»

Выполнил: студент гр. ПИ-022

Фамилия И.О.

Проверил: ст. преподаватель

Якунина Е.А.

Казань 2022

Оглавление

[Постановка задачи 3](#_Toc105331406)

[1. Описание используемых структур данных с алгоритмами выполнения основных операций 5](#_Toc105331407)

[1.1 Двунаправленный список. 5](#_Toc105331408)

[1.2 Кольцевая очередь на основе обычного массива 6](#_Toc105331409)

[2. Краткие сведения об объектном подходе 8](#_Toc105331410)

[3. Формализованное описание разработанных классов 14](#_Toc105331411)

[4. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков 19](#_Toc105331412)

[5. Описание структуры проекта в соответствии с использованным инструментом разработки 24](#_Toc105331413)

[Список литературы 26](#_Toc105331414)

[Листинг программы 28](#_Toc105331415)

Постановка задачи

**Цели работы:** отработка навыков курсов «программная инженерия» и «структуры и алгоритмы обработки данных».

**Постановка задачи:** разработать объектную программу для хранения и обработки данных об услугах, оказываемых некоторой юридической фирмой. Учет организуется за счет ведения списка юристов фирмы с указанием фамилии и номера мобильного телефона. Для каждого юриста создается отдельный список его клиентов с указанием фамилии клиента, вида услуги и ее стоимости.

Разработка программы включает в себя:

* определение необходимых объектов и способов их взаимодействия;
* формальное описание объектов в виде классов;
* программную реализацию всех необходимых методов, включая корректировку вкладов и подсчет текущего суммарного объема всех вкладов;
* всестороннее тестирование методов с помощью консольного (при разработке) и оконного (в окончательном варианте) приложения.

Для объединения юристов используется структура данных в виде адресного стека. Для объединения клиентов используется кольцевая очередь на основе обычного массива.

Разработка выполняется с учетом следующих требований:

1. имена классов, свойств и методов должны носить содержательный смысл и соответствовать информационной задаче
2. обязательное соблюдение принципа инкапсуляции – использование в классах только закрытых свойств и реализация необходимого набора методов доступа
3. наличие двух методов для сохранения всей объектной структуры во внешнем файле с обратной загрузкой, при этом стандартные механизмы сериализации разрешается использовать только как дополнение к самостоятельно реализованным методам
4. тестовое оконное приложение должно обладать удобным пользовательским интерфейсом с контролем вводимых данных и отображением текущего состояния объектной структуры с помощью списковых или табличных компонентов
5. стандартные контейнеры/коллекции (включая обобщенные классы) разрешается использовать только как дополнение к самостоятельно разработанным классам
6. в качестве языка разработки разрешается использовать Java, С, C++, Object/Free Pascal и соответствующие инструменты быстрой разработки приложений.
7. **Описание используемых структур данных с алгоритмами выполнения основных операций**

1.1 Двунаправленный список.

Для ускорения многих операций целесообразно применять переходы между элементами списка в обоих направлениях. Это реализуется с помощью двунаправленных списков, которые являются сложной динамической структурой.

Двунаправленный (двусвязный) список – это структура данных, состоящая из последовательности элементов, каждый из которых содержит информационную часть и два указателя на соседние элементы ( рис. 29.4). При этом два соседних элемента должны содержать взаимные ссылки друг на друга.

В таком списке каждый элемент (кроме первого и последнего) связан с предыдущим и следующим за ним элементами. Каждый элемент двунаправленного списка имеет два поля с указателями: одно поле содержит ссылку на следующий элемент, другое поле – ссылку на предыдущий элемент и третье поле – информационное. Наличие ссылок на следующее звено и на предыдущее позволяет двигаться по списку от каждого звена в любом направлении: от звена к концу списка или от звена к началу списка, поэтому такой список называют двунаправленным.

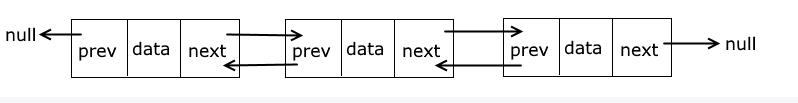


Рис. 1 Двусвязный список

Основные операции, выполняемые над двунаправленным списком, те же, что и для однонаправленного списка. Так как двунаправленный список более гибкий, чем однонаправленный, то при включении элемента в список, нужно использовать указатель как на элемент, за которым происходит включение, так и указатель на элемент, перед которым происходит включение. При исключении элемента из списка нужно использовать как указатель на сам исключаемый элемент, так и указатели на предшествующий или следующий за исключаемым элементы. Но так как элемент двунаправленного списка имеет два указателя, то при выполнении операций включения/исключения элемента надо изменять больше связей, чем в однонаправленном списке.

Рассмотрим основные операции, осуществляемые с двунаправленными списками, такие как:

создание списка;

печать (просмотр) списка;

вставка элемента в список;

удаление элемента из списка;

поиск элемента в списке;

проверка пустоты списка;

удаление списка.

Особое внимание следует обратить на то, что в отличие от однонаправленного списка здесь нет необходимости обеспечивать позиционирование какого-либо указателя именно на первый элемент списка, так как благодаря двум указателям в элементах можно получить доступ к любому элементу списка из любого другого элемента, осуществляя переходы в прямом или обратном направлении. Однако по правилам хорошего тона программирования указатель желательно ставить на заголовок списка.

1.2 Кольцевая очередь на основе обычного массива

Кольцевая очередь работает по принципу FIFO (First – In – First – Out): «первым пришел - первым вышел». Отличие между кольцевой очередью и обычной очередью заключается в способе выхода из очереди первого элемента. В кольцевой очереди первый элемент перемещается в конец очереди. Примерами кольцевой очереди могут быть движение троллейбусов по кольцевому маршруту, круговорот воды в природе и т.д.

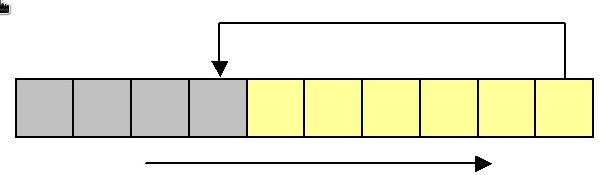


Рис. 2: Кольцевая очередь. Перемещение первого элемента в конец очереди

Кольцевая очередь работает по следующему принципу.

1. Сначала в очередь добавляются элементы до тех пор, пока не будет достигнут какой-то максимум. Элементы добавляются в конец очереди.
2. После достижения определенного (максимального) количества элементов происходит циклическое изменение первого элемента по следующему принципу: первый элемент очереди перемещается в конец очереди, а остальные элементы этой очереди сдвигаются на одну позицию впе ред. Таким образом второй элемент очереди становится первым, третий элемент становится вторым и т.д.

Базовый перечень выполняемых операций для кольцевой очереди следую щий:

* добавить новый элемент в очередь;
* проверка, есть ли очередь пустой;
* проверка, есть ли очередь полной;
* очистка очереди (удаление всех элементов очереди);
* вытянуть первый элемент из очереди и поместить в конец очереди.

1. Краткие сведения об объектном подходе

Объектно-ориентированное программирование – одно из важнейших методологий разработки, которая основано на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Основные принципы ООП:

Инкапсуляция

Одним из самых важных факторов при проектировании компонентов приложения является сокрытие внутренних данных компонента и деталей его реализации от других компонентов приложения и предоставление набора методов для взаимодействия с ним (API). Этот принцип является одним из четырёх фундаментальных принципов ООП и называется инкапсуляцией.

Правильная инкапсуляция важна по многим причинам:

1. Она способствует переиспользованию компонентов: поскольку в этом случае компоненты взаимодействуют друг с другом только посредством их API и безразличны к изменениям внутренней структуры, они могут использоваться в более широком контексте.

2. Инкапсуляция ускоряет процесс разработки: слабо связанные друг с другом компоненты (то есть компоненты, чей код как можно меньше обращается или использует код других компонентов) могут разрабатываться, тестироваться и дополняться независимо.

3. Правильно инкапсулированные компоненты более легки для понимания и процесса отладки, что упрощает поддержку приложения.

В языке Java инкапсуляция реализована с помощью системы классов, которые позволяют собрать информацию об объекте в одном месте; пакетов, которые группируют классы по какому-либо критерию, и модификаторов доступа, которыми можно пометить весь класс или его поле или метод.

Всего модификаторов доступа четыре:

public – полный доступ к сущности (полю или методу класса) из любого пакета;

protected – доступ к сущности только для классов своего пакета и наследников класса;

неявный модификатор по умолчанию (при отсутствии трёх явных) – доступ к сущности только для классов своего пакета;

private – доступ только внутри класса, в котором объявлена сущность.

Для достижения правильной инкапсуляции также необходимо предоставить корректный API для работы с компонентом. Например, в сеттер для переменной можно включить логику по проверке передаваемых значений либо не предоставлять сеттеры в классе вовсе, если класс должен быть доступен только для чтения.

Пример корректной инкапсуляции класса:

В примере выше значение переменной name задаётся при создании объекта и не может быть изменено извне, так как сеттер для переменной отсутствует. В сеттере для переменной age реализована проверка на корректность передаваемого параметра и выброс исключения при неверном значении.

Наследование

Наследование является одним из важнейших принципов объектно-ориентированного программирования, поскольку оно позволяет создавать иерархические структуры объектов. Используя наследование, можно создать общий класс, который будет определять характеристики и поведение, свойственные какому-то набору связанных объектов. В дальнейшем этот класс может быть унаследован другими, более частными классами, каждый из которых будет добавлять уникальные, свойственные только ему характеристики и дополнять или изменять поведение базового класса. В терминах Java такой общий класс называется суперклассом (superclass), или базовым классом (base class), или классом-родителем (parent class), а класс, его наследующий, - подклассом (subclass), или дочерним классом (child class), или классом-потомком (derived class).

Наследование реализует отношение «является» (“is-a”) между суперклассом и подклассом. Пусть, например, классы Employee и Manager представляют собой абстракцию понятий «Сотрудник» и «Менеджер». Каждый менеджер также является сотрудником компании, в которой он работает, следовательно, класс Manager находится в отношении “is-a” с классом Employee. Таким образом, с точки зрения наследования, при выстраивании иерархии классов класс Employee будет являться суперклассом, а класс Manager – дочерним классом. При этом класс, который является наследником какого-либо класса, может быть суперклассом для одного или нескольких других классов. Также в отличие, например, от C++, в Java в строгом смысле отсутствует множественное наследование, то есть любой класс может иметь не более одного класса-родителя. А все классы, суперкласс у которых явно не указан, по умолчанию наследуются от класса Object.

Класс Employee в примере выше является суперклассом не потому, что он главнее класса Manager или содержит больше функциональности. На самом деле верно обратное: функциональность подклассов не ýже, а зачастую существенно шире, чем функциональность их классов-родителей. Приставки «супер-» и «под-» пришли в Java из математики: множество всех менеджеров содержится во множестве всех сотрудников и, таким образом, является подмножеством множества сотрудников.

Для того чтобы унаследовать какой-либо класс в Java, используется ключевое слово extends:

В примере выше класс Employee является базовым классом для класса Manager, а класс Manager – подклассом класса Employee. Класс Employee абстрагирует базовые характеристики для всех сотрудников компании – имя, фамилию, размер оклада и дату приёма на работу, а класс Manager дополняет эти характеристики процентом премии для менеджеров и меняет поведение метода getSalary() базового класса, используя полиморфизм.

Полиморфизм

При рассмотрении полиморфизма необходимо помнить, что этот принцип неразрывно связан с другим принципом ООП – наследованием, которое помогает реализовать полиморфизм.

Возьмем для примера абстрактный класс «Автомобиль», который наследуют два конкретных класса – «Спортивный автомобиль» и «Грузовой автомобиль».

И спортивные, и грузовые автомобили будут обладать общими характеристиками и будут иметь возможность выполнять общие для всех автомобилей действия, которые указаны в абстрактном классе-родителе, но конкретная реализация этих действий может быть разной.

Например, общее для всех автомобилей действие «завестись» у спортивного автомобиля может быть реализовано путем нажатия кнопки, а у грузового - с помощью ключа. Один результат – разные решения. В этом и состоит полиморфизм.

Более строго, полиморфизм - один из принципов ООП, позволяющий вызовом переопределённого метода через переменную класса-родителя получить поведение, которое будет соответствовать реальному классу-потомку, на который ссылается эта переменная.

Код выше представляет пример полиморфизма. Сначала переменной родительского класса Vehicle присваивается объект дочернего класса SportCar. При вызове метода start() на консоль будет выведено: "Starting my fancy sport car!"

При дальнейшем присвоении этой же переменной объекта дочернего класса Truck и вызове того же метода start() на консоль будет выведено: "Starting my heavy truck!"

Абстракция

Относительно недавно в качестве самостоятельного четвёртого принципа начали выделять абстракцию.

Одно из определений слова «абстракция», которые можно встретить в современных словарях:

Абстракция (от лат. abstractio — выделение, отвлечение или отделение) — теоретический прием исследования, позволяющий отвлечься от некоторых несущественных в определенном отношении свойств изучаемых явлений и выделить свойства существенные и определяющие.

Все языки программирования предоставляют их пользователю определённые абстракции. Так, языки семейства ассемблер являются в своём роде абстракцией соответствующих микропроцессоров, поскольку позволяют отвлечься от деталей их реализации и общаться с ними через определённый набор более высокоуровневых инструкций. Императивные языки программирования, последовавшие за ассемблером, например Basic, Fortran, C, являются более высоким уровнем абстракции над ассемблерными языками – они дают возможность использовать более привычные человеку синтаксические конструкции за счёт приближения синтаксиса к естественным языкам.

Объектно-ориентированные языки, такие как Java, выводят разработку на ещё более высокий уровень абстракции: объекты в ООП по своей сути представляют собой модели понятий окружающего мира, таких как Работник, Сервер, Запись в дневнике, и выделяют только те свойства этого понятия, которые необходимы в конкретном случае для решения конкретной проблемы.

Например, класс Student в приложении учёта студентов университета, кроме общих полей, таких как имя, фамилия, дата рождения и т.д., будет содержать поля, отражающие информацию о номере зачётной книжки, статусе студента (действующий, академический отпуск, отчислен), факультете, номере его группы, оценках за семестры и т.д. Но для того же класса Student в приложении учёта студентов в тренинг-центре EPAM такая информация будет неактуальна: класс будет содержать поля, отражающие учебный проект, на который был распределён студент, уровень его английского языка по результатам последнего тестирования, количество посещаемых мероприятий и т.д.

В этом и состоит абстракция: фокусировка разработчика на конкретных свойствах объекта зависит от тех задач, которые призван решать объект. Следствием такого подхода является то, что, если в императивных языках разработчику необходимо думать в терминах компьютерной логики, в объектно-ориентированных языках разработчик думает в терминах проблемной области, в которой он разрабатывает приложения.

3. Формализованное описание разработанных классов

Базовый класс необходимый во всех остальных классах. Содержит данные о клиентах юриста: фамилию и стоимость потраченную на услугу.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class Client

**{**

**private** string surname**,** service**;**//фамилия клиента, оказанные услуги

**private** int cost**;**//стоимость услуг

**public** Client**(**string \_surname**,** string \_service**,** int \_cost**)**

**{**

**this.**surname **=** \_surname**;**

**this.**service **=** \_service**;**

**this.**cost **=** \_cost**;**

**}**

**public** string Surname//методы доступа для фамилии

**{** **get** **{** **return** surname**;** **}** **set** **{** surname **=** **value;** **}** **}**

**public** int Cost //методы доступа для стоимости

**{** **get** **{** **return** cost**;** **}** **set** **{** cost **=** **value;** **}** **}**

**public** string Service //методы доступа для оказанной услуги

**{** **get** **{** **return** service**;** **}** **set** **{** service **=** **value;** **}** **}**

**}**

**}**

Структура данных адресный список реализует контейнер, который хранит всех клиненто у юриста. Необходим для создания класса юриста.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class List

**{**

**private** Client client**;**//важные данные

**private** List last**,** first**,** second**,**pred**;**//переменные-ссылки

**public** List **(**Client \_client**)**//конструктор

**{**

**this.**client **=** \_client**;**

**this.**last **=** **null;**

**this.**first **=** **null;**

**this.**second **=** **null;**

**}**

**public** Client cClient **{** **get** **{** **return** client**;** **}set** **{** **this.**client **=** **value;** **}** **}**

**public** List Second//гет и сет для следующего элемента

**{**

**get** **{** **return** second**;** **}**

**set** **{** second **=** **value;** **}**

**}**

**public** List Last//гет и сет для последнего элемента

**{**

**get** **{** **return** last**;** **}**

**set** **{** last **=** **value;** **}**

**}**

**public** List First//гет и сет для первого элемента

**{**

**get** **{** **return** first**;** **}**

**set** **{** first **=** **value;** **}**

**}**

**public** List Pred

**{**

**get** **{** **return** pred**;** **}**

**set** **{** **this.**pred **=** **value;** **}**

**}**

**}**

**}**

Класс юрист содержит в себе сведенья о юристе с его фамилией и номером, а так же список клиентов которые его посещали.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class Urist

**{**

**private** string surnameUrist**;**//фамилия юриста

**private** int sum**,** count**;**//сумма полученных денег этим юристом

**private** List head**,** pTemp**,**pHelp**;**//заголовок списка и вспомогательная переменная

**public** Urist **(**string \_surnameUrist**)**//конструктор класса

**{**

**this.**surnameUrist **=** \_surnameUrist**;**

head **=** **new** List**(null);**

head**.**First **=** **null;**

head**.**Last **=** **null;**

**}**

**public** int Count **{** **get** **{** **return** count**;** **}** **}**

**public** List Head **{** **get** **{** **return** head**.**First**;** **}** **}**

**public** string SurnameUrist **{** **get** **{** **return** surnameUrist**;** **}** **set** **{** **this.**surnameUrist **=** **value;** **}** **}**

**public** void AddClient**(**string \_surname**,** string \_service**,** int \_cost**)**

**{**

Client dobav **=** **new** Client**(**\_surname**,** \_service**,** \_cost**);**

List dobavs **=** **new** List**(**dobav**);**

count**++;**

**if** **(**head**.**First **==** **null)**

**{**

head**.**First **=** dobavs**;**

head**.**First**.**Pred **=** **null;**

head**.**First**.**Second **=** head**.**Last**;**

head**.**Last **=** dobavs**;**

**}**

**else**

**{**

pHelp **=** head**.**Last**;**

head**.**Last **=** dobavs**;**

pHelp**.**Second **=** head**.**Last**;**

head**.**Last**.**Pred **=** pHelp**;**

**}**

**}**

**public** void DellClient**(**string \_surname**)**

**{**

pHelp **=** head**.**First**;**

**while** **(**pHelp **!=** **null)**

**{**

List pTemp**;**

**if** **(**pHelp **==** head**.**First **&&** pHelp**.**cClient**.**Surname**.**Equals**(**\_surname**))**

**{**

**if** **(**pHelp**.**Second **==** **null)**

**{**

head**.**First **=** **null;**

**}**

**else**

**{**

pTemp **=** head**.**First**.**Second**;**

pHelp**.**Second**.**Pred **=** **null;**

pHelp**.**Second **=** **null;**

head**.**First **=** pTemp**;**

**}**

**}**

**else** **if** **(**pHelp**.**cClient**.**Surname**.**Equals**(**\_surname**))**

**{**

**if** **(**pHelp**.**Pred **!=** **null** **&&** pHelp**.**Second **!=** **null)**

**{**

pHelp**.**Pred**.**Second **=** pHelp**.**Second**;**

pHelp**.**Second**.**Pred **=** pHelp**.**Pred**;**

**}**

**else**

**{**

head**.**Last **=** head**.**Last**.**Pred**;**

**}**

**}**

pHelp **=** pHelp**.**Second**;**

**}**

**}**

**public** Client SearchCl**(**string \_surname**)**

**{**

List vspom **=** head**.**First**;**

**while(**vspom **!=** **null)**

**{**

**if** **(**vspom**.**cClient**.**Surname**.**Equals**(**\_surname**))**

**{**

**return** vspom**.**cClient**;**

**}**

**else** **{** vspom **=** vspom**.**Second**;** **}**

**}**

**return** **null;**

**}**

**public** int Sum**()**

**{**

sum **=** 0**;**

List vspom **=** head**.**First**;**

**while(**vspom **!=** **null)**

**{**

sum **+=** vspom**.**cClient**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

**return** sum**;**

**}**

**public** string GetInfo**()**

**{**

string info **=** ""**;**

List vspom **=** head**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

info **+=** vspom**.**cClient**.**Surname**;**

info **+=** vspom**.**cClient**.**Cost**;**

**}**

info **+=** Sum**();**

**return** info**;**

**}**

**public** string getInfoFor**()**

**{**

string info **=** Convert**.**ToString**(**count**);**

List vspom **=** head**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

info **+=** $"\n{vspom.cClient.Surname}"**;**

info **+=** $"\n{vspom.cClient.Service}"**;**

info **+=** $"\n{vspom.cClient.Cost}"**;**

vspom **=** vspom**.**Second**;**

**}**

**return** info**;**

**}**

**}**

**}**

Класс фирма представляет собой контейнер в виде кольцевой очереди для работы с классом юрист.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class Firm

**{**

**private** string nameFirm**;**//название фирмы

**private** Urist**[]** ochered**;**//массив обьектов класса юрист

**private** int size**,** first**,** last**,** count**,** sum**;**//необходимые переменные для

//подсчета суммы, размера массива и координации в очереди

**public** Firm**(**string \_nameFirm**,** int \_size**)**//констуктор

**{**

**this.**nameFirm **=** \_nameFirm**;**

**this.**size **=** \_size**;this.**last **=** 0**;**

**this.**first **=** 0**;** **this.**count **=** 0**;**

ochered **=** **new** Urist**[**size**];**

**}**

**public** int Count **{** **get** **{** **return** count**;** **}** **}**

**public** void AddUrist**(**string \_surnameUrist**)**//метод добавления

**{**

**if(**count **<=** size**)**

**{**

Urist dobav **=** **new** Urist**(**\_surnameUrist**);**

ochered**[**last**]** **=** dobav**;**

last**++;** count**++;**

**}**

**}**

**public** void DellUrist**()**//метод удаления

**{**

**if** **(**count **>** 0**)**

**{**

ochered**[**first**]** **=** **null;**

count**--;**

**for** **(**int i **=** **(**first **+** 1**);** i **<** last**;** i**++)**

**{**

ochered**[**i **-** 1**]** **=** ochered**[**i**];**

**}**

last**--;**

ochered**[**last**]** **=** **null;**

**}**

**}**

**public** Urist SearchUrist**(**string \_surnameUrist**)**//поиск для дальнейшего добавления и удаления клиентов

**{**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

**if** **(**ochered**[**i**]** **!=** **null** **&&** ochered**[**i**].**SurnameUrist**.**Equals**(**\_surnameUrist**))**

**{**

**return** ochered**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** int GetSumm**()**//сумма полученных денег

**{**

sum **=** 0**;**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

sum **+=** ochered**[**i**].**Sum**();**

**}**

**return** sum**;**

**}**

**public** string GetInfoOChe**()**//вся информация о фирме

**{**

string alldata **=**""**;**

**for(**int i**=**0**;** i **<** count**;** i**++)**

**{**

alldata **+=**"\n"**+** ochered**[**i**].**GetInfo**();**

**}**

alldata **+=** GetSumm**();**

**return** alldata**;**

**}**

**public** string GetInfoForSave**()**//вся информация о фирме

**{**

string alldata **=** Convert**.**ToString**(**count**);**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

alldata **+=** $"\n{ochered[i].SurnameUrist}"**;**

**}**

alldata **+=** "\n"**;**

**for(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

alldata **+=** $"\n{ochered[i].getInfoFor()}"**;**

**}**

**return** alldata**;**

**}**

**}**

**}**

4. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков

При запуске программы создаётся экземпляр класса фирмы с входными параметрами: длина массива очереди и названием фирмы необходимое для записи в файл и инициализации сохранённых файлов программы.

**{**

**public** **partial** class Form1 **:** Form

**{**

Firm voenik **=** **new** Firm**(**"Призыва Нет!"**,** 10**);**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

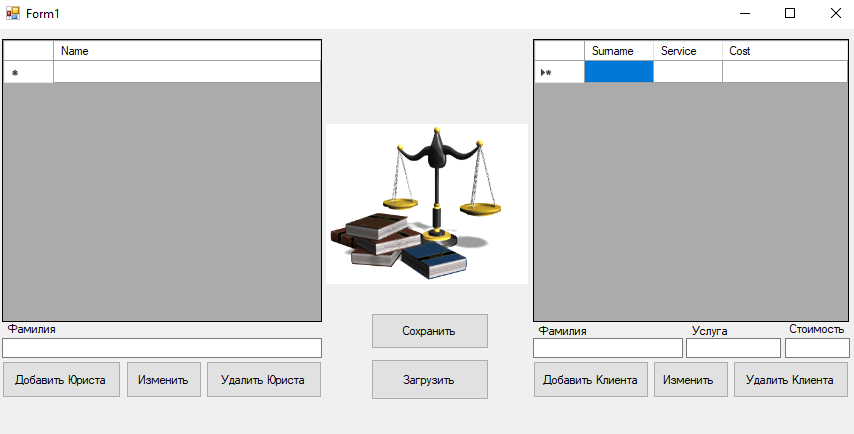


Рис.4 Запуск программы.

Устанавливается валидатор на ввод только числовых значений и корректного ввода Фамилии клиента, Фамилии адвоката и длины ввода с TextBox.

**private** void textFamUrist\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textServiceClient\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textNameClient\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textClientCost\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

textClientCost**.**MaxLength **=** 6**;**

**}**

**private** void textClientCost\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

char ch **=** e**.**KeyChar**;**

**if** **(!**Char**.**IsDigit**(**ch**)** **&&** ch **!=** 8**)**

**{**

e**.**Handled **=** **true;**

**}**

Метод добавления юриста как в таблицу так и в фирму. Данные беруться из TextBox.

**private** void AddUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textFamUrist**.**Text **!=** ""**)**

**{**

voenik**.**AddUrist**(**textFamUrist**.**Text**);**

dataGridView1**.**Rows**.**Add**(**textFamUrist**.**Text**);**

**}**

**else** MessageBox**.**Show**(**"Добавление не выполнено"**);**

**}**

Удаляется юрист как из фирмы так и из таблицы.

**private** void DeleteUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**voenik**.**Count **>** 0**)**

**{**

voenik**.**DellUrist**();**

dataGridView1**.**Rows**.**RemoveAt**(**0**);**

dataGridView2**.**Rows**.**Clear**();**

**}**

**else** MessageBox**.**Show**(**"Удалять нечего"**);**

**}**

При изменении данных соответствующие данные берутся из TextBox и изменяются в таблице и экземпляре.

**private** void ChangeUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textFamUrist**.**Text **!=** "" **&&** dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

voenik**.**SearchUrist**(**text**).**SurnameUrist **=** textFamUrist**.**Text**;**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textFamUrist**.**Text**;**

**}**

**else** MessageBox**.**Show**(**"Выберите нужный элемент"**);**

**}**

При добавлении клиента создаётся новый элемент в адресном списке и таблице.

**private** void AddClient\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **)**

**{**

**if** **(**textClientCost**.**Text **!=** "" **&&** textNameClient**.**Text **!=** "" **&&** textServiceClient**.**Text **!=** ""**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

voenik**.**SearchUrist**(**text**).**AddClient**(**textNameClient**.**Text**,** textServiceClient**.**Text**,** int**.**Parse**(**textClientCost**.**Text**));**

dataGridView2**.**Rows**.**Add**(**textNameClient**.**Text**,** textServiceClient**.**Text**,** textClientCost**.**Text**);**

**}**

**}**

**else** MessageBox**.**Show**(**"Добавление не выполнено"**);**

**}**

**private** void DeleteClient\_Click**(object** sender**,** EventArgs e**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

string text2**=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**

**if** **(**text **!=** "" **&&** text2 **!=** ""**)**

**{**

int index **=** dataGridView2**.**CurrentRow**.**Index**;**

voenik**.**SearchUrist**(**text**).**DellClient**(**text2**);**

dataGridView2**.**Rows**.**RemoveAt**(**index**);**

**}**

**else** MessageBox**.**Show**(**"Удаление не выполнено"**);**

**}**

**private** void ChangeClient\_Click**(object** sender**,** EventArgs e**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

string text2 **=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**

**if** **(**text **!=** "" **&&** text2 **!=** "" **&&** textClientCost**.**Text **!=** "" **&&** textNameClient**.**Text **!=** "" **&&** textServiceClient**.**Text **!=** ""**)**

**{**

voenik**.**SearchUrist**(**text**).**SearchCl**(**text2**).**Surname **=** textNameClient**.**Text**;**

voenik**.**SearchUrist**(**text**).**SearchCl**(**textNameClient**.**Text**).**Service **=** textServiceClient**.**Text**;**

voenik**.**SearchUrist**(**text**).**SearchCl**(**textNameClient**.**Text**).**Cost **=** Convert**.**ToInt32**(**textClientCost**.**Text**);**

**}**

**else** MessageBox**.**Show**(**"Изменение не выполнено"**);**

**}**

**private** void Form1\_Load**(object** sender**,** EventArgs e**){}**

При выборе юриста во второй таблице обновляются посетители выбранного юриста.

**private** void dataGridView1\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

dataGridView2**.**Rows**.**Clear**();**

Urist vspom **=** voenik**.**SearchUrist**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**());**

**if** **(**vspom **!=** **null)**

**{**

List pTemp **=** vspom**.**Head**;**

int chet **=** voenik**.**SearchUrist**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;**

**for** **(**int i **=** 0**;** i **<** chet**;** i**++)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**cClient**.**Surname**,** pTemp**.**cClient**.**Service**,** pTemp**.**cClient**.**Cost**);**

pTemp **=** pTemp**.**Second**;**

**}**

**}**

**}**

**else** MessageBox**.**Show**(**"Выбрана пустая строчка"**);**

**}**

Метод выгрузки данных в файл формата txt. Для дальнейшего чтения в начале файла пишем "Файл", чтобы можно было идентифицировать файл программы от других txt.

**private** void SaVeinFile\_Click**(object** sender**,** EventArgs e**)**

**{**

string path**;**

string fileText **=** "Призыва нет!\n"**;**

**using** **(**SaveFileDialog saveFile **=** **new** SaveFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** saveFile**.**FileName**;**

fileText **+=** voenik**.**GetInfoForSave**();**

**using** **(**FileStream file **=** **new** FileStream**(**path**,** FileMode**.**OpenOrCreate**))**

**using** **(**StreamWriter stream **=** **new** StreamWriter**(**file**))**

**{**

stream**.**Write**(**fileText**);**

stream**.**Close**();**

file**.**Close**();**

**}**

**}**

**}**

**}**

Метод загрузки из файла. Открывает диалоговое окно для отображения txt файлов. Если файл удалось открыть происходит проверка на корректные данные, т.е. первая сточка в файле должна быть "Файл"после этого происходит дальнейшее чтение файла, заполнение структуры и таблицы данными из файла.

**private** void LoaDinFile\_Click**(object** sender**,** EventArgs e**)**

**{**

string path **=** ""**,** doIt **=** ""**;**

int Count**,** CusCount**;**

**using** **(**OpenFileDialog openFile **=** **new** OpenFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**openFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** openFile**.**FileName**;**

**}**

**}**

**using** **(**StreamReader stream3 **=** **new** StreamReader**(**path**))**

**{**

doIt **=** stream3**.**ReadLine**();**

//Проверка на пригодность файла для работы с программой

**if** **(**doIt **==** "Призыва нет!"**)**

**{**

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

**{**

voenik**.**DellUrist**();**

dataGridView1**.**Rows**.**Clear**();**

dataGridView2**.**Rows**.**Clear**();**

**}**

Count **=** int**.**Parse**(**stream3**.**ReadLine**());**

**for** **(**int i **=** 1**;** i **<=** Count**;)**

**{**

string Surname **=** stream3**.**ReadLine**();**

voenik**.**AddUrist**(**Surname**);**

dataGridView1**.**Rows**.**Add**();**

dataGridView1**.**Rows**[**i **-** 1**].**Cells**[**0**].**Value **=** Surname**;**

**if** **(**i **!=** Count**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();** int hall **=** 0**;**

**while** **(**stream3**.**Peek**()** **!=** **-**1**)**

**{**

CusCount **=** int**.**Parse**(**stream3**.**ReadLine**());**

**if** **(**CusCount **!=** 0**)**

**for** **(**int i **=** 1**;** i **<=** CusCount**;)**

**{**

string surname **=** stream3**.**ReadLine**();**

string service **=** stream3**.**ReadLine**();**

int cost **=** int**.**Parse**(**stream3**.**ReadLine**());**

string name **=** Convert**.**ToString**(**dataGridView1**.**Rows**[**hall**].**Cells**[**0**].**Value**);**

voenik**.**SearchUrist**(**name**).**AddClient**(**surname**,** service**,** cost**);**

**if** **(**i **!=** CusCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else** **{** MessageBox**.**Show**(**"Данный файл не подходит для работы с программой"**);** **}**

stream3**.**Close**();**

**}**

**}**

**}**

**}**

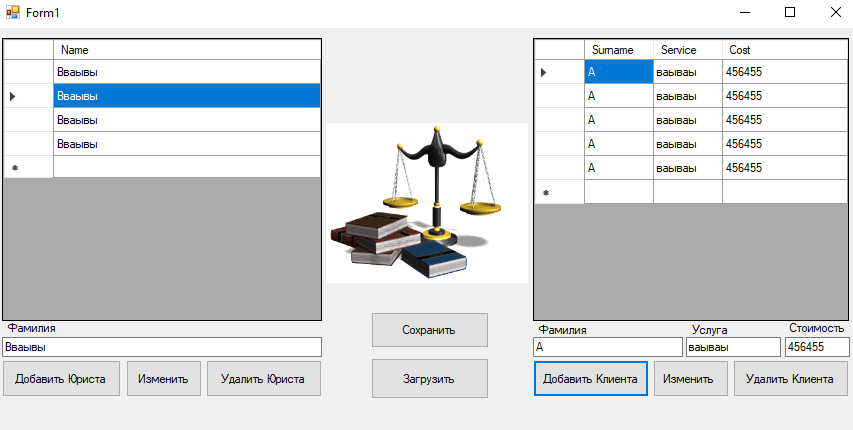


Рис 5 Работа программы.

1. Описание структуры проекта в соответствии с использованным инструментом разработки

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода.

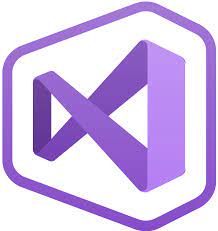


Рис 6 Visual Studio.

В главной папке проекта находятся разработанные классы \*.cs , так же в App.config находятся настройки для среды разработки Visual Studio находятся настройки конфигурации для запуска приложения. Файл \*.resx содержит в себе дизайн оконного приложения, составленный на языке разметки XML.

В папке Propertis находятся все подключенные в разработанных классах стандартные библиотеки для установления зависимостей в проекте.

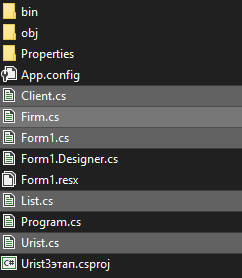


Рис. 7 Содержимое папки проекта.

Папка obj содержит в себе данные последнего Debug запуска проекта с соответствующими файлами конфигурации и логами.

В парке bin находиться главный исполняемый файл программы с последней сборки проекта, а также папка Relies с исполняемым файлом готовым к установке программы со всеми зависимостями на любой Windows операционной системе.

# **Список литературы**

1. Алгоритмы и структуры данных: учебное пособие / О.Б. Фофанов; Томский политехнический университет. – Томск: Изд-во Томского политех нического университета, 2019.
2. Васильев А. Н. Java. Объектно-ориентированное программирование. Учебное пособие. Стандарт третьего поколения. – "Издательский дом Питер 2021.
3. Кувшинов Д. Р., Осипов С. И. Основы программирования: язык C++: учебное пособие. – 2021.
4. Ляпин А. А., Быкова Ю. А. РАЗРАБОТКА БАЛЛИСТИЧЕСКОЙ ФУНКЦИИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C/C++ //ПОЛИ- ХОТОМИЧЕСКИЕ КЛАССИФИКАЦИИ СПЕЦИАЛЬНЫХ ТЕРМИ- НОВ В НАУЧНОЙ И УЧЕБНОЙ ЛИТЕРАТУРЕ. – 2021. – С. 308-311.
5. Объектно-ориентированное программирование. Учебно-методическое пособие/ Козин.А.Н., Якунина Е.А. – Казань: УВО «Университет управления «ТИСБИ», 2020.
6. Структуры и алгоритмы обработки данных. Часть 2. Усложнённые структуры данных. Теория и методика обучения. Учебно-методическое пособие / А.Н. Козин, Л.Б. Таренко. - Казань: УВО "Университет управления "ТИСБИ 2018
7. Павловская Т. А., Щупак Ю. А. C/C++. Структурное и объектно- ориентированное программирование. Практикум. – "Издательский дом Питер 2021.
8. Структуры и алгоритмы обработки данных. Часть 3. Поиск и сортировка данных. Теория и методика обучения. Учебно-методическое пособие/ А.Н. Козин, Л.Б. Таренко. - Казань: УВО "Университет управления "ТИСБИ 2020
9. Puuronen M. Implementing Horizontal Layout for the Qt Design Studio’s Component Library. – 2021.
10. Sherriff N. Learn Qt 5: Build modern, responsive cross-platform desktop applications with Qt, C++, and QML. – Packt Publishing Ltd, 2018.

Листинг программы

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**IO**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Text**.**RegularExpressions**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** Urist3этап

**{**

**public** **partial** class Form1 **:** Form

**{**

Firm voenik **=** **new** Firm**(**"Призыва Нет!"**,** 10**);**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

**private** void textFamUrist\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textServiceClient\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textNameClient\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textClientCost\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

textClientCost**.**MaxLength **=** 6**;**

**}**

**private** void textClientCost\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

char ch **=** e**.**KeyChar**;**

**if** **(!**Char**.**IsDigit**(**ch**)** **&&** ch **!=** 8**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void AddUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textFamUrist**.**Text **!=** ""**)**

**{**

voenik**.**AddUrist**(**textFamUrist**.**Text**);**

dataGridView1**.**Rows**.**Add**(**textFamUrist**.**Text**);**

**}**

**else** MessageBox**.**Show**(**"Добавление не выполнено"**);**

**}**

**private** void DeleteUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**voenik**.**Count **>** 0**)**

**{**

voenik**.**DellUrist**();**

dataGridView1**.**Rows**.**RemoveAt**(**0**);**

dataGridView2**.**Rows**.**Clear**();**

**}**

**else** MessageBox**.**Show**(**"Удалять нечего"**);**

**}**

**private** void ChangeUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textFamUrist**.**Text **!=** "" **&&** dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

voenik**.**SearchUrist**(**text**).**SurnameUrist **=** textFamUrist**.**Text**;**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textFamUrist**.**Text**;**

**}**

**else** MessageBox**.**Show**(**"Выберите нужный элемент"**);**

**}**

**private** void AddClient\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **)**

**{**

**if** **(**textClientCost**.**Text **!=** "" **&&** textNameClient**.**Text **!=** "" **&&** textServiceClient**.**Text **!=** ""**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

voenik**.**SearchUrist**(**text**).**AddClient**(**textNameClient**.**Text**,** textServiceClient**.**Text**,** int**.**Parse**(**textClientCost**.**Text**));**

dataGridView2**.**Rows**.**Add**(**textNameClient**.**Text**,** textServiceClient**.**Text**,** textClientCost**.**Text**);**

**}**

**}**

**else** MessageBox**.**Show**(**"Добавление не выполнено"**);**

**}**

**private** void DeleteClient\_Click**(object** sender**,** EventArgs e**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

string text2**=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**

**if** **(**text **!=** "" **&&** text2 **!=** ""**)**

**{**

int index **=** dataGridView2**.**CurrentRow**.**Index**;**

voenik**.**SearchUrist**(**text**).**DellClient**(**text2**);**

dataGridView2**.**Rows**.**RemoveAt**(**index**);**

**}**

**else** MessageBox**.**Show**(**"Удаление не выполнено"**);**

**}**

**private** void ChangeClient\_Click**(object** sender**,** EventArgs e**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

string text2 **=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**

**if** **(**text **!=** "" **&&** text2 **!=** "" **&&** textClientCost**.**Text **!=** "" **&&** textNameClient**.**Text **!=** "" **&&** textServiceClient**.**Text **!=** ""**)**

**{**

voenik**.**SearchUrist**(**text**).**SearchCl**(**text2**).**Surname **=** textNameClient**.**Text**;**

voenik**.**SearchUrist**(**text**).**SearchCl**(**textNameClient**.**Text**).**Service **=** textServiceClient**.**Text**;**

voenik**.**SearchUrist**(**text**).**SearchCl**(**textNameClient**.**Text**).**Cost **=** Convert**.**ToInt32**(**textClientCost**.**Text**);**

**}**

**else** MessageBox**.**Show**(**"Изменение не выполнено"**);**

**}**

**private** void Form1\_Load**(object** sender**,** EventArgs e**){}**

**private** void dataGridView1\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

dataGridView2**.**Rows**.**Clear**();**

Urist vspom **=** voenik**.**SearchUrist**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**());**

**if** **(**vspom **!=** **null)**

**{**

List pTemp **=** vspom**.**Head**;**

int chet **=** voenik**.**SearchUrist**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;**

**for** **(**int i **=** 0**;** i **<** chet**;** i**++)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**cClient**.**Surname**,** pTemp**.**cClient**.**Service**,** pTemp**.**cClient**.**Cost**);**

pTemp **=** pTemp**.**Second**;**

**}**

**}**

**}**

**else** MessageBox**.**Show**(**"Выбрана пустая строчка"**);**

**}**

**private** void SaVeinFile\_Click**(object** sender**,** EventArgs e**)**

**{**

string path**;**

string fileText **=** "Призыва нет!\n"**;**

**using** **(**SaveFileDialog saveFile **=** **new** SaveFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** saveFile**.**FileName**;**

fileText **+=** voenik**.**GetInfoForSave**();**

**using** **(**FileStream file **=** **new** FileStream**(**path**,** FileMode**.**OpenOrCreate**))**

**using** **(**StreamWriter stream **=** **new** StreamWriter**(**file**))**

**{**

stream**.**Write**(**fileText**);**

stream**.**Close**();**

file**.**Close**();**

**}**

**}**

**}**

**}**

**private** void LoaDinFile\_Click**(object** sender**,** EventArgs e**)**

**{**

string path **=** ""**,** doIt **=** ""**;**

int Count**,** CusCount**;**

**using** **(**OpenFileDialog openFile **=** **new** OpenFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**openFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** openFile**.**FileName**;**

**}**

**}**

**using** **(**StreamReader stream3 **=** **new** StreamReader**(**path**))**

**{**

doIt **=** stream3**.**ReadLine**();**

//Проверка на пригодность файла для работы с программой

**if** **(**doIt **==** "Призыва нет!"**)**

**{**

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

**{**

voenik**.**DellUrist**();**

dataGridView1**.**Rows**.**Clear**();**

dataGridView2**.**Rows**.**Clear**();**

**}**

Count **=** int**.**Parse**(**stream3**.**ReadLine**());**

**for** **(**int i **=** 1**;** i **<=** Count**;)**

**{**

string Surname **=** stream3**.**ReadLine**();**

voenik**.**AddUrist**(**Surname**);**

dataGridView1**.**Rows**.**Add**();**

dataGridView1**.**Rows**[**i **-** 1**].**Cells**[**0**].**Value **=** Surname**;**

**if** **(**i **!=** Count**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();** int hall **=** 0**;**

**while** **(**stream3**.**Peek**()** **!=** **-**1**)**

**{**

CusCount **=** int**.**Parse**(**stream3**.**ReadLine**());**

**if** **(**CusCount **!=** 0**)**

**for** **(**int i **=** 1**;** i **<=** CusCount**;)**

**{**

string surname **=** stream3**.**ReadLine**();**

string service **=** stream3**.**ReadLine**();**

int cost **=** int**.**Parse**(**stream3**.**ReadLine**());**

string name **=** Convert**.**ToString**(**dataGridView1**.**Rows**[**hall**].**Cells**[**0**].**Value**);**

voenik**.**SearchUrist**(**name**).**AddClient**(**surname**,** service**,** cost**);**

**if** **(**i **!=** CusCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else** **{** MessageBox**.**Show**(**"Данный файл не подходит для работы с программой"**);** **}**

stream3**.**Close**();**

**}**

**}**

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class List

**{**

**private** Client client**;**//важные данные

**private** List last**,** first**,** second**,**pred**;**//переменные-ссылки

**public** List **(**Client \_client**)**//конструктор

**{**

**this.**client **=** \_client**;**

**this.**last **=** **null;**

**this.**first **=** **null;**

**this.**second **=** **null;**

**}**

**public** Client cClient **{** **get** **{** **return** client**;** **}set** **{** **this.**client **=** **value;** **}** **}**

**public** List Second//гет и сет для следующего элемента

**{**

**get** **{** **return** second**;** **}**

**set** **{** second **=** **value;** **}**

**}**

**public** List Last//гет и сет для последнего элемента

**{**

**get** **{** **return** last**;** **}**

**set** **{** last **=** **value;** **}**

**}**

**public** List First//гет и сет для первого элемента

**{**

**get** **{** **return** first**;** **}**

**set** **{** first **=** **value;** **}**

**}**

**public** List Pred

**{**

**get** **{** **return** pred**;** **}**

**set** **{** **this.**pred **=** **value;** **}**

**}**

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class Urist

**{**

**private** string surnameUrist**;**//фамилия юриста

**private** int sum**,** count**;**//сумма полученных денег этим юристом

**private** List head**,** pTemp**,**pHelp**;**//заголовок списка и вспомогательная переменная

**public** Urist **(**string \_surnameUrist**)**//конструктор класса

**{**

**this.**surnameUrist **=** \_surnameUrist**;**

head **=** **new** List**(null);**

head**.**First **=** **null;**

head**.**Last **=** **null;**

**}**

**public** int Count **{** **get** **{** **return** count**;** **}** **}**

**public** List Head **{** **get** **{** **return** head**.**First**;** **}** **}**

**public** string SurnameUrist **{** **get** **{** **return** surnameUrist**;** **}** **set** **{** **this.**surnameUrist **=** **value;** **}** **}**

**public** void AddClient**(**string \_surname**,** string \_service**,** int \_cost**)**

**{**

Client dobav **=** **new** Client**(**\_surname**,** \_service**,** \_cost**);**

List dobavs **=** **new** List**(**dobav**);**

count**++;**

**if** **(**head**.**First **==** **null)**

**{**

head**.**First **=** dobavs**;**

head**.**First**.**Pred **=** **null;**

head**.**First**.**Second **=** head**.**Last**;**

head**.**Last **=** dobavs**;**

**}**

**else**

**{**

pHelp **=** head**.**Last**;**

head**.**Last **=** dobavs**;**

pHelp**.**Second **=** head**.**Last**;**

head**.**Last**.**Pred **=** pHelp**;**

**}**

**}**

**public** void DellClient**(**string \_surname**)**

**{**

pHelp **=** head**.**First**;**

**while** **(**pHelp **!=** **null)**

**{**

List pTemp**;**

**if** **(**pHelp **==** head**.**First **&&** pHelp**.**cClient**.**Surname**.**Equals**(**\_surname**))**

**{**

**if** **(**pHelp**.**Second **==** **null)**

**{**

head**.**First **=** **null;**

**}**

**else**

**{**

pTemp **=** head**.**First**.**Second**;**

pHelp**.**Second**.**Pred **=** **null;**

pHelp**.**Second **=** **null;**

head**.**First **=** pTemp**;**

**}**

**}**

**else** **if** **(**pHelp**.**cClient**.**Surname**.**Equals**(**\_surname**))**

**{**

**if** **(**pHelp**.**Pred **!=** **null** **&&** pHelp**.**Second **!=** **null)**

**{**

pHelp**.**Pred**.**Second **=** pHelp**.**Second**;**

pHelp**.**Second**.**Pred **=** pHelp**.**Pred**;**

**}**

**else**

**{**

head**.**Last **=** head**.**Last**.**Pred**;**

**}**

**}**

pHelp **=** pHelp**.**Second**;**

**}**

**}**

**public** Client SearchCl**(**string \_surname**)**

**{**

List vspom **=** head**.**First**;**

**while(**vspom **!=** **null)**

**{**

**if** **(**vspom**.**cClient**.**Surname**.**Equals**(**\_surname**))**

**{**

**return** vspom**.**cClient**;**

**}**

**else** **{** vspom **=** vspom**.**Second**;** **}**

**}**

**return** **null;**

**}**

**public** int Sum**()**

**{**

sum **=** 0**;**

List vspom **=** head**.**First**;**

**while(**vspom **!=** **null)**

**{**

sum **+=** vspom**.**cClient**.**Cost**;**

vspom **=** vspom**.**Second**;**

**}**

**return** sum**;**

**}**

**public** string GetInfo**()**

**{**

string info **=** ""**;**

List vspom **=** head**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

info **+=** vspom**.**cClient**.**Surname**;**

info **+=** vspom**.**cClient**.**Cost**;**

**}**

info **+=** Sum**();**

**return** info**;**

**}**

**public** string getInfoFor**()**

**{**

string info **=** Convert**.**ToString**(**count**);**

List vspom **=** head**.**First**;**

**while** **(**vspom **!=** **null)**

**{**

info **+=** $"\n{vspom.cClient.Surname}"**;**

info **+=** $"\n{vspom.cClient.Service}"**;**

info **+=** $"\n{vspom.cClient.Cost}"**;**

vspom **=** vspom**.**Second**;**

**}**

**return** info**;**

**}**

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class Client

**{**

**private** string surname**,** service**;**//фамилия клиента, оказанные услуги

**private** int cost**;**//стоимость услуг

**public** Client**(**string \_surname**,** string \_service**,** int \_cost**)**

**{**

**this.**surname **=** \_surname**;**

**this.**service **=** \_service**;**

**this.**cost **=** \_cost**;**

**}**

**public** string Surname//методы доступа для фамилии

**{** **get** **{** **return** surname**;** **}** **set** **{** surname **=** **value;** **}** **}**

**public** int Cost //методы доступа для стоимости

**{** **get** **{** **return** cost**;** **}** **set** **{** cost **=** **value;** **}** **}**

**public** string Service //методы доступа для оказанной услуги

**{** **get** **{** **return** service**;** **}** **set** **{** service **=** **value;** **}** **}**

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** Urist3этап

**{**

**internal** class Firm

**{**

**private** string nameFirm**;**//название фирмы

**private** Urist**[]** ochered**;**//массив обьектов класса юрист

**private** int size**,** first**,** last**,** count**,** sum**;**//необходимые переменные для

//подсчета суммы, размера массива и координации в очереди

**public** Firm**(**string \_nameFirm**,** int \_size**)**//констуктор

**{**

**this.**nameFirm **=** \_nameFirm**;**

**this.**size **=** \_size**;this.**last **=** 0**;**

**this.**first **=** 0**;** **this.**count **=** 0**;**

ochered **=** **new** Urist**[**size**];**

**}**

**public** int Count **{** **get** **{** **return** count**;** **}** **}**

**public** void AddUrist**(**string \_surnameUrist**)**//метод добавления

**{**

**if(**count **<=** size**)**

**{**

Urist dobav **=** **new** Urist**(**\_surnameUrist**);**

ochered**[**last**]** **=** dobav**;**

last**++;** count**++;**

**}**

**}**

**public** void DellUrist**()**//метод удаления

**{**

**if** **(**count **>** 0**)**

**{**

ochered**[**first**]** **=** **null;**

count**--;**

**for** **(**int i **=** **(**first **+** 1**);** i **<** last**;** i**++)**

**{**

ochered**[**i **-** 1**]** **=** ochered**[**i**];**

**}**

last**--;**

ochered**[**last**]** **=** **null;**

**}**

**}**

**public** Urist SearchUrist**(**string \_surnameUrist**)**//поиск для дальнейшего добавления и удаления клиентов

**{**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

**if** **(**ochered**[**i**]** **!=** **null** **&&** ochered**[**i**].**SurnameUrist**.**Equals**(**\_surnameUrist**))**

**{**

**return** ochered**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** int GetSumm**()**//сумма полученных денег

**{**

sum **=** 0**;**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

sum **+=** ochered**[**i**].**Sum**();**

**}**

**return** sum**;**

**}**

**public** string GetInfoOChe**()**//вся информация о фирме

**{**

string alldata **=**""**;**

**for(**int i**=**0**;** i **<** count**;** i**++)**

**{**

alldata **+=**"\n"**+** ochered**[**i**].**GetInfo**();**

**}**

alldata **+=** GetSumm**();**

**return** alldata**;**

**}**

**public** string GetInfoForSave**()**//вся информация о фирме

**{**

string alldata **=** Convert**.**ToString**(**count**);**

**for** **(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

alldata **+=** $"\n{ochered[i].SurnameUrist}"**;**

**}**

alldata **+=** "\n"**;**

**for(**int i **=** 0**;** i **<** count**;** i**++)**

**{**

alldata **+=** $"\n{ochered[i].getInfoFor()}"**;**

**}**

**return** alldata**;**

**}**

**}**

**}**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**IO**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Text**.**RegularExpressions**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** Urist3этап

**{**

**public** **partial** class Form1 **:** Form

**{**

Firm voenik **=** **new** Firm**(**"Призыва Нет!"**,** 10**);**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

**private** void textFamUrist\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textServiceClient\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textNameClient\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

string Symbol **=** e**.**KeyChar**.**ToString**();**

**if** **(!**Regex**.**Match**(**Symbol**,** @"[а-яА-Я]|[a-zA-Z]"**).**Success **&&** e**.**KeyChar **!=** **(**char**)**Keys**.**Back**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void textClientCost\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

textClientCost**.**MaxLength **=** 6**;**

**}**

**private** void textClientCost\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

char ch **=** e**.**KeyChar**;**

**if** **(!**Char**.**IsDigit**(**ch**)** **&&** ch **!=** 8**)**

**{**

e**.**Handled **=** **true;**

**}**

**}**

**private** void AddUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textFamUrist**.**Text **!=** ""**)**

**{**

voenik**.**AddUrist**(**textFamUrist**.**Text**);**

dataGridView1**.**Rows**.**Add**(**textFamUrist**.**Text**);**

**}**

**else** MessageBox**.**Show**(**"Добавление не выполнено"**);**

**}**

**private** void DeleteUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**voenik**.**Count **>** 0**)**

**{**

voenik**.**DellUrist**();**

dataGridView1**.**Rows**.**RemoveAt**(**0**);**

dataGridView2**.**Rows**.**Clear**();**

**}**

**else** MessageBox**.**Show**(**"Удалять нечего"**);**

**}**

**private** void ChangeUrist\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**textFamUrist**.**Text **!=** "" **&&** dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

voenik**.**SearchUrist**(**text**).**SurnameUrist **=** textFamUrist**.**Text**;**

dataGridView1**.**Rows**[**dataGridView1**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textFamUrist**.**Text**;**

**}**

**else** MessageBox**.**Show**(**"Выберите нужный элемент"**);**

**}**

**private** void AddClient\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null** **)**

**{**

**if** **(**textClientCost**.**Text **!=** "" **&&** textNameClient**.**Text **!=** "" **&&** textServiceClient**.**Text **!=** ""**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

voenik**.**SearchUrist**(**text**).**AddClient**(**textNameClient**.**Text**,** textServiceClient**.**Text**,** int**.**Parse**(**textClientCost**.**Text**));**

dataGridView2**.**Rows**.**Add**(**textNameClient**.**Text**,** textServiceClient**.**Text**,** textClientCost**.**Text**);**

**}**

**}**

**else** MessageBox**.**Show**(**"Добавление не выполнено"**);**

**}**

**private** void DeleteClient\_Click**(object** sender**,** EventArgs e**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

string text2**=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**

**if** **(**text **!=** "" **&&** text2 **!=** ""**)**

**{**

int index **=** dataGridView2**.**CurrentRow**.**Index**;**

voenik**.**SearchUrist**(**text**).**DellClient**(**text2**);**

dataGridView2**.**Rows**.**RemoveAt**(**index**);**

**}**

**else** MessageBox**.**Show**(**"Удаление не выполнено"**);**

**}**

**private** void ChangeClient\_Click**(object** sender**,** EventArgs e**)**

**{**

string text **=** dataGridView1**.**CurrentCell**.**Value**.**ToString**();**

string text2 **=** dataGridView2**.**CurrentCell**.**Value**.**ToString**();**

**if** **(**text **!=** "" **&&** text2 **!=** "" **&&** textClientCost**.**Text **!=** "" **&&** textNameClient**.**Text **!=** "" **&&** textServiceClient**.**Text **!=** ""**)**

**{**

voenik**.**SearchUrist**(**text**).**SearchCl**(**text2**).**Surname **=** textNameClient**.**Text**;**

voenik**.**SearchUrist**(**text**).**SearchCl**(**textNameClient**.**Text**).**Service **=** textServiceClient**.**Text**;**

voenik**.**SearchUrist**(**text**).**SearchCl**(**textNameClient**.**Text**).**Cost **=** Convert**.**ToInt32**(**textClientCost**.**Text**);**

**}**

**else** MessageBox**.**Show**(**"Изменение не выполнено"**);**

**}**

**private** void Form1\_Load**(object** sender**,** EventArgs e**){}**

**private** void dataGridView1\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

**if** **(**dataGridView1**.**CurrentCell**.**Value **!=** **null)**

**{**

dataGridView2**.**Rows**.**Clear**();**

Urist vspom **=** voenik**.**SearchUrist**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**());**

**if** **(**vspom **!=** **null)**

**{**

List pTemp **=** vspom**.**Head**;**

int chet **=** voenik**.**SearchUrist**(**dataGridView1**.**CurrentCell**.**Value**.**ToString**()).**Count**;**

**for** **(**int i **=** 0**;** i **<** chet**;** i**++)**

**{**

dataGridView2**.**Rows**.**Add**(**pTemp**.**cClient**.**Surname**,** pTemp**.**cClient**.**Service**,** pTemp**.**cClient**.**Cost**);**

pTemp **=** pTemp**.**Second**;**

**}**

**}**

**}**

**else** MessageBox**.**Show**(**"Выбрана пустая строчка"**);**

**}**

**private** void SaVeinFile\_Click**(object** sender**,** EventArgs e**)**

**{**

string path**;**

string fileText **=** "Призыва нет!\n"**;**

**using** **(**SaveFileDialog saveFile **=** **new** SaveFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**saveFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** saveFile**.**FileName**;**

fileText **+=** voenik**.**GetInfoForSave**();**

**using** **(**FileStream file **=** **new** FileStream**(**path**,** FileMode**.**OpenOrCreate**))**

**using** **(**StreamWriter stream **=** **new** StreamWriter**(**file**))**

**{**

stream**.**Write**(**fileText**);**

stream**.**Close**();**

file**.**Close**();**

**}**

**}**

**}**

**}**

**private** void LoaDinFile\_Click**(object** sender**,** EventArgs e**)**

**{**

string path **=** ""**,** doIt **=** ""**;**

int Count**,** CusCount**;**

**using** **(**OpenFileDialog openFile **=** **new** OpenFileDialog**()** **{** Filter **=** "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*" **})**

**{**

**if** **(**openFile**.**ShowDialog**()** **==** DialogResult**.**OK**)**

**{**

path **=** openFile**.**FileName**;**

**}**

**}**

**using** **(**StreamReader stream3 **=** **new** StreamReader**(**path**))**

**{**

doIt **=** stream3**.**ReadLine**();**

//Проверка на пригодность файла для работы с программой

**if** **(**doIt **==** "Призыва нет!"**)**

**{**

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

**{**

voenik**.**DellUrist**();**

dataGridView1**.**Rows**.**Clear**();**

dataGridView2**.**Rows**.**Clear**();**

**}**

Count **=** int**.**Parse**(**stream3**.**ReadLine**());**

**for** **(**int i **=** 1**;** i **<=** Count**;)**

**{**

string Surname **=** stream3**.**ReadLine**();**

voenik**.**AddUrist**(**Surname**);**

dataGridView1**.**Rows**.**Add**();**

dataGridView1**.**Rows**[**i **-** 1**].**Cells**[**0**].**Value **=** Surname**;**

**if** **(**i **!=** Count**)**

i**++;**

**else**

**break;**

**}**

stream3**.**ReadLine**();** int hall **=** 0**;**

**while** **(**stream3**.**Peek**()** **!=** **-**1**)**

**{**

CusCount **=** int**.**Parse**(**stream3**.**ReadLine**());**

**if** **(**CusCount **!=** 0**)**

**for** **(**int i **=** 1**;** i **<=** CusCount**;)**

**{**

string surname **=** stream3**.**ReadLine**();**

string service **=** stream3**.**ReadLine**();**

int cost **=** int**.**Parse**(**stream3**.**ReadLine**());**

string name **=** Convert**.**ToString**(**dataGridView1**.**Rows**[**hall**].**Cells**[**0**].**Value**);**

voenik**.**SearchUrist**(**name**).**AddClient**(**surname**,** service**,** cost**);**

**if** **(**i **!=** CusCount**)**

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

**else** **{** MessageBox**.**Show**(**"Данный файл не подходит для работы с программой"**);** **}**

stream3**.**Close**();**

**}**

**}**

**}**

**}**