УВО «Университет Управления «ТИСБИ»

Факультет информационных технологий

Кафедра информационных технологий

**Курсовая работа**

по дисциплине «Объектно-ориентированное программирование»

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

Выполнил: студент гр. П-011 Сучёв Н.Е.

Проверил: ст. преп. Якунина Е.А.

Казань 2022

Оглавление

[Постановка задачи 3](#_Toc104845698)

[1. Описание используемых структур данных с алгоритмами выполнения основных операций 5](#_Toc104845699)

[1.1 Адресный замкнутый (кольцевой) упорядоченный однонаправленный список без заголовка. 5](#_Toc104845700)

[1.2 Кольцевая очередь на основе обычного массива 8](#_Toc104845701)

[2. Краткие сведения об объектном подходе 10](#_Toc104845702)

[3. Формализованное описание разработанных классов 16](#_Toc104845703)

[1. Описание демонстрационного модуля с характеристикой использованных стандартных компонентов и списком реализованных обработчиков 21](#_Toc104845704)

[2. Описание структуры проекта в соответствии с использованным инструментом разработки 36](#_Toc104845705)

[Листинг программы 40](#_Toc104845706)

Постановка задачи

**Цели работы:** отработка навыков курсов «объектно-ориентированное программирование» и «структуры и алгоритмы обработки данных».

**Постановка задачи:** разработать объектную программу для хранения данных об услугах, оказанных некоторой юридической фирмой. Учёт организуется за счёт ведения списка юристов фирмы с указанием фамилии и номера мобильного телефона. Для каждого юриста создать отдельный список его клиентов с указанием фамилии клиента, вида услуги и её 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 Очередь на основе массива.

Очередь – структура данных типа «список», позволяющая добавлять элементы лишь в конец списка, и извлекать их из его начала. Она функционирует по принципу FIFO (First In, First Out — «первым пришёл — первым вышел»), для которого характерно, что все элементы a1, a2, …, an-1, an, добавленные раньше элемента an+1, должны быть удалены прежде, чем будет удален элемент an+1. Также очередь может быть определена как частный случай односвязного списка, который обслуживает элементы в порядке их поступления. Как и в «живой» очереди, здесь первым будет обслужен тот, кто пришел первым.

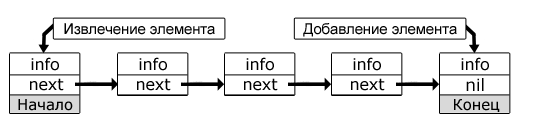


Рис. 1 Очередь.

Стандартный набор операций (часто у разных авторов он не идентичен), выполняемых над очередями, совпадает с тем, что используется при обработке стеков:

* добавление элемента;
* удаление элемента;
* чтение первого элемента.

Только, если в отношении стека в момент добавления или удаления элемента допустимо задействование лишь его вершины, то касательно очереди эти две операции должны быть применены так, как это регламентировано в определении этой структуры данных, т. е. добавление – в конец, удаление – из начала. Далее, при реализации интерфейса очереди, список стандартных операций будет расширен.

Выделяют два способа программной реализации очереди. Первый из них основан на базе массива, а второй на базе указателей (связного списка). Первый способ – статический, т. к. очередь представляется в виде простого статического массива, второй – динамический.

1.2 Адресный стек.

Стек представляет собой структуру данных, которая работает по принципу LIFO (Last In First Out - "последний пришел - первый вышел"). Графически стек можно представить в виде столбика или стопки объектов:

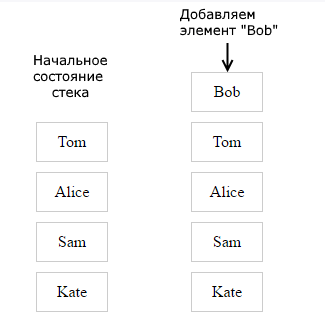


Рис. 2 Стек

Стек имеет вершину, который образует последний добавленный элемент. При добавлении новый элемент помещается поверх вершины стека и образует новую вершину. При удалении удаляется элемент из вершины стека, а предыдущий элемент образует новую вершину.

Так, на приведенном рисунке вначале вершиной стека является "Tom". После добавления нового элемента "Bob" этот элемент располагается поверх элемента "Tom" и становится новой вершиной.

В библиотеке классов .NET в принципе уже есть свой класс, который выполняет роль стека. Это класс - System.Collections.Generic.Stack. Но рассмотрим, как мы сами можем реализовать структуру в виде стека.

Структура стек вне зависимости от языка программирования обладает неким общим функционалом, который составляют метод добавления элемента (как правило, называется push()) и метод извлечения элемента из вершины стека (обычно называется pop()). Кроме того, нередко реализации стеков содержат метод получения элемента из вершины без его извлечения, метод определения размера стека и ряд других.

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

В иерархии наследования класс Customers находиться на самой вершине его наследует класс Lawyer содержащий сведенья о посетителях адвоката.

// Сustomer

**using** System**;**

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

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

**namespace** Third

**{**

class Customers

**{**

**private** string surname**;**//фамилия заказчика

**private** string service**;**//оказанная услуга

**private** int cost**;**//стоимость услуги

**public** Customers**(**string \_surname**,** string \_service**,** int \_cost**)**//конструктор

**{**

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

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

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

**}**

**public** string Surname //get/set для фамилии

**{**

**get** **{** **return** surname**;** **}**

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

**}**

**public** string Service//get/set для оказанной услуги

**{**

**get** **{** **return** service**;** **}**

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

**}**

**public** int Cost//get/set для стоимости

**{**

**get** **{** **return** cost**;** **}**

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

**}**

**}**

**}**

Класс Lower хранит в себе сведенья о юристе (Фамилию). Данные о посетителях в виде очереди на массиве, а также заработанную сумму от всех посетителей.

**// Lawyer**

**using** System**;**

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

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

**namespace** Third

**{**

class Lawyer

**{**

**private** string lSurname**;**//фамилия юриста

**private** long number**;**

**private** Customers**[]** queue**;**//массив-очередь

**private** int first**;**//ссылка на 1 элемент

**private** int last**;**//ссылка на последний элемент

**private** int sumCost**,** size**,**count**;**//сумма, потраченная на услуги

**public** Lawyer**(**string \_lSurname**,** long \_number**,** int \_size**)** //конструктор

**{**

**this.**lSurname **=** \_lSurname**;**

**this.**number **=** \_number**;**

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

queue **=** **new** Customers**[**size**];**

first **=** 0**;** last **=** 0**;** count **=** 0**;**

**}**

**public** Customers Search**(**string \_lSurname**)**

**{**

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

**{**

**if(**queue**[**i**].**Surname**.**Equals**(**\_lSurname**))**

**{**

**return** queue**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** long Number **{** **get** **{** **return** number**;** **}** **set** **{** **this.**number **=** **value;** **}** **}**

**public** Customers**[]** Queue **{** **get** **{** **return** queue**;** **}** **}**

**public** string Surname**{** **get** **{** **return** lSurname**;** **}set** **{** **this.**lSurname **=** **value;** **}** **}**

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

**public** void addCust**(**string \_surname**,** string \_service**,** int \_cost**)** //добавление заказчика

**{**

**if** **(**size **>=** count**)**

**{**

**if** **(**first **>** 0 **&&** last **>=** size**)** last **=** 0**;**

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

queue**[**last**]** **=** dobav**;**

last**++;**

count**++;**

**}**

**}**

**public** void delCust**()** //удаление заказчика по переменной first

**{**

**if** **(**count **!=** 0**)**

**{**

**if** **(**first **>** size**)** first **=** 0**;**

queue**[**first**]** **=** **null;**

count**--;**

first**++;**

**}**

**}**

**public** int SumCost**()**//возвращает сумму потраченных денег

**{**

sumCost **=** 0**;**

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

**{**

**if** **(**queue**[**i**]** **!=** **null)**

**{**

sumCost **+=** queue**[**i**].**Cost**;**

**}**

**}**

**return** sumCost**;**

**}**

**public** string getInfo**()** //получение информации об одном юристе и его заказчиках

**{**

string getinfo **=** "\n<=======Lawyer: " **+** lSurname **+** "=======>"**;**

Customers help**;**

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

**{**

help **=** queue**[**i**];**

**if** **(**help **!=** **null)**

**{**

getinfo **+=** "\n\nCustomer: " **+** help**.**Surname**;**

getinfo **+=** "\nService: " **+** help**.**Service**;**

getinfo **+=** "\nCost: " **+** help**.**Cost **+** "\n"**;**

**}**

**}**

getinfo **+=** "\nsum of cost: " **+** SumCost**();**

**return** getinfo**;**

**}**

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

**{**

string getinfo **=** Convert**.**ToString**(**Count**);**

Customers help**;**

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

**{**

help **=** queue**[**i**];**

**if** **(**help **!=** **null)**

**{**

getinfo **+=** "\n" **+** help**.**Surname**;**

getinfo **+=** "\n" **+** help**.**Service**;**

getinfo **+=** "\n" **+** help**.**Cost**;**

**}**

**}**

**return** getinfo**;**

**}**

**}**

**}**

Класс Company содержит в себе экземпляры с юристами, хранящимися в контейнере – стек. Company является самым низшим классом в наследовании и позволяет обращаться к методам класса Lower и Customers.

// Company

**using** System**;**

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

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

**namespace** Third

**{**

class Company

**{**

**private** string nameComp**;**//название юр.фирмы

**private** Stack firstElenem**,**SP**;**//первый элемент в стеке

**private** int countElem**;**//количество элементов

**public** Company**(**string \_nameComp**)** //конструктор

**{**

**this.**nameComp **=** \_nameComp**;**

SP **=** **null;**

firstElenem **=** **null;**

countElem **=** 0**;**

**}**

**public** string Info**()**

**{**

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

Stack vspom **=** SP**;**

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

**{**

info **+=** "\n" **+** vspom**.**date**.**Surname**;**

info **+=** "\n" **+** vspom**.**date**.**Number**;**

vspom **=** vspom**.**getLaw**();**

**}**

**return** info**;**

**}**

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

**public** void addLaw**(**string \_lSurname**,** long \_number**,** int \_size**)** //добавление юриста

**{**

Lawyer dobLaw **=** **new** Lawyer**(**\_lSurname**,** \_number**,** \_size**);**

Stack help **=** **new** Stack**(**dobLaw**,** SP**);**

**if** **(**SP **==** **null)** firstElenem **=** help**;**

SP **=** help**;**

countElem**++;**

**}**

**public** void delLaw**()** //удаление юриста

**{**

**if(**SP **!=** **null)**

**{**

SP **=** SP**.**getLaw**();**

countElem**--;**

**}**

**}**

**public** Lawyer searchLaw**(**string \_lSurname**)** //поиск юриста по фамилии

**{**

Stack vspom **=** SP**;**

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

**{**

**if** **(**vspom**.**date**.**Surname**.**Equals**(**\_lSurname**))**

**{**

**return** vspom**.**date**;**

**}**

**else** vspom **=** vspom**.**getLaw**();**

**}**

**return** **null;**

**}**

**public** int SumCostLaw**()**//заработок всей компании

**{**

int summ **=** 0**;**

Stack vspom **=** SP**;**

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

**{**

summ **+=** vspom**.**date**.**SumCost**();**

vspom **=** vspom**.**getLaw**();**

**}**

**return** summ**;**

**}**

**public** string getInfoComp**()** //получение всей информации о компании

**{**

string getInfoAll **=** "\n\nName Company: " **+** nameComp**;**

Stack vspom **=** SP**;**

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

**{**

getInfoAll **+=** "\n" **+** vspom**.**date**.**getInfo**()** **+** "\n"**;**

vspom **=** vspom**.**getLaw**();**

**}**

getInfoAll **+=** "\n[summ Cost all company: " **+** SumCostLaw**()** **+** "]"**;**

**return** getInfoAll**;**

**}**

**public** string getInfoFor**()** //получение всей информации о компании

**{**

string getInfoAll **=** ""**;**

Stack vspom **=** SP**;**

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

**{**

getInfoAll **+=** "\n" **+** vspom**.**date**.**getInfoFor**();**

vspom **=** vspom**.**getLaw**();**

**}**

**return** getInfoAll**;**

**}**

**}**

**}**

В классе Company данные о юристах храниться в стеке. В программе реализована структура стек на основе адресных переменных.

// Stack

**using** System**;**

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

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

**namespace** Third

**{**

class Stack

**{**

**public** Lawyer date**;**//Данные

**private** Stack pred**;**//ссылка на предка

**public** Stack**(**Lawyer \_date**,** Stack \_SP**)**//конструктор

**{**

**this.**date **=** \_date**;**

pred **=** \_SP**;**

**}**

**public** Stack getLaw**()**//получение ссылки на предка

**{**

**return** pred**;**

**}**

**}**

**}**

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

При запуске программы создаётся экземпляр класса Company с названием компании необходимое для создания файлов сохранения данных в файл.

// Form1

**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**.**Threading**.**Tasks**;**

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

**namespace** Third

**{**

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

**{**

Company Akyi **=** **new** Company**(**"Company Akyi"**);**

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

**{**

InitializeComponent**();**

**}**

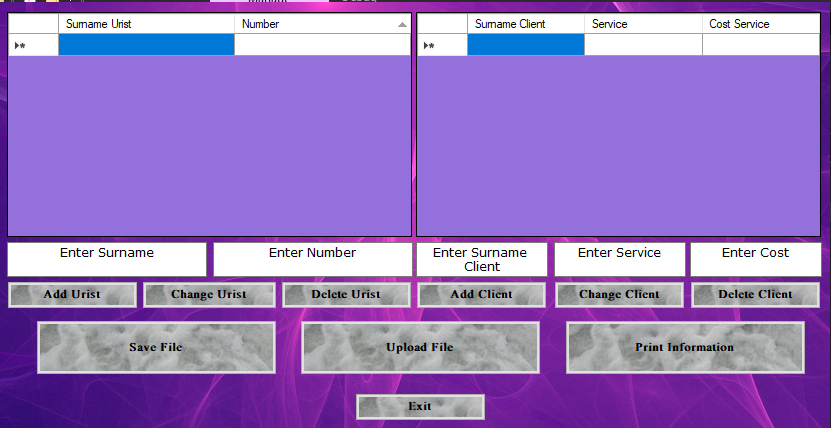


Рис. 3 Запуск программы.

Прописываем события для существующих TextBox, что при начале ввода, старая информация стирается.

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

**{**

textSurUrist**.**Clear**();**

**}**

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

**{**

textCost**.**Clear**();**

**}**

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

**{**

textNumber**.**Clear**();**

**}**

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

**{**

textSurClient**.**Clear**();**

**}**

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

**{**

textService**.**Clear**();**

**}**

Данный метод отрабатывает событие, когда пользователь выбирает клиента в первой таблице клиентов. Метод необходим для дальнейшего добавления посещений конкретному пациенту. Если пользователь не выбрал строчку, то программа не будет работать. А также у выбранного юриста показывает сведенья о посетителях заполняя вторую таблицу.

**private** void dataGridView2\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

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

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

Lawyer vspom **=** Akyi**.**searchLaw**(**text**);**

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

**{**

Customers**[]** copy1 **=** vspom**.**Queue**;**

int chet **=** Akyi**.**searchLaw**(**text**).**Count**;**

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

**{**

dataGridView1**.**Rows**.**Add**(**copy1**[**i**].**Surname**,** copy1**[**i**].**Service**,**copy1**[**i**].**Cost**);**

**}**

**}**

**}**

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

**{**

**}**

Метод кнопки добавления юриста. Добавляет данные фамилии в таблицу. Создаёт экземпляр в стеке класса Lower.

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

**{**

Akyi**.**addLaw**(**textSurUrist**.**Text**,** Convert**.**ToInt64**(**textNumber**.**Text**),** 10**);**

dataGridView2**.**Rows**.**Add**(**textSurUrist**.**Text**,** textNumber**.**Text**);**

textSurUrist**.**Clear**();**textNumber**.**Clear**();**

**}**

Метод закрывает окно программы.

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

**{**

Close**();**

**}**

Метод обрабатывающий событие нажатия пользователем кнопки добавления посетителя юристу. Происходит валидация данных с TextBox, добавление в классе Lower экземпляра Customers и добавление данных в таблицу.

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

**{**

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

Akyi**.**searchLaw**(**info**).**addCust**(**textSurClient**.**Text**,**textService**.**Text**,**Convert**.**ToInt32**(**textCost**.**Text**));**

dataGridView1**.**Rows**.**Add**(**textSurClient**.**Text**,** textService**.**Text**,** textCost**.**Text**);**

**}**

Валидатор ввода номера на максимальное значение из 10^11. А таже ограничение на ввод только цифр.

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

**{**

textNumber**.**MaxLength **=** 11**;**

**}**

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

**{**

// ввод в texBox только цифр и кнопки Backspace

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

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

**{**

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

**}**

**}**

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

**{**

// ввод в texBox только цифр и кнопки Backspace

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

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

**{**

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

**}**

**}**

Валидатор на имя пользователя длиной не более 10 символов.

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

**{**

textCost**.**MaxLength **=** 10**;**

**}**

Метод изменения данных о юристе. После валидации TextBox происходит обращение к методам Company для ввода новых значений и обновление таблицы юристов.

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

**{**

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

Akyi**.**searchLaw**(**info**).**Surname **=** textSurUrist**.**Text**;**

Akyi**.**searchLaw**(**textSurUrist**.**Text**).**Number **=** Convert**.**ToInt64**(**textNumber**.**Text**);**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textSurUrist**.**Text**;**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textNumber**.**Text**;**

textSurClient**.**Clear**();**textNumber**.**Clear**();**

**}**

Метод удаления юриста из компании. Очищает таблицу и удаляет юриста из класса Company.

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

**{**

Akyi**.**delLaw**();**

dataGridView2**.**Rows**.**RemoveAt**(**Akyi**.**Count**);**

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

**}**

Конопка вывода данных о всей фирме.

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

**{**

MessageBox**.**Show**(**Akyi**.**getInfoComp**());**

**}**

Метод обработки события изменения данных посетителя юриста. После валидации TextBox происходит обращение к методам Company для ввода новых значений посетителей и обновление таблицы посетителей.

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

**{**

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

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

Akyi**.**searchLaw**(**infoUrist**).**Search**(**infoClient**).**Surname **=** textSurClient**.**Text**;**

Akyi**.**searchLaw**(**infoUrist**).**Search**(**textSurClient**.**Text**).**Cost **=** Convert**.**ToInt32**(**textCost**.**Text**);**

Akyi**.**searchLaw**(**infoUrist**).**Search**(**textSurClient**.**Text**).**Service **=** textService**.**Text**;**

dataGridView1**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textSurClient**.**Text**;**

dataGridView1**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textService**.**Text**;**

dataGridView1**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**2**].**Value **=** textCost**.**Text**;**

**}**

Метод удаления посетителя из класса и из таблицы удаляется экземпляр Customers.

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

**{**

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

Akyi**.**searchLaw**(**infoUr**).**delCust**();**

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

**}**

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

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

**{**

string path**;**

string fileText **=** "Akyi Company\n"**;**

//Создание экземпляра диалогового окна (Для того чтобы не занимать поток) + фильтр на создание текстовых файлов

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

**{**

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

**{**

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

fileText **+=** Akyi**.**Info**()** **+** "\n"**;**

fileText **+=** Akyi**.**getInfoFor**();**

//Та же работа с экземплярами файла и записи файла

**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 "Akyi Company" после этого происходит дальнейшее чтение файла, заполнение структуры и таблицы данными из файла.

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

**{**

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

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

**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 **==** "Akyi Company"**)**

**{**

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

**{**

Akyi**.**delLaw**();**

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

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

**}**

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

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

**{**

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

long Number **=** long**.**Parse**(**stream3**.**ReadLine**());**

Akyi**.**addLaw**(**SurnameUrist**,** Number**,** 10**);**

dataGridView2**.**Rows**.**Add**();**

dataGridView2**.**Rows**[**i **-** 1**].**Cells**[**0**].**Value **=** SurnameUrist**;**

dataGridView2**.**Rows**[**i **-** 1**].**Cells**[**1**].**Value **=** Number**;**

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

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

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

string name **=** Convert**.**ToString**(**dataGridView2**.**Rows**[**hall**].**Cells**[**0**].**Value**);**

Akyi**.**searchLaw**(**name**).**addCust**(**SurnCli**,** Service**,** gName**);**

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

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

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

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

**}**

**}**

**}**

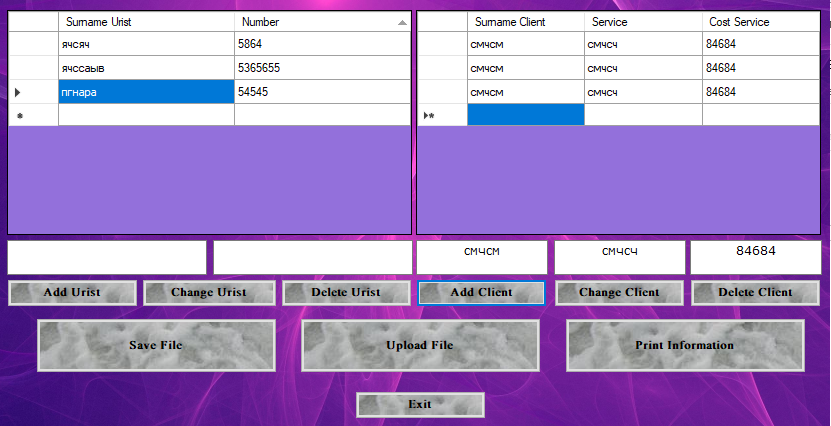


Рис. 4 Работа программы

1. Описание структуры проекта в соответствии с использованным инструментом разработки

Программа была разработана в среде разработки Visual Studio 2022. Это универсальная среда разработке позволяющая разрабатывать программы на языке программирования C# , а так же С++ и Python. IDE позволяет созывать комплексные программы, а интуитивно понятный интерфейс позволяет легко разобраться в программе и концентрироваться только на разработке.

Помимо удобного интерфейса программа позволяет тестировать программу на ресурсоёмкость, показывая программисту потребление программой ресурсов оперативной памяти и процессора.

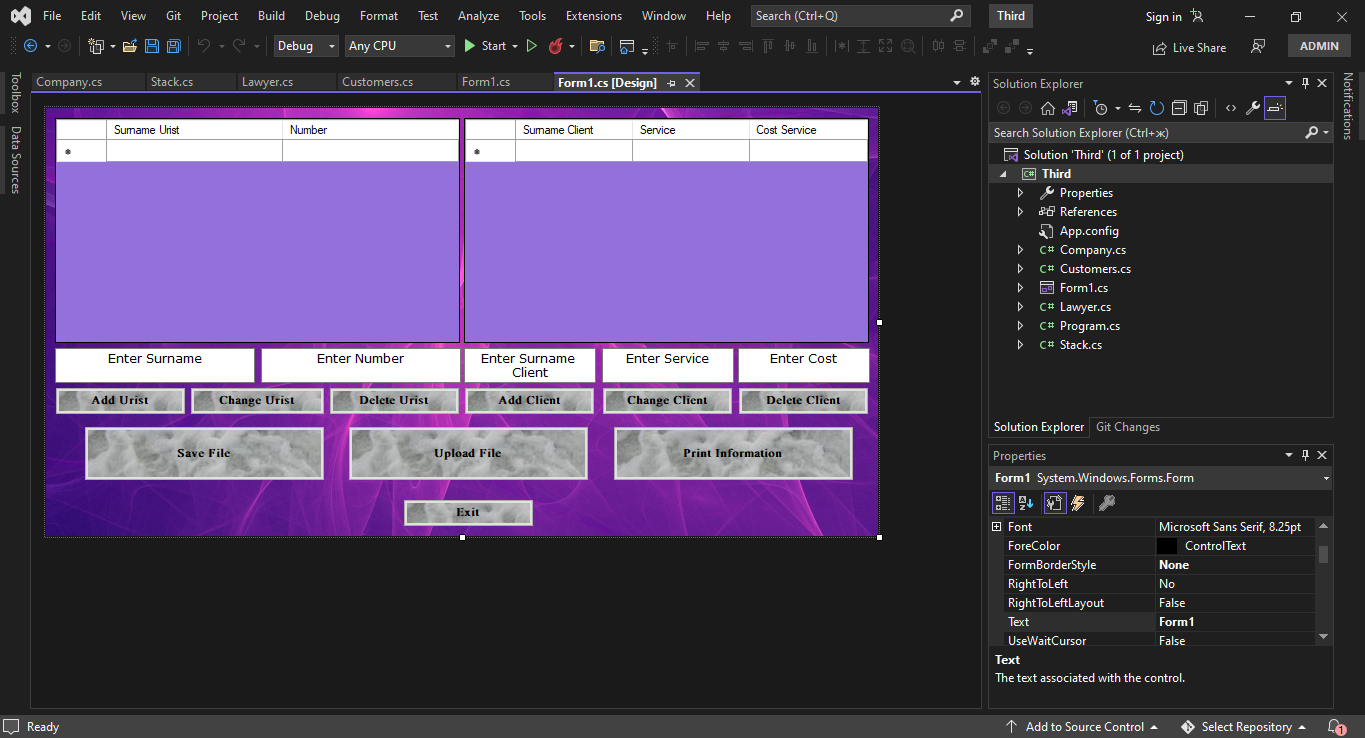


Рис. 5 Работа Visual Studio 2022

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.

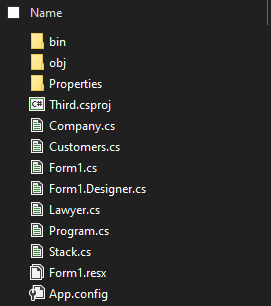


Рис. 6 Содержимое папки проекта.

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

// Form1

**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**.**Threading**.**Tasks**;**

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

**namespace** Third

**{**

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

**{**

Company Akyi **=** **new** Company**(**"Company Akyi"**);**

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

**{**

InitializeComponent**();**

**}**

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

**{**

textSurUrist**.**Clear**();**

**}**

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

**{**

textCost**.**Clear**();**

**}**

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

**{**

textNumber**.**Clear**();**

**}**

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

**{**

textSurClient**.**Clear**();**

**}**

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

**{**

textService**.**Clear**();**

**}**

**private** void dataGridView2\_CellContentClick**(object** sender**,** DataGridViewCellEventArgs e**)**

**{**

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

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

Lawyer vspom **=** Akyi**.**searchLaw**(**text**);**

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

**{**

Customers**[]** copy1 **=** vspom**.**Queue**;**

int chet **=** Akyi**.**searchLaw**(**text**).**Count**;**

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

**{**

dataGridView1**.**Rows**.**Add**(**copy1**[**i**].**Surname**,** copy1**[**i**].**Service**,**copy1**[**i**].**Cost**);**

**}**

**}**

**}**

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

**{**

**}**

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

**{**

Akyi**.**addLaw**(**textSurUrist**.**Text**,** Convert**.**ToInt64**(**textNumber**.**Text**),** 10**);**

dataGridView2**.**Rows**.**Add**(**textSurUrist**.**Text**,** textNumber**.**Text**);**

textSurUrist**.**Clear**();**textNumber**.**Clear**();**

**}**

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

**{**

Close**();**

**}**

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

**{**

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

Akyi**.**searchLaw**(**info**).**addCust**(**textSurClient**.**Text**,**textService**.**Text**,**Convert**.**ToInt32**(**textCost**.**Text**));**

dataGridView1**.**Rows**.**Add**(**textSurClient**.**Text**,** textService**.**Text**,** textCost**.**Text**);**

**}**

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

**{**

textNumber**.**MaxLength **=** 11**;**

**}**

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

**{**

// ввод в texBox только цифр и кнопки Backspace

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

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

**{**

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

**}**

**}**

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

**{**

// ввод в texBox только цифр и кнопки Backspace

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

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

**{**

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

**}**

**}**

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

**{**

textCost**.**MaxLength **=** 10**;**

**}**

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

**{**

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

Akyi**.**searchLaw**(**info**).**Surname **=** textSurUrist**.**Text**;**

Akyi**.**searchLaw**(**textSurUrist**.**Text**).**Number **=** Convert**.**ToInt64**(**textNumber**.**Text**);**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textSurUrist**.**Text**;**

dataGridView2**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textNumber**.**Text**;**

textSurClient**.**Clear**();**textNumber**.**Clear**();**

**}**

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

**{**

Akyi**.**delLaw**();**

dataGridView2**.**Rows**.**RemoveAt**(**Akyi**.**Count**);**

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

**}**

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

**{**

MessageBox**.**Show**(**Akyi**.**getInfoComp**());**

**}**

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

**{**

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

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

Akyi**.**searchLaw**(**infoUrist**).**Search**(**infoClient**).**Surname **=** textSurClient**.**Text**;**

Akyi**.**searchLaw**(**infoUrist**).**Search**(**textSurClient**.**Text**).**Cost **=** Convert**.**ToInt32**(**textCost**.**Text**);**

Akyi**.**searchLaw**(**infoUrist**).**Search**(**textSurClient**.**Text**).**Service **=** textService**.**Text**;**

dataGridView1**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**0**].**Value **=** textSurClient**.**Text**;**

dataGridView1**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**1**].**Value **=** textService**.**Text**;**

dataGridView1**.**Rows**[**dataGridView2**.**CurrentRow**.**Index**].**Cells**[**2**].**Value **=** textCost**.**Text**;**

**}**

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

**{**

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

Akyi**.**searchLaw**(**infoUr**).**delCust**();**

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

**}**

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

**{**

string path**;**

string fileText **=** "Akyi Company\n"**;**

//Создание экземпляра диалогового окна (Для того чтобы не занимать поток) + фильтр на создание текстовых файлов

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

**{**

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

**{**

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

fileText **+=** Akyi**.**Info**()** **+** "\n"**;**

fileText **+=** Akyi**.**getInfoFor**();**

//Та же работа с экземплярами файла и записи файла

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

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

**{**

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

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

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

**}**

**}**

**}**

**}**

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

**{**

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

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

**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 **==** "Akyi Company"**)**

**{**

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

**{**

Akyi**.**delLaw**();**

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

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

**}**

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

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

**{**

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

long Number **=** long**.**Parse**(**stream3**.**ReadLine**());**

Akyi**.**addLaw**(**SurnameUrist**,** Number**,** 10**);**

dataGridView2**.**Rows**.**Add**();**

dataGridView2**.**Rows**[**i **-** 1**].**Cells**[**0**].**Value **=** SurnameUrist**;**

dataGridView2**.**Rows**[**i **-** 1**].**Cells**[**1**].**Value **=** Number**;**

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

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

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

string name **=** Convert**.**ToString**(**dataGridView2**.**Rows**[**hall**].**Cells**[**0**].**Value**);**

Akyi**.**searchLaw**(**name**).**addCust**(**SurnCli**,** Service**,** gName**);**

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

i**++;**

**else**

**break;**

**}**

hall**++;**

**}**

**}**

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

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

**}**

**}**

**}**

**}**

// Сustomer

**using** System**;**

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

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

**namespace** Third

**{**

class Customers

**{**

**private** string surname**;**//фамилия заказчика

**private** string service**;**//оказанная услуга

**private** int cost**;**//стоимость услуги

**public** Customers**(**string \_surname**,** string \_service**,** int \_cost**)**//конструктор

**{**

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

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

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

**}**

**public** string Surname //get/set для фамилии

**{**

**get** **{** **return** surname**;** **}**

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

**}**

**public** string Service//get/set для оказанной услуги

**{**

**get** **{** **return** service**;** **}**

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

**}**

**public** int Cost//get/set для стоимости

**{**

**get** **{** **return** cost**;** **}**

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

**}**

**}**

**}**

// Company

**using** System**;**

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

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

**namespace** Third

**{**

class Company

**{**

**private** string nameComp**;**//название юр.фирмы

**private** Stack firstElenem**,**SP**;**//первый элемент в стеке

**private** int countElem**;**//количество элементов

**public** Company**(**string \_nameComp**)** //конструктор

**{**

**this.**nameComp **=** \_nameComp**;**

SP **=** **null;**

firstElenem **=** **null;**

countElem **=** 0**;**

**}**

**public** string Info**()**

**{**

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

Stack vspom **=** SP**;**

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

**{**

info **+=** "\n" **+** vspom**.**date**.**Surname**;**

info **+=** "\n" **+** vspom**.**date**.**Number**;**

vspom **=** vspom**.**getLaw**();**

**}**

**return** info**;**

**}**

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

**public** void addLaw**(**string \_lSurname**,** long \_number**,** int \_size**)** //добавление юриста

**{**

Lawyer dobLaw **=** **new** Lawyer**(**\_lSurname**,** \_number**,** \_size**);**

Stack help **=** **new** Stack**(**dobLaw**,** SP**);**

**if** **(**SP **==** **null)** firstElenem **=** help**;**

SP **=** help**;**

countElem**++;**

**}**

**public** void delLaw**()** //удаление юриста

**{**

**if(**SP **!=** **null)**

**{**

SP **=** SP**.**getLaw**();**

countElem**--;**

**}**

**}**

**public** Lawyer searchLaw**(**string \_lSurname**)** //поиск юриста по фамилии

**{**

Stack vspom **=** SP**;**

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

**{**

**if** **(**vspom**.**date**.**Surname**.**Equals**(**\_lSurname**))**

**{**

**return** vspom**.**date**;**

**}**

**else** vspom **=** vspom**.**getLaw**();**

**}**

**return** **null;**

**}**

**public** int SumCostLaw**()**//заработок всей компании

**{**

int summ **=** 0**;**

Stack vspom **=** SP**;**

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

**{**

summ **+=** vspom**.**date**.**SumCost**();**

vspom **=** vspom**.**getLaw**();**

**}**

**return** summ**;**

**}**

**public** string getInfoComp**()** //получение всей информации о компании

**{**

string getInfoAll **=** "\n\nName Company: " **+** nameComp**;**

Stack vspom **=** SP**;**

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

**{**

getInfoAll **+=** "\n" **+** vspom**.**date**.**getInfo**()** **+** "\n"**;**

vspom **=** vspom**.**getLaw**();**

**}**

getInfoAll **+=** "\n[summ Cost all company: " **+** SumCostLaw**()** **+** "]"**;**

**return** getInfoAll**;**

**}**

**public** string getInfoFor**()** //получение всей информации о компании

**{**

string getInfoAll **=** ""**;**

Stack vspom **=** SP**;**

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

**{**

getInfoAll **+=** "\n" **+** vspom**.**date**.**getInfoFor**();**

vspom **=** vspom**.**getLaw**();**

**}**

**return** getInfoAll**;**

**}**

**}**

**}**

// Stack

**using** System**;**

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

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

**namespace** Third

**{**

class Stack

**{**

**public** Lawyer date**;**//Данные

**private** Stack pred**;**//ссылка на предка

**public** Stack**(**Lawyer \_date**,** Stack \_SP**)**//конструктор

**{**

**this.**date **=** \_date**;**

pred **=** \_SP**;**

**}**

**public** Stack getLaw**()**//получение ссылки на предка

**{**

**return** pred**;**

**}**

**}**

**}**

// Lawyer

**using** System**;**

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

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

**namespace** Third

**{**

class Lawyer

**{**

**private** string lSurname**;**//фамилия юриста

**private** long number**;**

**private** Customers**[]** queue**;**//массив-очередь

**private** int first**;**//ссылка на 1 элемент

**private** int last**;**//ссылка на последний элемент

**private** int sumCost**,** size**,**count**;**//сумма, потраченная на услуги

**public** Lawyer**(**string \_lSurname**,** long \_number**,** int \_size**)** //конструктор

**{**

**this.**lSurname **=** \_lSurname**;**

**this.**number **=** \_number**;**

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

queue **=** **new** Customers**[**size**];**

first **=** 0**;** last **=** 0**;** count **=** 0**;**

**}**

**public** Customers Search**(**string \_lSurname**)**

**{**

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

**{**

**if(**queue**[**i**].**Surname**.**Equals**(**\_lSurname**))**

**{**

**return** queue**[**i**];**

**}**

**}**

**return** **null;**

**}**

**public** long Number **{** **get** **{** **return** number**;** **}** **set** **{** **this.**number **=** **value;** **}** **}**

**public** Customers**[]** Queue **{** **get** **{** **return** queue**;** **}** **}**

**public** string Surname**{** **get** **{** **return** lSurname**;** **}set** **{** **this.**lSurname **=** **value;** **}** **}**

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

**public** void addCust**(**string \_surname**,** string \_service**,** int \_cost**)** //добавление заказчика

**{**

**if** **(**size **>=** count**)**

**{**

**if** **(**first **>** 0 **&&** last **>=** size**)** last **=** 0**;**

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

queue**[**last**]** **=** dobav**;**

last**++;**

count**++;**

**}**

**}**

**public** void delCust**()** //удаление заказчика по переменной first

**{**

**if** **(**count **!=** 0**)**

**{**

**if** **(**first **>** size**)** first **=** 0**;**

queue**[**first**]** **=** **null;**

count**--;**

first**++;**

**}**

**}**

**public** int SumCost**()**//возвращает сумму потраченных денег

**{**

sumCost **=** 0**;**

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

**{**

**if** **(**queue**[**i**]** **!=** **null)**

**{**

sumCost **+=** queue**[**i**].**Cost**;**

**}**

**}**

**return** sumCost**;**

**}**

**public** string getInfo**()** //получение информации об одном юристе и его заказчиках

**{**

string getinfo **=** "\n<=======Lawyer: " **+** lSurname **+** "=======>"**;**

Customers help**;**

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

**{**

help **=** queue**[**i**];**

**if** **(**help **!=** **null)**

**{**

getinfo **+=** "\n\nCustomer: " **+** help**.**Surname**;**

getinfo **+=** "\nService: " **+** help**.**Service**;**

getinfo **+=** "\nCost: " **+** help**.**Cost **+** "\n"**;**

**}**

**}**

getinfo **+=** "\nsum of cost: " **+** SumCost**();**

**return** getinfo**;**

**}**

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

**{**

string getinfo **=** Convert**.**ToString**(**Count**);**

Customers help**;**

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

**{**

help **=** queue**[**i**];**

**if** **(**help **!=** **null)**

**{**

getinfo **+=** "\n" **+** help**.**Surname**;**

getinfo **+=** "\n" **+** help**.**Service**;**

getinfo **+=** "\n" **+** help**.**Cost**;**

**}**

**}**

**return** getinfo**;**

**}**

**}**

**}**