Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “ Об’єктно -орієнтоване програмування”

Відділ кадрів

Керівник професор Бонрарєв В.М.

Студент гр. ПЗПІ-19-2 Сагайдачний Н. І.

Комісія: Ст.викд. \_\_\_\_\_\_\_\_\_\_\_\_ Черепанова Ю.Ю., Проф.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В.М.,

Доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Побіженко І. О.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РАДІОЕЛЕКТРОНІКИ

Кафедра: Програмної інженерії

Дисципліна: Об’єктно-орієнтоване програмування

Спеціальність: 121 Інженерія програмного забезпечення

Освітня програма: Програмна інженерія

Курс 1 . Група ПЗПІ-19 -2 . Семестр 2 .

ЗАВДАННЯ

на курсовий проект студента

Сагайдачного Нікіти Івановича

1 Тема проекту: Відділ кадрів

2 Термін здачі студентом закінченого проекту: “19” - червня - 2020 р.

3 Вихідні дані до проекту: Специфікація програми, методичні вказівки до виконання курсової роботи.

4 Зміст розрахунково-пояснювальної записки: Вступ, специфікація програми, проектна специфікація, інструкція користувача, висновки.

5 Перелік графічного матеріалу: Схема об’єктної моделі, алгоритми, приклади екранних форм.

Студент: Сагайдачний Нікіта Іванович

Керівник: Бондарєв Володимир Михайлович

« 21 »\_лютого\_ 2020 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 45 с., 14 рис., 1 додаток.

Метою роботи є розробка програми “Відділ кадрів” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2019, Windows Forms, платформи .NET Framework 4.7.2, мови програмування C#.

В результаті отримана програма під назвою “Human Department”, яка дозволяє здобувачам реєструватися та надсилати заяви на влаштування на роботу в свою чергу формується наказ на влаштування за певною вакансією і директор або підтверджує, або скасовує його, у разі працевлаштування на основі здобувача та вакансії формується співробітник у якого є відокремлений додаток з можливістю подати заяву на звільнення.

# Оглавление

[Оглавление 4](#_Toc41858891)

[Перечень условных сокращений 5](#_Toc41858892)

[Введение 6](#_Toc41858893)

[1. Спецификация программы 8](#_Toc41858894)

[1.1. Пользователи программы 8](#_Toc41858895)

[1.2 Функции программы: 8](#_Toc41858896)

[1.2.1 Функции директора: 9](#_Toc41858897)

[1.2.2 Функции соискателей: 11](#_Toc41858898)

[1.2.3 Функции сотрудников: 12](#_Toc41858899)

[1.2.4 Функции которые доступны всем(общие): 14](#_Toc41858900)

[2 Проектная спецификация 15](#_Toc41858901)

[2.1 Архитектура 15](#_Toc41858902)

[2.2 Структура Проэкта 15](#_Toc41858903)

[2.3 Объектная Модель 16](#_Toc41858904)

[2.4 Пользовательские Типы: 17](#_Toc41858905)

[2.5 Остальные Типы: 17](#_Toc41858906)

[2.6 Методы: 17](#_Toc41858907)

[2.7 Диаграмма Классов 18](#_Toc41858908)

[2.8 Хранилище и Доступ к Данным 18](#_Toc41858909)

[3 Инструкция пользователя 20](#_Toc41858910)

[3.1 Приложение директора: 21](#_Toc41858911)

[3.2 Приложение работника: 23](#_Toc41858912)

[3.3 Приложение соискателя 24](#_Toc41858913)

[3.4 Доступ к коду проекта 25](#_Toc41858914)

[4 Тестирование 26](#_Toc41858915)

[Выводы 27](#_Toc41858916)

[Список ресурсов 28](#_Toc41858917)

[Приложение А 29](#_Toc41858918)

[Код програмы 29](#_Toc41858919)

# Перечень условных сокращений

Программирование - процесс проектирования, написания, тестирования, отладки и поддержки компьютерных программ.

ООП - объектно ориентированное программирование.

Полиморфизм - концепция в программировании и теории типов, согласно которой используется общий интерфейс для обработки различных специализированных типов.

Наследование - механизм образования новых классов на основе использования уже существующих.

Инкапсуляция - механизм, который сочетает данные и методы, обрабатывающие эти данные и защищает и то и другое от внешнего воздействия или неверного использования.

GUI - графический интерфейс.

# Введение

Целью моей задачи является создание системы для предприятия, смоделировать отдел кадров.

Задачей моей работы является создание системы с использованием объектно ориентированного подхода.

Использование объектно ориентированного подхода при разработке систем является более целесообразным.

Во-первых, это дает возможность для модификации продукта в дальнейшем.

Во-вторых, этот подход упрощает разработку программного продукта, ведь когда главные зависимости (классы) уже созданы, нужно строить взаимосвязь между ними вместо того, чтобы переписывать уже имеющий в системе текст программы.

Не забываем, что объектно ориентированная парадигма также включает набор концепций, которые мы используем в ходе разработки. Концепции, такие как:

а) Полиморфизм;

б) Наследование;

в) Инкапсуляция.

Они позволяют наладить каждый этап работы программы, как это нужно разработчику.

Такой подход даст возможность использовать созданную систему, как основу для других систем.

Хотя бы эти преимущества уже побуждают на выбор объектно ориентированного подхода, над другими

Сферой применения будем считать любое предприятие поскольку программа предоставляет доступ разным типам пользователей которые имеются на большинстве предприятий (директор, работники, соискатели).

В основном программа нацелена на взаимодействие директора с отделом кадров(решение директора по приказам), но работники и соискатели также имеют необходимые возможности.

# Спецификация программы

## Пользователи программы

У проекта имеются три категории пользователей, которым необходимы различные функции программы следовательно это:

- Соискатели пытаются устроиться на работу заполняют необходимые данные для проверки наличия соответственных вакансий.

- Директор у которого есть доступ ко всем разветвлениям программы(имеет возможность принимать сотрудников на работу увольнять по собственным рассуждениям )приказы без подтверждения(подписи) директора являются недействительными, следовательно задача директора работа с документами в частности подпись приказов об увольнении и зачислении.

- Сотрудники фирмы (имеют возможность мониторить свои данные такие как оклад и тд. так же увольняться по собственному желанию)

## Функции программы:

Программа представляет соискателям следующие возможности:

1. Регистрация

2. Вход в программу

3. Заполнение личных данных

4. Поиск вакансий на данном предприятии

Программа представляет директору следующие возможности:

1. Вход в программу.

2. Подтверждение(подпись) приказов об увольнении и зачислении на работу сотрудников.

3. Личное составление приказов об увольнении и зачислении на работу сотрудников(вне зависимости от всех необходимых критериев, то есть за собственными рассуждениями).

Программа представляет сотрудникам следующие возможности:

1. Вход в программу.

2. Самовольно увольняться(заявление об увольнении по собственному желанию ).

3. Отслеживать данные об изменении зарплаты за определенный период.

Общие функции:

1. Просмотр статистика предприятия(количество сотрудников, и вакансий на предприятии).

2. Загрузка данных из постоянной памяти в оперативную.

3. Сохранение текущего состояния данных в постоянной памяти.

4. Генерация тестовых данных.

### Функции директора:

Функция «Вход в программу»

Директор входит в программу при помощи личных данных (имя + фамилия) и пароля.

Личных данные и пароль директора сохраняются в данных программы.

Функции директора должны быть скрыты от основных пользователей программы – сотрудников и соискателей. Унификация проходит на этапе входа при совпадении логина(имя + фамилия) и пароля открывается соответствующее приложение.

Основной сценарий

1. Директор вводит имя и пароль и нажимает кнопку «Login». (рис.1).

2. Введенные данные проверяются и открывается главное окно директора(рис.2).

3. Если данные не прошли проверку, на форме ввода появляется сообщение об этом и можно попытаться войти снова(количество попыток не ограничено).

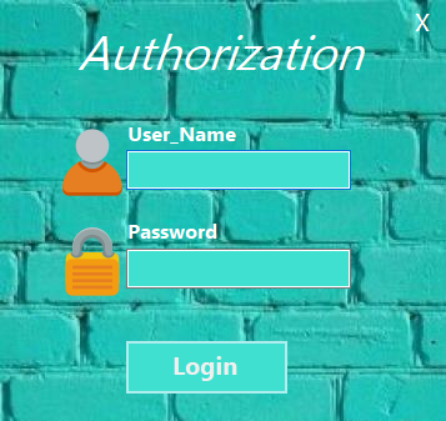


Рисунок 1.1 – Панель входа .

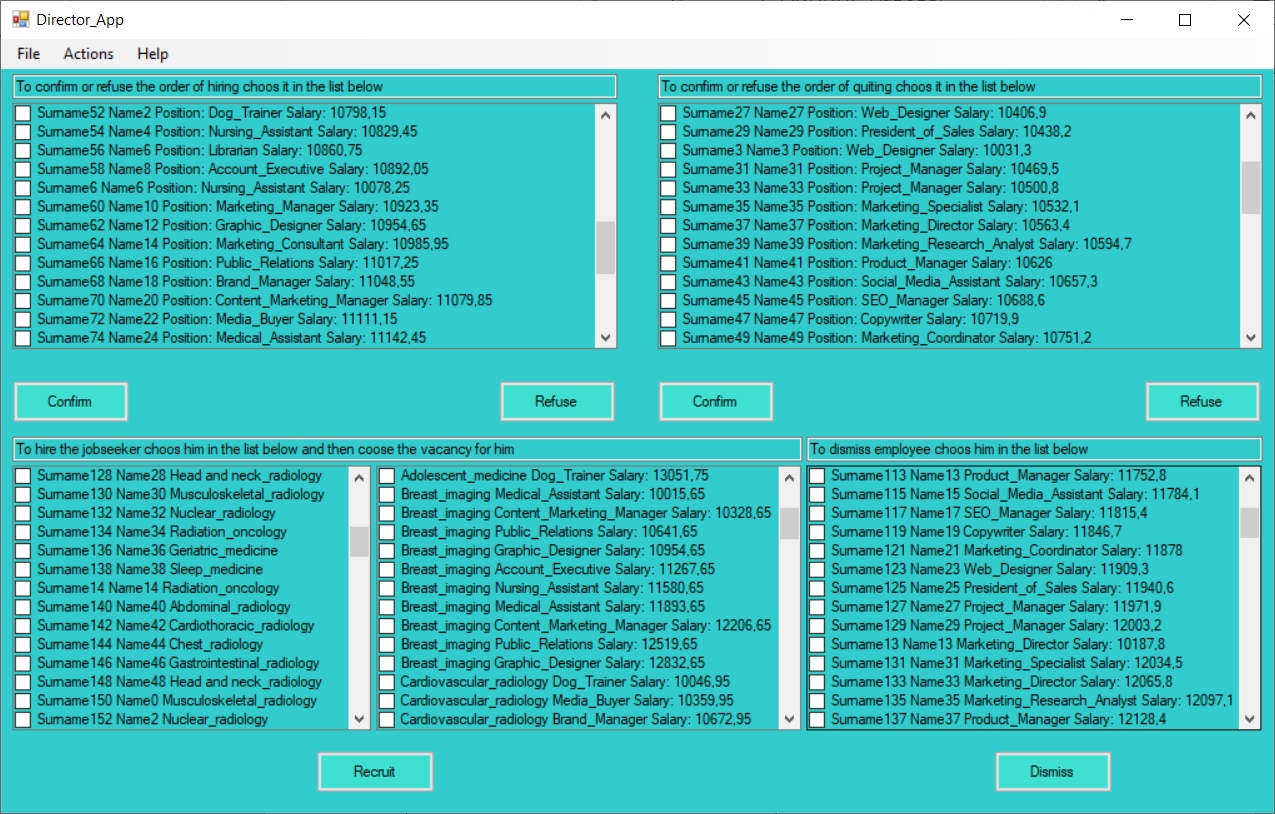


Рисунок 1.2 – Рабочее меню директора

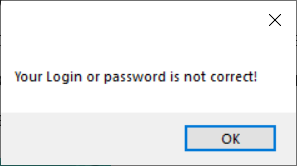


Рисунок 1.3 – Вывод ошибки о неправильно введенных данных.

Функция «Подтверждение(подпись) приказов об увольнении и зачислении на работу сотрудников»

На рабочем меню директора имеется список приказов требующих подтверждения (рис.2) по нажатию на которые имеется данная возможность.

Функция «Составление нового приказа»

На рабочем меню директора имеется кнопка (составить новый приказ о зачислении/увольнении(Recruit/Dismiss)), нажав на которую выведется окно подтверждения действия.

### Функции соискателей:

Функция «Регистрации»

В случае если пользователь не зарегистрирован ему предоставляется такая возможность по нажатию на соответствующую кнопку.

После чего пользователь должен ввести необходимые данные и придумать пароль.

Функция «Вход в программу»

Соискатели входят в программу при помощи личных данных(имя + фамилия) и пароля.

Личные данные и пароли соискателей сохраняются в данных программы.

Вход соискателей отражается на стартовой странице

Основной сценарий

1. Соискатель вводит личные данные(имя + фамилия) и пароль и нажимает кнопку «Login»(рис.1).

2. Введенные данные проверяются и открывается окно соискателя(рис.4).

3. Если данные не прошли проверку, на форме ввода появляется сообщение об этом и можно попытаться войти снова(количество попыток не ограничено)( Рисунок 3).

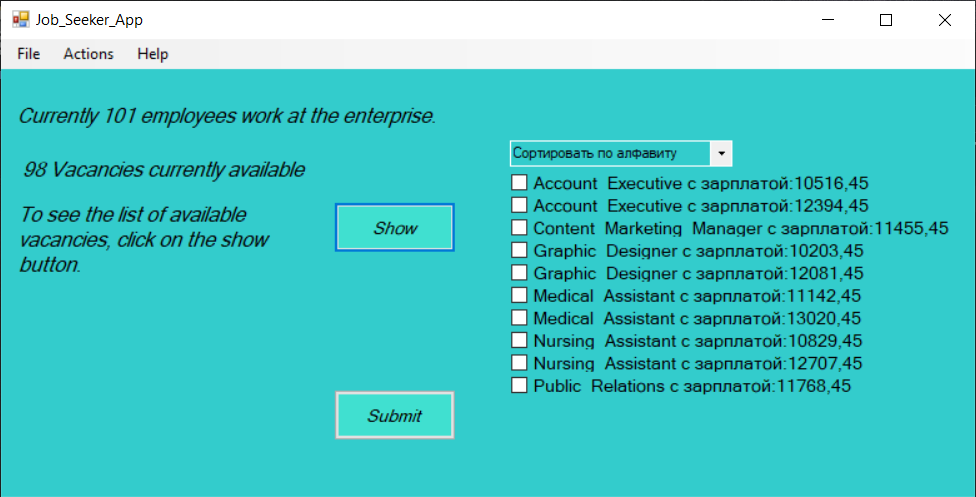


Рисунок 1.4 – окно соискателя

Функция «Поиска вакансий»

Соискателю выводиться список вакансий.

Если его все устраивает то он имеет возможность заявить об этом и после чего будет составлен соответствующий приказ.

### Функции сотрудников:

Функция «Вход в программу»

Сотрудники входят в программу при помощи личных данные (имя + фамилия) и пароля.

Личных данные и пароли сотрудников сохраняются в данных программы.

Вход сотрудников отражается на стартовой странице

Основной сценарий

1. Сотрудник вводит имя и пароль и нажимает кнопку «Login»(рис.1).

2. Введенные данные проверяются и открывается окно сотрудника(рис.5).

3. Если данные не прошли проверку, на форме ввода появляется сообщение об этом и можно попытаться войти снова(количество попыток не ограничено)(рисунок 3).

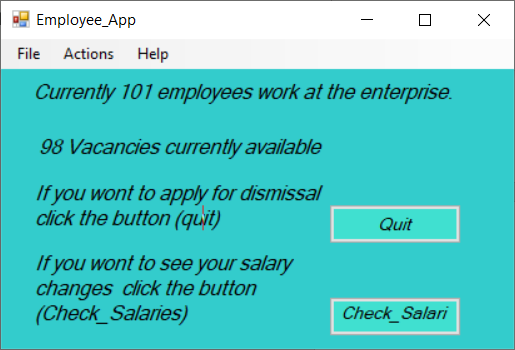


Рисунок 1.5 – окно работника

Функция «Отслеживания изменений в зарплате»

По нажатию на соответствующую кнопку сотрудник может видеть изменения своей зарплаты за весь период работы на фирме.

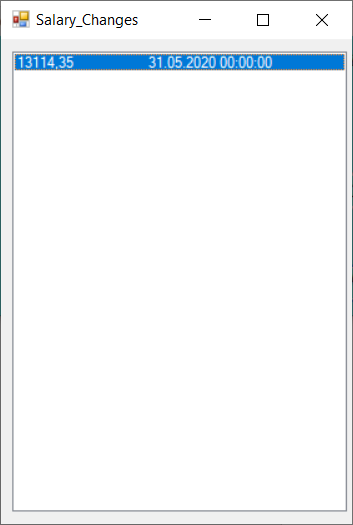


Рисунок 1.6 – список изменения зарплаты сотрудника.

Функция «Увольнения»

По нажатию на соответствующую кнопку сотрудник заполняет необходимые поля и после чего формируется заявление об увольнении которое сотрудник должен отправить используя соответствующую кнопку тем самым подтвердив свои намерения.

### Функции которые доступны всем(общие):

На главной странице все пользователи имеют возможность просматривать статистику предприятия(количество сотрудников и вакансий).

Функция «Выход из программы»

В процессе работы пользователь изменяет данные, которые находятся в оперативной памяти.

Если данные изменены, но не сохранены в постоянной памяти, то перед закрытием программы пользователь получает предложение сохранить данные.

# Проектная спецификация

## Архитектура

Программа будет создаваться на языне С# с использованием среды разработки Microsoft Visual Studio 2019 на платформе .NET Framework с исплользованием интерфейса програмирования приложений Windows Forms.

Выбор языка обусловлен объектно-ориентированным подходом к написанию программы. В процессе проектирования рассматривались 2 платформы, а именно .NET Core и .NET Framework.

Hаш выбор пал на последнюю по причине большего количества управляющих элементов и более высокого уровня развития платформы. К сожалению, это означает что приложение будет работать исключительно на машинах с операционной системой Windows.

Для реализации пользовательского интерфейса было решено использовать GUI так как он простой для восприятия и интуитивно понятен. Для реализации интерфейса мы будем использовать интерфейс Windows Forms который является частью платформы NET Framework. C eгo помощью у нас будет возможность создать полноценный интерфейс для пользователей программы.

## Структура Проэкта

Для разумной реализации программы было принято решение разделить на три отдельные части: для работников, соискателей и директора, поскольку требуется унифицировать каждый тип пользователей и разделить их возможности.

Хранение данных будет происходить на компьютере пользователя.

## Объектная Модель

*Программа представляет соискателям следующие возможности:*

1. Регистрация

2. Вход в программу

3. Заполнение личных данных

4. Поиск вакансий на данном предприятии

*Программа представляет директору следующие возможности:*

1. Вход в программу.
2. Подтверждение(подпись) приказов об увольнении и зачислении на работу сотрудников.
3. Личное составление приказов об увольнении и зачислении на работу сотрудников(вне зависимости от всех необходимых критериев, то есть за собственными рассуждениями).

*Программа представляет сотрудникам следующие возможности:*

1. Вход в программу.
2. Самовольно увольняться(заявление об увольнении по собственному желанию ).
3. Отслеживать данные об изменении зарплаты за определенный период.

Общие функции:

1. Просмотр статистики предприятия(количество сотрудников и доступных вакансий).
2. Загрузка данных из постоянной памяти в оперативную.
3. Сохранение текущего состояния данных в постоянной памяти.
4. Генерация тестовых данных.

## Пользовательские Типы:

-Пользователь – логин + пароль + личная информация(указанная при регистрации).

-Приказы об увольнении/зачислении с/на работу – подтверждение(подпись директора) + личная информация того кто подал заявление + дата + вакансия (по которой соискатель устраивается на работу, либо которая станет доступной после увольнения сотрудника ).

## Остальные Типы:

-Коллекция вакансий.

-Коллекция сотрудников.

-Коллекция соискателей.

-Коллекция приказов.

-Предприятие – это коллекция сотрудников + данные о директоре + коллекция вакансий + коллекция соискателей + приказы.

## Методы:

Методы:

1. Регистрация новых пользователей.
2. Подтверждение(подпись) приказов директором.
3. Подбор вакансий.

Более подробное описание методов находиться в спецификации к проекту.

## Диаграмма Классов

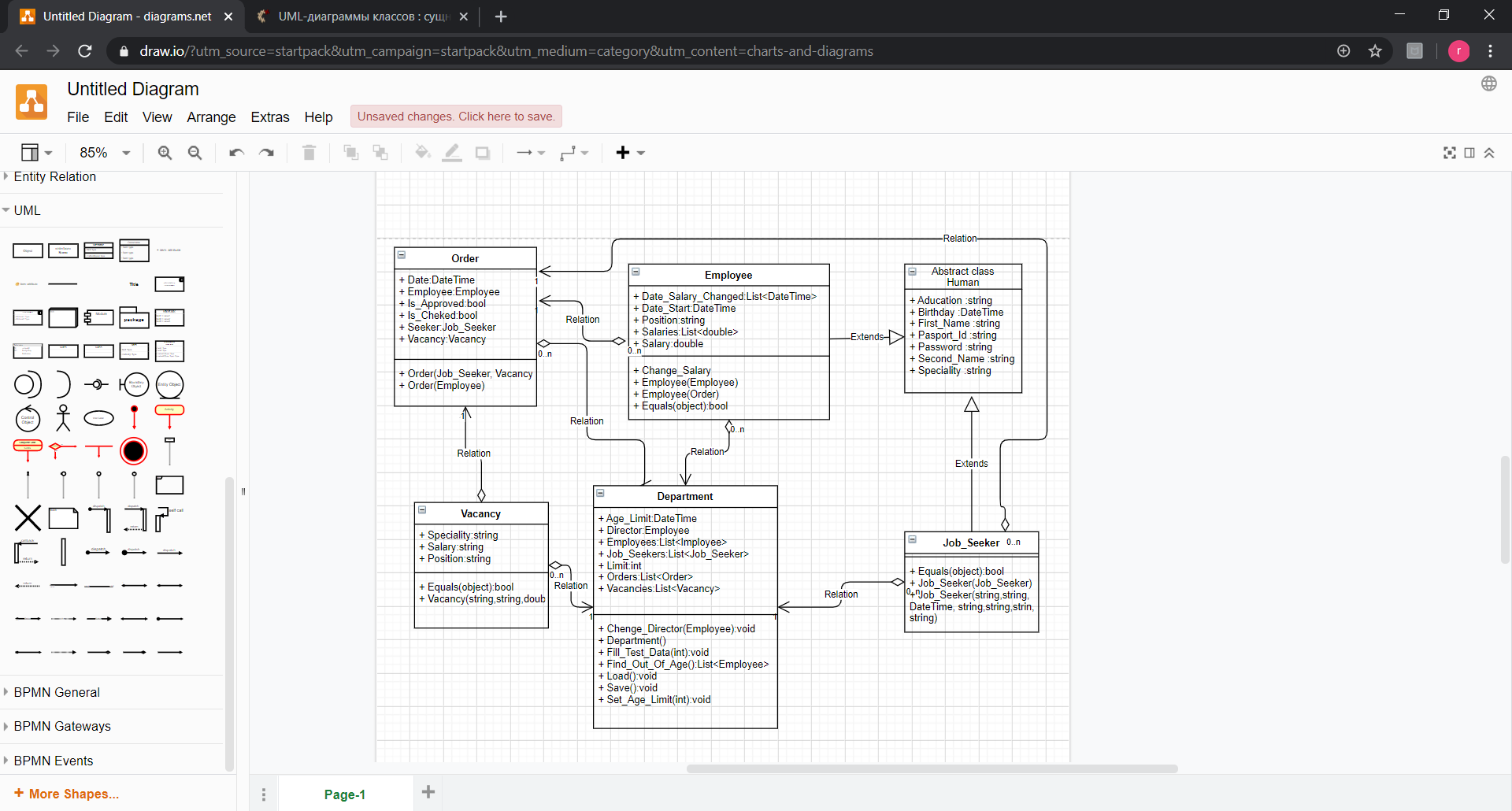


Рисунок 2.1 диаграмма классов

## Хранилище и Доступ к Данным

Данные хранятся в отдельных бинарных файлах в каталоге проекта.

Из текстовых в бинарные данные будут переводиться с помощью сериализации.

Защита данных не предусматривается. Также существует текстовые файлы для генерации тестовых данных.

# Инструкция пользователя

Проект доступен по ссылке: <https://github.com/SAGAaga/MyKursuch>

Для запуска достаточно скачать папку Start\_Human\_Department открыть ее и запустить файл с расширением .exe после чего вы попадаете на меню выбора (рис.3.1) либо регистрации нового пользователя ( соискателя ) (рис.3.2), либо входа в программу(рис.3.3).

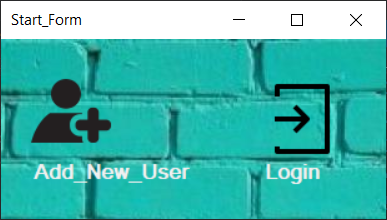


Рисунок 3.1 стартовая форма.

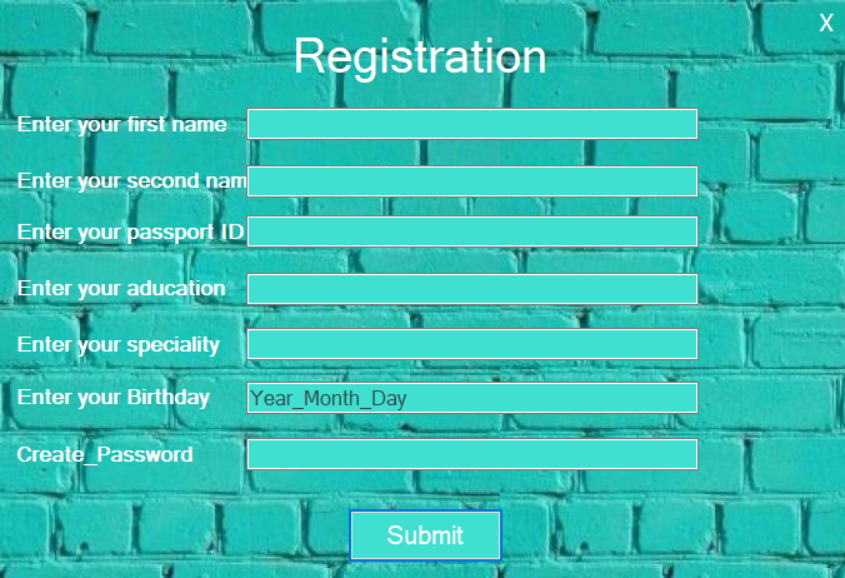


Рисунок 3.2 форма регистрации.

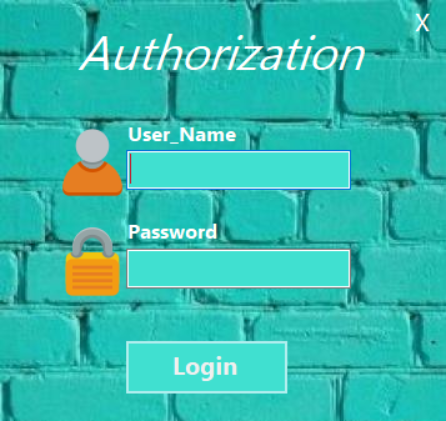


Рисунок 3.3 форма входа.

Дальше в зависимости от введенных данных(логином является либо имя + фамилия, либо фамилия + имя, записанные через пробел) вы попадаете в приложение того пользователя кем вы являетесь (соискатель, работник, директор).

## Приложение директора:

После введения логина и пароля директора вы попадаете в главное приложение директора(рис.3.4), на котором расположены различные списки с возможностью выбора:

-списки работников(возможность уволить любого для этого выбираем работника после чего появляется соответствующая кнопка ”Dismiss” по нажатию на которую будет выведено диалоговое окно которое ждет подтверждение действий);

-списки соискателей и доступных вакансий (возможность нанять нового работника также необходимо отметить соискателя а затем вакансию после чего появляется кнопка “Recruit” также по нажатию появиться диалоговое окно для подтверждения действий)

-два списка приказов с идентичными возможностями но разбитые на отдельные части поскольку один – список приказов об увольнении, а другой – об зачислении на работу. Также как и в предыдущих “списках выбора” по отметки приказа появиться две кнопки “Confirm”, “Refuse”, что в свою очередь подтвердить или отказать выполнение приказа. По нажатию появляется диалоговое окно подтверждения действий.

В верхнем меню имеется 3 пункта:

-File(по нажатию выпадает меню с возможностью сохранить изменения данных либо закрыть приложение)

-Actions(по нажатию выпадает список по которому доступны все действия как и с главной формы + возможность изменить пользователя (вы покидаете приложение и переходите к стартовой форме)).

-Help(по нажатию появляется пункт “About us” по нажатию на который открывается новая форма с краткой информацией о предприятии).

Если были произведены какие-либо действия по выходу программа с спросит у вас хотите ли вы сохранить измененные данные.

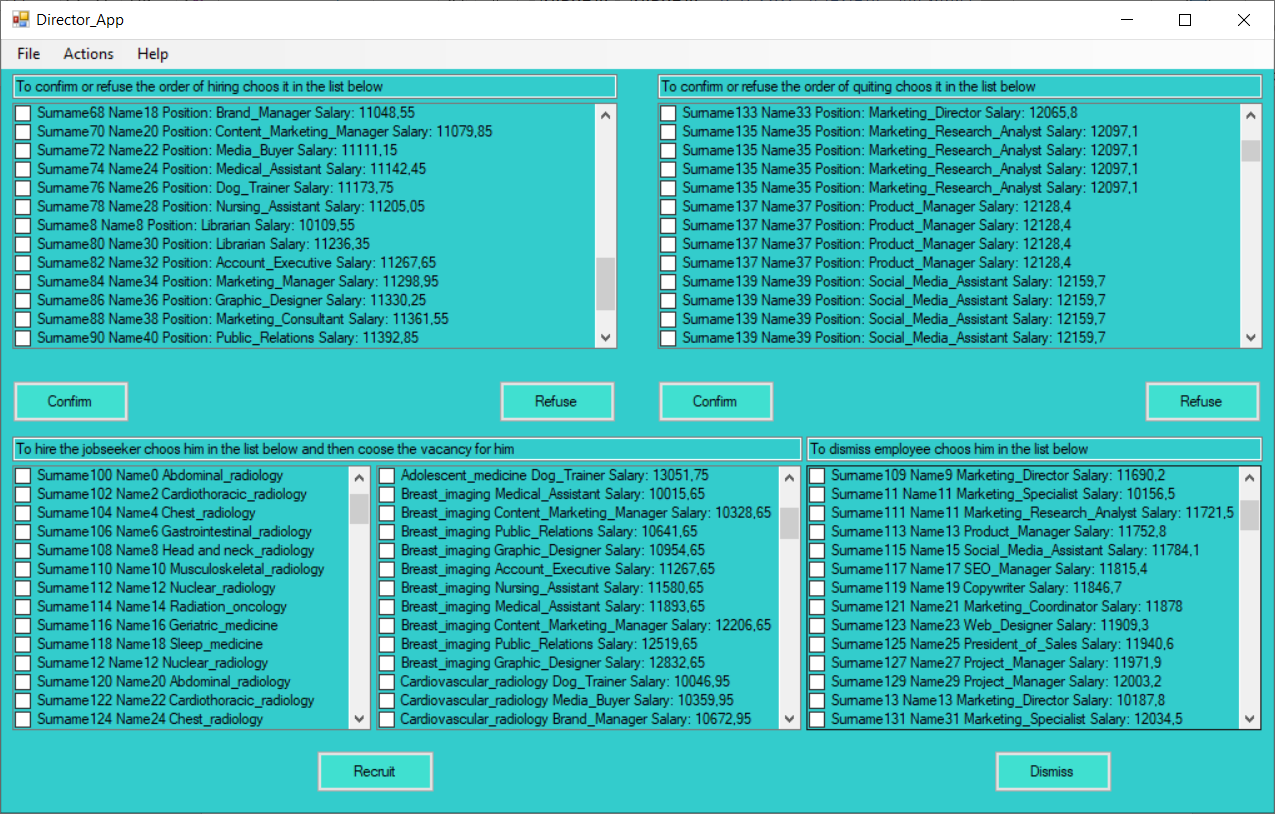


Рисунок 3.4 приложение директора.

## Приложение работника:

После введения логина и пароля работника вы попадаете в главное приложение работника(рис.3.5), на котором расположены краткие данные о предприятии и две возможности действий (просмотр изменения зарплаты(рис.3.6), составление приказа об увольнение по нажатию на кнопу”Quit” (появляется диалоговое окно которое ждет подтверждения)).

В верхнем меню имеется 3 пункта:

-File(по нажатию выпадает меню с возможностью сохранить изменения данных либо закрыть приложение)

-Actions(по нажатию выпадает список по которому доступны все действия как и с главной формы + возможность изменить пользователя (вы покидаете приложение и переходите к стартовой форме)).

-Help(по нажатию появляется пункт “About us” по нажатию на который открывается новая форма с краткой информацией о предприятии).

Если были произведены какие-либо действия по выходу программа с спросит у вас хотите ли вы сохранить измененные данные.

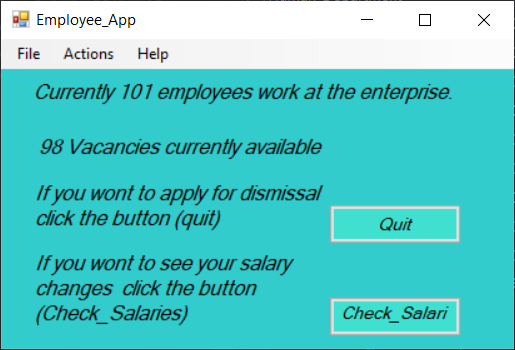


Рисунок 3.5 приложение работника.

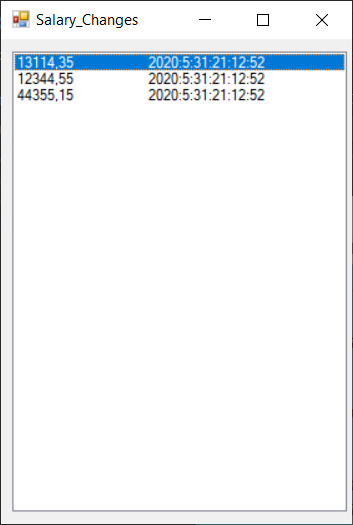


Рисунок 3.6 изменения зарплаты(поскольку тестовые данные заполняются одномоментно то дата получается одинаковая)

## Приложение соискателя

После введения логина и пароля соискателя вы попадаете в главное приложение соискателя(рис.3.7), на котором расположены краткие данные о предприятии возможность (отправлять заявления на работу по выбору доступных вакансий, которые могут быть отсортированы как по алфавиту так и по возрастанию заработной платы которую предлагает данная вакансия).

В верхнем меню имеется 3 пункта:

-File(по нажатию выпадает меню с возможностью сохранить изменения данных либо закрыть приложение)

-Actions(по нажатию выпадает список по которому доступны все действия как и с главной формы + возможность изменить пользователя (вы покидаете приложение и переходите к стартовой форме)).

-Help(по нажатию появляется пункт “About us” по нажатию на который открывается новая форма с краткой информацией о предприятии).

Если были произведены какие-либо действия по выходу программа с спросит у вас хотите ли вы сохранить измененные данные.

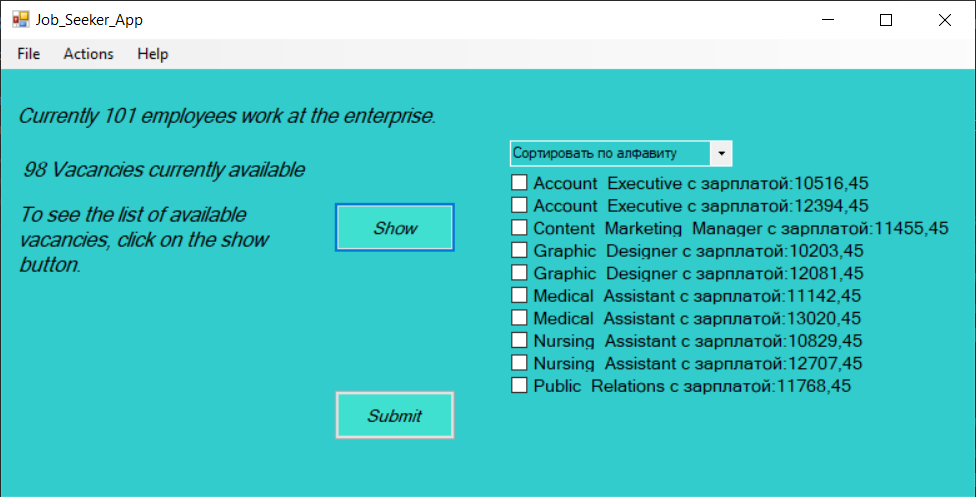


Рисунок 3.7 приложение соискателя

## Доступ к коду проекта

Код проекта доступен для скачивания по ссылке: <https://github.com/SAGAaga/MyKursuch>

Необходимо скачать весь проект, после первого запуска необходимо переместить в папку там где находить фай с разрешение exe(скорее всего это будет Human\_Department/ bin/ Debug) переместить и заменить 3 файла (“Speciality.txt”, “ Position.txt”, “ store.bin”). При следующем запуске программа уже заполниться тестовыми данными для доступа ко всем разветвлениям программы предлагаю следующие логины и соответствующие пароли:

1. Директор логин: DirectorName DirectorSurname пароль: Director
2. Соискатель логин: JobseekerName JobseekerSurname пароль: Jobseeker
3. Работник логин: EmployeeName EmployeeSurname пароль: Employee

# Тестирование

Каждый продукт, который выходит к релизу должен быть тщательно протестированным и выдавать достоверную информацию. Чтобы убедиться в правильности было проведено отладку программы. При запуске кода синтаксических ошибок не было обнаружено. Также был создан отдельный тестирующий класс в котором было проверено правильность заполнения тестовыми данными.

Была проведена верификация на установление правильности кода, ошибок не было обнаружено код выполнял все необходимые задачи, поставленные в работе.

Также была проведена валидация программа была запущена и сверены все требования, ошибок найдено не было.

При проведении статического метода тестирования «инспекции» на всех этапах ЖЦ было обнаружено что не срабатывает один метод программы (метод который срабатывает при выходе из программы и спрашивает у пользователя сохранить ли измененные данные), проблема была быстро исправлена (при изменении данных мен не помечали что было проведено изменение) и именно здесь проявились достаток данного типа проверки поскольку ошибки оказываются на ранних этапах проектирования

# Выводы

Была разработана система, которая представляет виртуальные отдел кадров предприятия. Было использовано объектно ориентированный подход к решению задачи. Была установлена ​​цель и задача работы, которым мы следовали при разработке системы. Также, нами был использован принцип наследования, инкапсуляции.

Было проведено тестирование на правильность исходных данных программы. Выводы показали, что программа работает правильно и без ошибок выполняет все необходимые действия.

Проанализировав текущий функционал системы, определили возможный список модификаций программы для использования системы, как основной и независимой от других. Анализ показал, что модификация возможна в различных аспектах от дополнения новым функционалом к взаимодействию с удаленными носителями информации.

Как итог, выделим, что система выполняет представленный нами функционал и полностью готова для использования на предприятии.

При разработке было усвоено алгоритм решения проблем, возникающих при разработке, принципы ООП и навыки в разработке систем.

# Список ресурсов

1. MICROSOFT DOCS [Электронный ресурс] режим доступа к ресурсу –<https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/>
2. METANIT [Электронный ресурс] режим доступа к ресурсу – <https://metanit.com/sharp/tutorial/2.12.php>
3. STACKOVERFLOW DOCS [Электронный ресурс] режим доступа к ресурсу – <https://ru.stackoverflow.com/>
4. PROFESSOR WEB [Электронный ресурс] режим доступа к ресурсу – <https://professorweb.ru/my/csharp/charp_theory/level1/index.php>
5. Методические указания к выполнению курсовой работы по дисциплине "объектно-ориентированное программирование" для студентов первого (бакалаврской) уровня высшего образования всех форм обучения 121 - «Инженерия программного обеспечения» образовательно-профессиональная программа «Программная инженерия» Составитель В. М. Бондарев, Ю.Ю. Черепанова Харьков: ХНУРЭ, 2018.

# Приложение А

## Код програмы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Human\_Department.Models

{

[Serializable]

public class Department

{

public List<Employee> Employees { private set; get; }

public List<Job\_Seeker> Job\_Seekers { private set; get; }

public List<Vacancy> Vacancies { private set; get; }

public List<Order> Orders { private set; get; }

public Employee Director { private set; get; }

public DateTime Age\_Limit { private set; get; }

public int Limit { private set; get; }

public Department()

{

Limit = 65;

DateTime now = DateTime.Now;

Age\_Limit = now.AddYears(-Limit);

Employees = new List<Employee>();

Job\_Seekers = new List<Job\_Seeker>();

Vacancies = new List<Vacancy>();

Orders = new List<Order>();

Load();

List<Employee> out\_of\_age = Find\_Out\_Of\_Age();

foreach (var t in out\_of\_age)

{

Orders.Add(new Order(t));

}

Fill\_Test\_Data(100);

}

public void Set\_Age\_Limit(int years)

{

Limit = years;

DateTime now = DateTime.Now;

Age\_Limit = now.AddYears(-Limit);

}

public void Save()

{

new Dao(this).Save();

}

public void Load()

{

new Dao(this).Load();

}

public void Chenge\_Director(Employee worker)

{

Director = worker;

Employees.Remove(worker);

}

public List<Employee> Find\_Out\_Of\_Age() =>

new List<Employee>(Employees.Where(x => x.Birthday < Age\_Limit));

public void Fill\_Test\_Data(int n)

{

List<string> pos = new List<string>();

try

{

using (StreamReader sr = new StreamReader("Position.txt",

System.Text.Encoding.Default))

{

string line;

while ((line = sr.ReadLine()) != null)

{

pos.Add(line);

}

}

}

catch (System.IO.FileNotFoundException)

{

File.Create("Position.txt");

}

List<string> spec = new List<string>();

try

{

using (StreamReader sr = new StreamReader("Speciality.txt",

System.Text.Encoding.Default))

{

string line;

while ((line = sr.ReadLine()) != null)

{

spec.Add(line);

}

}

}

catch (System.IO.FileNotFoundException)

{

File.Create("Speciality.txt");

}

if (pos.Count != 0 && spec.Count != 0)

{

//Vacancies

if (Vacancies.Count == 0)

{

Vacancies.Clear();

for (int i = 0; i < n \* 2; ++i)

{

Vacancies.Add(new Vacancy(pos[i % pos.Count],

spec[i % spec.Count], 10000 + (i \* 15.65)));

}

}

//Job Seekers

if (Job\_Seekers.Count == 0)

{

Job\_Seekers.Clear();

for (int i = 1; i <= n \* 2; ++i)

{

Job\_Seekers.Add(new Job\_Seeker($"Name{i % 50}", $"Surname{i % 200}",

DateTime.Now - TimeSpan.FromDays(7000) - TimeSpan.FromDays(i \* i),

"" + Convert.ToChar(i % 120) + $"{i \* 8}", "Higher",

spec[i % spec.Count], $"{i \* 100 + (i + 3) \* 10 + i}"));

}

}

//Employees

if (Employees.Count == 0)

{

Employees.Clear();

for (int i = 0; i < n; ++i)

{

Employees.Add(new Employee(Job\_Seekers[i], Vacancies[i]));

Job\_Seekers.Remove(Job\_Seekers[i]);

Vacancies.Remove(Vacancies[i]);

}

}

//Orders

if (Orders.Count == 0)

{

Orders.Clear();

for (int i = 0; i < n / 2; ++i)

{

Orders.Add(new Order(Employees[i]));

Orders.Add(new Order(Job\_Seekers[i], Vacancies[i]));

}

}

if (Director == null)

{

//Director

//Создадим по 1 виду пользователя с более конкретнымми данными

//для авторизации и проверки работоспособности приложения

Employees.Add(new Employee(new Job\_Seeker("EmployeeName",

"EmployeeSurname", DateTime.Now - TimeSpan.FromDays(7000),

"ANK1254", "Hight", "Loyer", "Employee"), Vacancies.Last()));

Employees.Last().Change\_Salary(12344.55);

Vacancies.Remove(Vacancies.Last());

Employees.Add(new Employee(new Job\_Seeker("DirectorName",

"DirectorSurname", DateTime.Now - TimeSpan.FromDays(8000),

"SDK9054", "Hight", "Loyer", "Director"), Vacancies.Last()));

Vacancies.Remove(Vacancies.Last());

Chenge\_Director(Employees.Last());

Job\_Seekers.Add(new Job\_Seeker("JobseekerName",

"JobseekerSurname", DateTime.Now - TimeSpan.FromDays(10000),

"RTG1569", "Hight", "Pediatric\_radiology", "Jobseeker"));

Employees.Last().Change\_Salary(44355.15);

}

Save();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Human\_Department.Models

{

[Serializable]

public class Employee: Human

{

public string Position { private set; get; } //должность

public double Salary { private set; get; }

public List<double> Salaries { private set; get; }

public List<DateTime> Date\_Salary\_Changed { private set; get; }

public DateTime Date\_Start { private set; get; }

public Employee(Job\_Seeker hum, Vacancy vac)

{

Salaries = new List<double>();

Date\_Salary\_Changed = new List<DateTime>();

this.Pasport\_Id = hum.Pasport\_Id;

this.First\_Name = hum.First\_Name;

this.Second\_Name = hum.Second\_Name;

this.Aducation = hum.Aducation;

this.Speciality = hum.Speciality;

this.Birthday = hum.Birthday;

this.Password = hum.Password;

this.Position = vac.Position;

this.Salary = vac.Salary;

Change\_Salary(this.Salary);

this.Date\_Start = DateTime.Now;

}

public Employee(Order or)

{

Salaries = new List<double>();

Date\_Salary\_Changed = new List<DateTime>();

this.Pasport\_Id = or.Seeker.Pasport\_Id;

this.First\_Name = or.Seeker.First\_Name;

this.Second\_Name = or.Seeker.Second\_Name;

this.Aducation = or.Seeker.Aducation;

this.Speciality = or.Seeker.Speciality;

this.Birthday = or.Seeker.Birthday;

this.Password = or.Seeker.Password;

this.Position = or.Vacancy.Position;

this.Salary = or.Vacancy.Salary;

Change\_Salary(this.Salary);

this.Date\_Start = DateTime.Now;

}

public Employee(Employee emp)

{

Salaries = new List<double>();

Date\_Salary\_Changed = new List<DateTime>();

Salaries.AddRange(emp.Salaries);

Date\_Salary\_Changed.AddRange(emp.Date\_Salary\_Changed);

this.Pasport\_Id = emp.Pasport\_Id;

this.First\_Name = emp.First\_Name;

this.Second\_Name = emp.Second\_Name;

this.Aducation = emp.Aducation;

this.Speciality = emp.Speciality;

this.Birthday = emp.Birthday;

this.Password = emp.Password;

this.Position = emp.Position;

this.Salary = emp.Salary;

this.Salary = emp.Salary;

this.Date\_Start = emp.Date\_Start;

}

public void Change\_Salary(double sal)

{

Salaries.Add(sal);

Date\_Salary\_Changed.Add(DateTime.Now);

}

public override bool Equals(object obj)

{

if (obj == null || this.GetType() != obj.GetType())

{

return false;

}

Employee h = (Employee)obj;

if (Pasport\_Id == h.Pasport\_Id &&

First\_Name == h.First\_Name &&

Second\_Name == h.Second\_Name &&

Birthday == h.Birthday &&

Password == h.Password)

{

return true;

}

else

{

return false;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Human\_Department.Models

{

[Serializable]

public abstract class Human

{

public string Pasport\_Id { protected set; get;}

public string First\_Name { protected set; get; }

public string Second\_Name { protected set; get; }

public DateTime Birthday { protected set; get; }

public string Aducation { protected set; get; }

public string Password { protected set; get; }

public string Speciality { protected set; get; } //специальность

public override bool Equals(object obj)

{

if(!base.Equals(obj))

{

return false;

}

Human h = (Human)obj;

if(Pasport\_Id==h.Pasport\_Id &&

First\_Name==h.First\_Name &&

Second\_Name==h.Second\_Name &&

Birthday==h.Birthday &&

Password == h.Password)

{

return true;

}

else

{

return false;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Human\_Department.Models

{

[Serializable]

public class Job\_Seeker:Human

{

public Job\_Seeker(string fn, string sn, DateTime Birth, string number, string adu, string spec ,string pw)

{

Pasport\_Id = number;

First\_Name = fn;

Second\_Name = sn;

Aducation = adu;

Speciality = spec;

Birthday = Birth;

Password = pw;

}

public Job\_Seeker(Job\_Seeker seeker)

{

Pasport\_Id = seeker.Pasport\_Id; ;

First\_Name = seeker.First\_Name;

Second\_Name = seeker.Second\_Name;

Aducation = seeker.Aducation;

Speciality = seeker.Speciality;

Birthday = seeker.Birthday;

Password = seeker.Password;

}

public override bool Equals(object obj)

{

if (obj == null || this.GetType() != obj.GetType())

{

return false;

}

Job\_Seeker h = (Job\_Seeker)obj;

if (Pasport\_Id == h.Pasport\_Id &&

First\_Name == h.First\_Name &&

Second\_Name == h.Second\_Name &&

Birthday == h.Birthday &&

Password == h.Password)

{

return true;

}

else

{

return false;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Human\_Department.Models

{

[Serializable]

public class Order

{

public DateTime Date { private set; get; }

public Job\_Seeker Seeker { private set; get; }

public Employee Employee { private set; get; }

public Vacancy Vacancy { private set; get; }

public bool Is\_Approved { set; get; } //решение директора о приказе

public bool Is\_Cheked { set; get; } //был ли приказ просмотрен

//и было ли принято решение

public Order(Job\_Seeker seeker,Vacancy vac)

{

this.Seeker =new Job\_Seeker (seeker);

Vacancy = vac;

Is\_Approved = false;

Is\_Cheked = false;

Employee = null;

Date = DateTime.Now;

}

public Order(Employee employee)

{

this.Employee = new Employee(employee);

Is\_Approved = false;

Is\_Cheked = false;

Vacancy = new Vacancy(employee.Position,employee.Speciality, employee.Salary);

Seeker = null;

Date = DateTime.Now;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Human\_Department.Models

{

[Serializable]

public class Vacancy

{

public string Position { set; get; }

public string Speciality { set; get; }

public double Salary{ set; get; }

public Vacancy(string pos,string spec,double sal)

{

Position = pos;

Salary = sal;

Speciality = spec;

}

public override bool Equals(object obj)

{

if (obj == null || this.GetType() != obj.GetType())

{

return false;

}

Vacancy vac = (Vacancy)obj;

if (Position==vac.Position &&

Salary==vac.Salary &&

Speciality==vac.Speciality)

{

return true;

}

else

{

return false;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Linq.Expressions;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

using Human\_Department.Models;

namespace Human\_Department

{

public class Dao

{

Department dep;

const string filePath = "store.bin";

public Dao(Department dep)

{

this.dep = dep;

}

public void Save()

{

using (Stream stream = File.Create(filePath))

{

var serializer = new BinaryFormatter();

serializer.Serialize(stream, dep);

stream.Close();

}

}

public void Load()

{

Read:

try

{

using (Stream stream = File.OpenRead(filePath))

{

var serializer = new BinaryFormatter();

try

{

Department st = (Department)serializer.Deserialize(stream);

Copy(st.Employees, dep.Employees);

Copy(st.Job\_Seekers, dep.Job\_Seekers);

Copy(st.Vacancies, dep.Vacancies);

Copy(st.Orders, dep.Orders);

dep.Chenge\_Director(st.Director);

dep.Set\_Age\_Limit(st.Limit);

}

catch (System.Runtime.Serialization.SerializationException)

{

}

}

}

catch (System.IO.FileNotFoundException)

{

File.Create(filePath);

goto Read;

}

void Copy<T>(List<T> from, List<T> to)

{

try

{

to.Clear();

to.AddRange(from);

}

catch (ArgumentNullException)

{

}

}

}

}

}

using Human\_Department.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Human\_Department

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Start\_Form());

}

}

}

using System;

using System.Linq;

using Human\_Department.Models;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace UnitTestProject\_Human\_Department

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void Fill\_Test\_Data()

{

Department dep = new Department();

dep.Fill\_Test\_Data(100);

Assert.AreEqual("JobseekerNameJobseekerSurnameRTG1569HightPediatric\_radiologyJobseeker",

(dep.Job\_Seekers.Last().First\_Name +

dep.Job\_Seekers.Last().Second\_Name +

dep.Job\_Seekers.Last().Pasport\_Id +

dep.Job\_Seekers.Last().Aducation +

dep.Job\_Seekers.Last().Speciality +

dep.Job\_Seekers.Last().Password));

Assert.AreEqual("EmployeeNameEmployeeSurnameANK1254HightLoyerEmployee",

(dep.Employees.Last().First\_Name +

dep.Employees.Last().Second\_Name +

dep.Employees.Last().Pasport\_Id +

dep.Employees.Last().Aducation +

dep.Employees.Last().Speciality +

dep.Employees.Last().Password));

Assert.AreEqual("DirectorNameDirectorSurnameSDK9054HightLoyerDirector",

(dep.Director.First\_Name +

dep.Director.Second\_Name +

dep.Director.Pasport\_Id +

dep.Director.Aducation +

dep.Director.Speciality +

dep.Director.Password));

Assert.AreEqual("Dog\_Trainer13051,75Adolescent\_medicine",

(dep.Vacancies.Last().Position +

dep.Vacancies.Last().Salary +

dep.Vacancies.Last().Speciality));

Assert.AreEqual("FalseFalsed800Dog\_Trainer",

(""+dep.Orders.Last().Is\_Approved+

dep.Orders.Last().Is\_Cheked +

dep.Orders.Last().Seeker.Pasport\_Id +

dep.Orders.Last().Vacancy.Position));

}

}

}