МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

(факультет)

Кафедра Систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине: «Объектно-ориентированное программирование» .

Тема: «Разработка приложения «Сведения о промышленных предприятиях»» .

Расчетно-пояснительная записка

Разработал студент А.С. Кочетов

Подпись, дата Инициалы, фамилия

Руководитель О.В. Минакова

Подпись, дата Инициалы, фамилия

Нормоконтролер О.В. Минакова

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

ВОРОНЕЖ

2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Кафедра Систем управления и информационных технологий в строительстве

ЗАДАНИЕ

на курсовой проект

по дисциплине: «Объектно-ориентированное программирование» .

Тема: «Разработка приложения «Сведения о промышленных предприятиях»» ..

Студент збПИЭ-211 Кочетов Александр Сергеевич .

Группа, фамилия, имя, отчество

Номер варианта. 14 .

Технические условия:. Среда разработки \*. Язык программирования java, редактор диаграмм draw.io.

Содержание и объем работы (графические работы, расчеты и прочее):. .

. 19 стр., 5 иллюстр. 1 табл. .

Сроки выполнения этапов:. анализ и постановка задачи (01.10-15.10); разработка пошаговой детализации программы (16.10-10.11); реализация программы (11.11-01.12); тестирование программы (01.12-10.12); оформление пояснительной записки (02.12-12.12).

.

Срок защиты курсового проекта: .

Руководитель . О.В. Минакова .

Подпись, дата Инициалы, фамилия

Задание принял студент . А.С. Кочетов

Подпись, дата Инициалы, фамилия

СОДЕРЖАНИЕ

ВВЕДЕНИЕ…………………………………………..………………….…………4

1. Постановка задачи 5

1.1. Обоснование полей класса и методов основной сущности…………..

программы………………………………….. 5

1.2. Диаграмма классов………………………………….. 6

1.3. CRC-карточка………………………………….. 7

2. Разработка интерфейсной части. 10

2.1 Описание графического интерфейса пользователя. 10

2.2 Диаграммы последовательностей и взаимодействия. 12

3. Разработка кода программы.. 14

3.1 Диаграмма классов приложения и её описание 14

3.2. Алгоритм реализации методов. 17

ЗАКЛЮЧЕНИЕ…………………………………………...…...…….……………18

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.…………...………..….……19

ВВЕДЕНИЕ

Современные промышленные предприятия играют ключевую роль в экономике любой страны, и эффективное управление их деятельностью требует наличия точной и актуальной информации. Сбор, анализ и представление данных о промышленных компаниях становятся непростыми задачами в условиях постоянно меняющихся экономических реалий. Настоящий курсовой проект посвящен разработке программного приложения «Сведения о промышленных предприятиях», которое позволяет пользователям удобно управлять информацией о предприятиях, включая их названия, адреса, численность сотрудников и годовые доходы.

Данная программа призвана значительно упростить процесс ввода и хранения данных, предоставляя интуитивно понятный графический интерфейс, где пользователи могут легко добавлять и просматривать информацию о различных предприятиях. С использованием языка программирования Java и технологий JavaFX, приложение обеспечивает современный и функциональный интерфейс, который поддерживает взаимодействие с пользователем и наглядное отображение данных.

В процессе работы над проектом будет рассмотрен процесс проектирования и реализации приложения, включающий в себя анализ требований, разработку интерфейса и кодов программы, а также тестирование функциональности. В результате, пользователи получат эффективные инструменты для управления информацией о промышленных предприятиях, что поможет им принимать обоснованные решения и оптимизировать процессы на уровне управления.

1. Постановка задачи

1.1 Обоснование полей класса и методов основной сущности программы

В данном разделе рассматривается приложение «Сведения о промышленных предприятиях», цель которого - собрать и предоставить пользователю информацию о различных предприятиях. Основной сущностью программы является класс Enterprise, который содержит информацию о предприятии. Рассмотрим поля этого класса [1]:

name: Название предприятия. Тип данных - String. Это обязательное поле, так как каждое предприятие должно иметь уникальное имя.

address: Адрес предприятия. Тип данных - String. Адрес также является обязательным полем, так как он необходим для идентификации местоположения предприятия.

numberOfEmployees: Количество сотрудников. Тип данных - int. Это числовое поле, которое позволяет отслеживать размер трудовых ресурсов предприятия.

annualRevenue: Годовая выручка. Тип данных - double. Это числовое поле, которое позволяет отслеживать финансовое состояние предприятия.

Каждое из этих полей имеет соответствующие геттеры и сеттеры для получения и изменения значений. Эти методы обеспечивают доступ к данным и возможность их модификации.

Диаграмма классов:

- Enterprise: содержит поля и методы для работы с данными о предприятии.

- MainController: главный контроллер, который управляет пользовательским интерфейсом и взаимодействует с объектами класса Enterprise.

1.2 Диаграмма классов

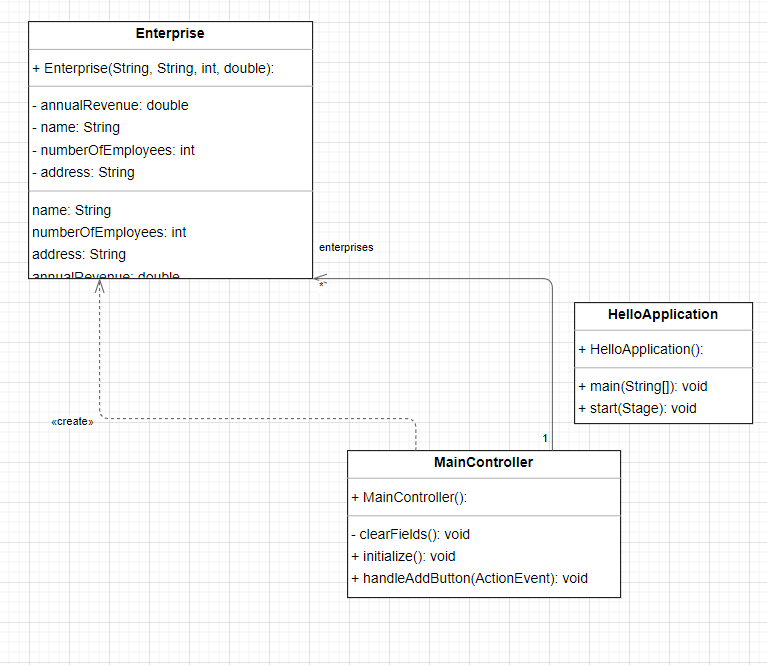
Для наглядного представления структуры класса Enterprise можно использовать диаграмму классов, представленную на рисунке 1.1., которая позволяет визуализировать основные компоненты и характеристики данного класса [2]. Эта диаграмма позволяет увидеть, как организованы атрибуты и методы класса, а также демонстрирует взаимосвязи с другими классами в системе.

Рисунок 1.1. Диаграмма классов.

1.3 CRC-карточка

Вот подробная таблица 1.1 CRC-карточки для классов [3]:

Таблица 1.1. CRC - карточка

|  |  |  |
| --- | --- | --- |
| **Класс** | **Ответственности** | **Сотрудничество** |
| MainController | -Управляет интерфейсом приложения для отображения и обработки данных об предприятиях. | - Соседствует с классом Enterprise для создания, хранения и управления объектами предприятия. |
|  | -Инициализирует таблицу в пользовательском интерфейсе и заполняет ее начальными данными (предприятиями). | - Использует ObservableList<  Enterprise> для хранения и управления коллекцией предприятий. |
|  | -Настраивает колонки таблицы, связывая их с полями класса Enterprise для отображения соответствующих данных. | - Взаимодействует с элементами управления TextField для получения пользовательского ввода. |
|  | - Обрабатывает события (например, нажатие кнопки) для добавления новых предприятий и обновления таблицы. |  |
|  | - Очищает поля ввода после добавления предприятия. |  |
|  |  |  |
| Enterprise | - Представляет данные о предприятии, включая название, адрес, количество сотрудников и годовой доход. | -Используется классом MainController для управления и отображения информации в таблице. |
|  | - Предоставляет методы для доступа и изменения своих полей (геттеры и сеттеры). |  |
|  |  |  |
| HelloApplication | - Запускает приложение, загружает интерфейс из FXML-файла и устанавливает основную сцену. | - Сотрудничает с JavaFX библиотеки для работы с интерфейсом. |
|  | - Определяет точку входа в приложение. |  |

Объяснение каждой ответственности и сотрудничества:

* 1. Класс MainController:
     + Ответственности: Класс отвечает за логику управления интерфейсом приложения. Он инициализирует таблицу, добавляет начальные данные, управляет пользовательским вводом и обновляет отображение данных.
     + Сотрудничество: взаимодействует с объектами класса Enterprise, чтобы создавать и хранить информацию о новых предприятиях. Также использует компоненты интерфейса TextField для получения данных от пользователя.
  2. Класс Enterprise:
     + Ответственности: Этот класс служит моделью данных, хранящей информацию о каждом предприятии. Он предоставляет доступ к своим полям через методы (геттеры и сеттеры), что упрощает управление данными в приложении.
     + Сотрудничество: используется классом MainController, который управляет коллекцией объектов Enterprise и отображает их в интерфейсе.
  3. Класс HelloApplication:
     + Ответственности: Класс стартует приложение, загружает интерфейс из файла FXML и отображает его в окне приложения [4].
     + Сотрудничество: взаимодействует с библиотекой JavaFX, чтобы управлять жизненным циклом приложения.

Эта таблица CRC-карты дает чёткое представление о структуре приложения, его классах, их функциях и взаимодействии.

2. Разработка интерфейсной части

2.1 Описание графического интерфейса пользователя

Графический интерфейс пользователя (GUI) включает в себя несколько элементов управления, которые позволяют пользователю взаимодействовать с программой [5].

Основные элементы управления:

TextField для ввода названия предприятия, адреса, количества сотрудников и годовой выручки.

TableView для отображения списка предприятий.

Button для добавления нового предприятия.

TableColumn для отображения отдельных полей в таблице.

Иерархия элементов управления:

- Главное окно (Stage)

- Панель верхнего уровня (BorderPane)

- Верхняя панель (HBox) для текстовых полей и кнопок

- TextField для ввода названия предприятия

- TextField для ввода адреса

- TextField для ввода количества сотрудников

- TextField для ввода годовой выручки

- Button для добавления нового предприятия

- Центральная панель (VBox) для отображения таблицы

- TableView для отображения списка предприятий

Обоснование выбора менеджера компоновки:

- Использование BorderPane позволяет организовать элементы управления в логической структуре: верхняя панель для ввода данных и центральная панель для отображения таблицы.

- Использование HBox для верхней панели позволяет разместить все текстовые поля и кнопку в одной строке.

- Использование VBox для центральной панели позволяет разместить таблицу в центре окна.

Произведем раскадровку рабочего окна, рисунок 2.1., на нем наглядно можно увидеть основные элементы управления.

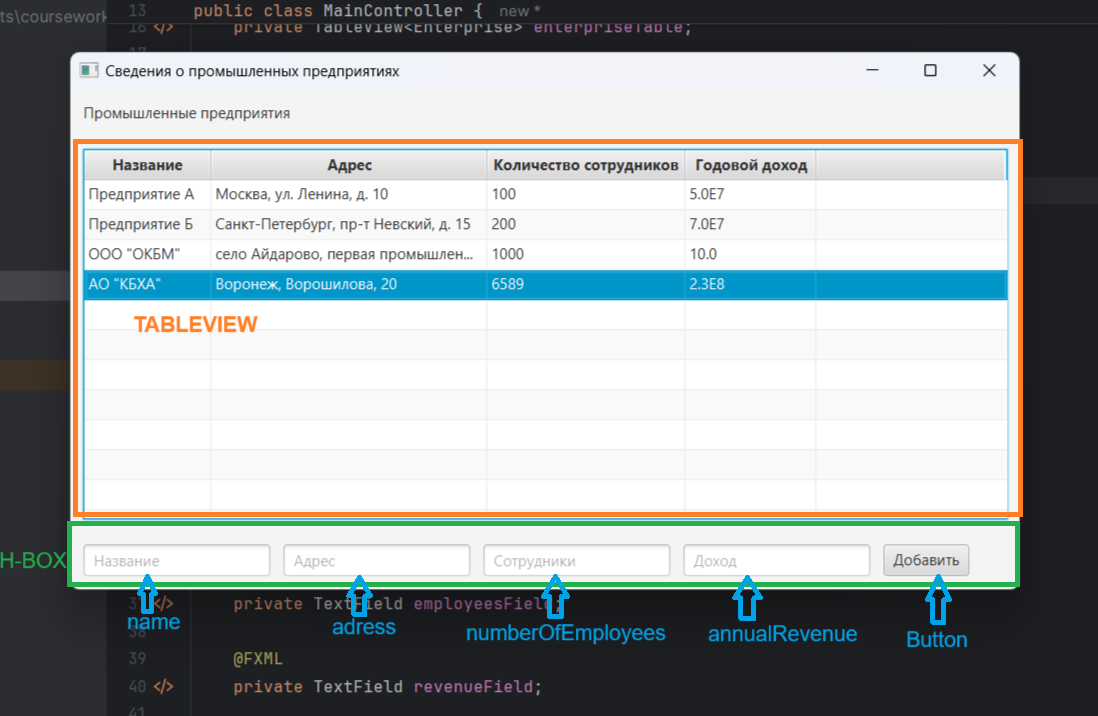


Рисунок 2.1. Раскадровка рабочего окна.

2.2 Диаграммы последовательностей и взаимодействия

Для более наглядной демонстрации взаимодействия пользователя с элементами управления и объектами класса Enterprise можно эффективно использовать диаграммы последовательностей, которые позволяют визуализировать последовательность взаимодействий между различными компонентами системы в течение определенного сценария. Такую диаграмму можно считать своеобразной картой, показывающей, как именно происходит взаимодействие между пользователем и программой, включая все этапы и шаги, которые необходимо выполнить.

Диаграмма последовательностей для добавления нового предприятия представлена на рисунке 2.2. На данном рисунке отображены ключевые моменты, начиная от инициирования процесса пользователем, где происходит нажатие кнопки для добавления нового предприятия, и заканчивая подтверждением успешного добавления записи в систему. Каждый шаг, который пользователь предпринимает, и каждый ответ системы на эти действия четко продемонстрированы, что помогает лучше понять процесс, а также выявить возможные узкие места или сложности, с которыми может столкнуться пользователь [6].

Таким образом, диаграммы последовательностей не только иллюстрируют ход взаимодействия, но и имеют важное значение для анализа и оптимизации пользовательского опыта, поскольку они дают возможность разработчикам видеть полный контекст работы программы и предпочтения пользователей.

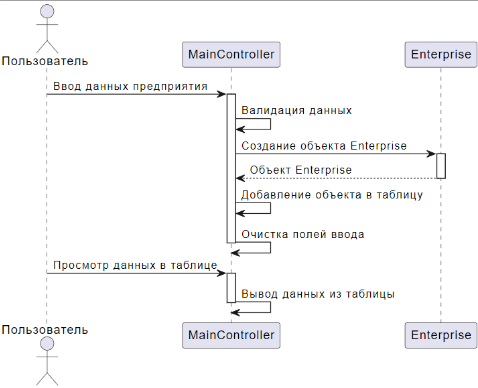


Рисунок 2.2. Диаграмма последовательностей для добавления нового предприятия.

Также теперь опишем более подробно процесс взаимодействия пользователя с программой, которая изображена на диаграмме взаимодействия (рисунок 2.3.), Взаимодействие пользователя с системой представляет собой последовательность шагов, которые пользователь предпринимает для достижения определённых целей и выполнения различных задач. Каждый шаг, визуализированный на диаграмме, играет ключевую роль в создании общего опыта работы с программой [7].

В целом, диаграмма взаимодействия (рисунок 2.3.) служит важным инструментом, позволяющим визуализировать и систематизировать шаги, которые пользователь проходит в процессе работы с программой, обеспечивая тем самым более глубокое понимание функциональности и эффективности взаимодействия с программным обеспечением.

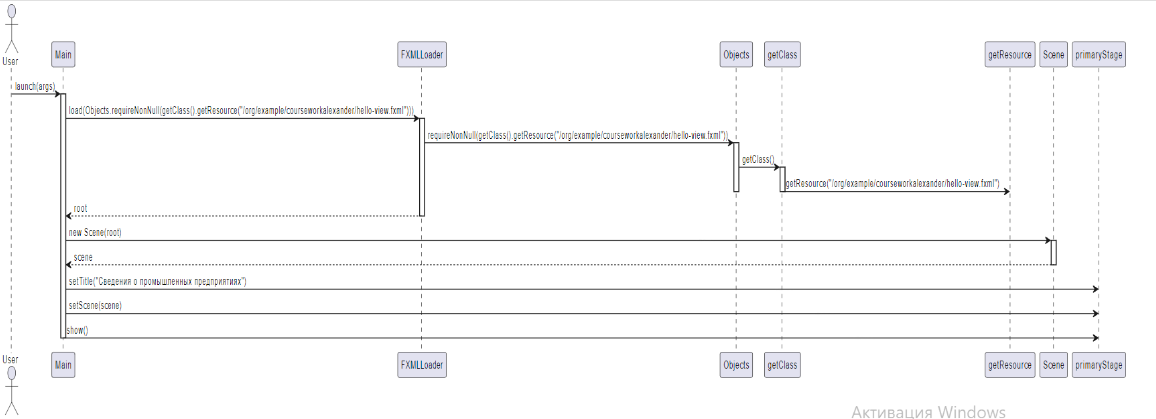


Рисунок 2.3. Диаграмма последовательностей для добавления нового предприятия.

3. Построение кода программы

3.1 Диаграмма классов приложения и её описание

Диаграмма классов приложения (рисунок 3.1.) отражает структуру классов и их взаимосвязь, что является очень важным аспектом для понимания архитектуры всего приложения. Каждый класс в этой диаграмме имеет свои уникальные атрибуты и методы, которые определяют его поведение и функции в рамках работы системы. Атрибуты представляют собой характеристики, которые описывают состояние объекта, а методы — это действия, которые объект может выполнять, создавая тем самым функциональность приложения [8].

Кроме того, взаимосвязи между классами демонстрируют, как они взаимодействуют друг с другом, что позволяет выстраивать осмысленную и эффективную логику работы программы. Наличие различных типов связей, таких как ассоциации, агрегации и композиций, также подчеркивает комплексность и многоуровневую структуру, которая обеспечивает согласованность и целостность всего приложения. Такой визуальный инструмент, как диаграмма классов, значительно упрощает процесс проектирования и разработки программного обеспечения, позволяя разработчикам и заинтересованным сторонам легко видеть, как различные компоненты системы связаны и взаимодействуют между собой.

В конечном итоге, диаграмма классов является не просто иллюстрацией, а настоящим инструментом, который способствует лучшему пониманию структуры приложения и служит основой для его дальнейшего развития и масштабирования.

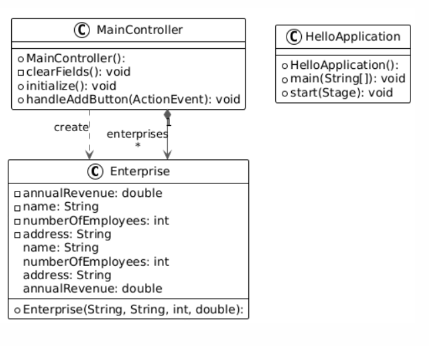


Рисунок 3.1. Диаграмма классов приложения.

Описание каждого класса [9]:

Enterprise - класс, представляющий информацию о предприятии:

* Атрибуты:
  + name: Название предприятия. Тип данных - String.
  + address: Адрес предприятия. Тип данных - String.
  + numberOfEmployees: Количество сотрудников. Тип данных - int.
  + annualRevenue: Годовая выручка. Тип данных - double.
* Методы:
  + getName(): Возвращает название предприятия.
  + getAddress(): Возвращает адрес предприятия.
  + getNumberOfEmployees(): Возвращает количество сотрудников.
  + getAnnualRevenue(): Возвращает годовую выручку.
  + setName(String): Устанавливает новое название предприятия.
  + setAddress(String): Устанавливает новый адрес предприятия.
  + setNumberOfEmployees(int): устанавливает новое количество сотрудников.
  + setAnnualRevenue(double): устанавливает новую годовую выручку.

HelloApplication - основной класс приложения:

* Атрибуты:
  + Нет.
* Методы:
  + start(Stage primaryStage): Запускает основное окно приложения.
  + main(String[] args): Точка входа в программу.

MainController - контроллер для управления функциональностью:

* Атрибуты:
  + enterpriseTable: Таблица для отображения списка предприятий.
  + nameColumn, addressColumn, employeesColumn, revenueColumn: Колонки таблицы для отображения отдельных полей.
  + nameField, addressField, employeesField, revenueField: Текстовые поля для ввода данных.
  + enterprises: Список предприятий для хранения данных.
* Методы:
  + initialize(): Настройка колонок таблицы и добавление тестовых данных.
  + handleAddButton(ActionEvent event): Обработка нажатия кнопки «Добавить» для добавления нового предприятия.
  + clearFields(): Очистка текстовых полей после добавления предприятия.

3.2 Алгоритм реализации методов

handleAddButton(ActionEvent event)

1. В первую очередь, необходимо получить значения из текстовых полей, таких как nameField, addressField, employeesField и revenueField, чтобы мы могли с ними далее работать.

2. После этого следует преобразовать значения, которые мы только что получили, в соответствующие типы данных, чтобы обеспечить их корректную обработку в дальнейшем.

3. Следующим важным шагом будет создание нового объекта класса Enterprise с использованием полученных значений, что позволяет нам эффективно организовать данные.

4. Затем, необходимо добавить только что созданный новый объект в заранее определенный список предприятий (enterprises), что даст нам возможность легко управлять всем списком.

5. Теперь необходимо выполнить очистку текстовых полей, чтобы подготовить их для ввода новой информации в будущем, что продлевает срок службы пользовательского интерфейса.

6. В заключение, обновить таблицу (enterpriseTable) для отображения нового предприятия, чтобы все изменения, которые мы сделали, были отражены в интерфейсе для пользователя, и он мог видеть актуальные данные [10].

ЗАКЛЮЧЕНИЕ

В ходе реализации курсового проекта «Сведения о промышленных предприятиях» была разработана функциональная программа, сочетающая в себе удобный графический интерфейс и инструментарий для работы с данными о предприятиях. Приложение позволяет пользователям эффективно заполнять, хранить и просматривать информацию о различных промышленных компаниях, что является важным аспектом для повышения качества управления и принятия решений в реальном времени.

Были подробно рассмотрены основные компоненты приложения, включая архитектуру кода, диаграммы классов и интерфейсные элементы. Проект продемонстрировал, как использование языка Java и платформы JavaFX может значительно упростить разработку сложных приложений. Также важным результатом является возможность дальнейшего улучшения приложения: планируется добавить функции редактирования и удаления записей, расширить функционал по работе с данными, а также интеграцию с базами данных для надежного хранения информации.

Таким образом, данный курсовой проект не только достиг поставленных целей, но и заложил основу для будущих усовершенствований, что подтверждает его актуальность и значимость в современном мире управления промышленными предприятиями.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Бурков А. В., Сидорцов Д. Н. Разработка мобильных приложений: учебное пособие. М.: Издательство "Диалог-МИФИ", 2020. 300 с.
2. Гридин В. И., Емельянов А. И. Проектирование и развитие баз данных. М.: Издательство Юрайт, 2021. 280 с.
3. Князев И. А. Разработка веб-приложений с использованием технологий HTML5 и CSS3. СПб.: Питер, 2022. 320 с.
4. Ильин П. А., Пантелеев В. А. Программирование на Python для предпринимателей. Екатеринбург: Издательство Уральского университета, 2021. 240 с.
5. Петрова М. Ю. Изучение и применение подходов к разработке web-приложений для бизнеса. Казань: Казанский федеральный университет, 2023.264 с.
6. Куликов А. И. Современные методы автоматизации процессов предприятия. М.: Наука, 2022. 312 с.
7. Панфилова Н. В. Информационные технологии в управлении промышленными предприятиями. М.: Инфра-М, 2021. 224 с.
8. Федорова Т. А. Основы проектирования интерфейсов для мобильных приложений. СПб.: БХВ-Петербург, 2023. 240 с.
9. Смирнов А. Л., Кузнецов С. С. Основы работы с RESTful API: практическое руководство. М.: Эксмо, 2020. 192 с.
10. Лебедев А. Н. Создание и управление базами данных для мобильных приложений. М.: ВАК, 2022. 350 с.

ПРИЛОЖЕНИЕ 1  
 Листинг программы.

Enterprise

package org.example.courseworkalexander;

public class Enterprise {

private String name;

private String address;

private int numberOfEmployees;

private double annualRevenue;

public Enterprise(String name, String address, int numberOfEmployees, double annualRevenue) {

this.name = name;

this.address = address;

this.numberOfEmployees = numberOfEmployees;

this.annualRevenue = annualRevenue;

}

// Геттеры

public String getName() {

return name;

}

public String getAddress() {

return address;

}

public int getNumberOfEmployees() {

return numberOfEmployees;

}

public double getAnnualRevenue() {

return annualRevenue;

}

// Сеттеры

public void setName(String name) {

this.name = name;

}

public void setAddress(String address) {

this.address = address;

}

public void setNumberOfEmployees(int numberOfEmployees) {

this.numberOfEmployees = numberOfEmployees;

}

public void setAnnualRevenue(double annualRevenue) {

this.annualRevenue = annualRevenue;

}

}

HelloAplication  
package org.example.courseworkalexander;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

import java.util.Objects;

public class HelloApplication extends Application {

@Override

public void start(Stage primaryStage) throws Exception{

Parent root = FXMLLoader.load(Objects.requireNonNull(getClass().getResource("/org/example/courseworkalexander/hello-view.fxml")));

Scene scene = new Scene(root);

primaryStage.setTitle("Сведения о промышленных предприятиях");

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

MainController

package org.example.courseworkalexander;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.TextField;

import javafx.scene.control.cell.PropertyValueFactory;

public class MainController {

@FXML

private TableView<Enterprise> enterpriseTable;

@FXML

private TableColumn<Enterprise, String> nameColumn;

@FXML

private TableColumn<Enterprise, String> addressColumn;

@FXML

private TableColumn<Enterprise, Integer> employeesColumn;

@FXML

private TableColumn<Enterprise, Double> revenueColumn;

@FXML

private TextField nameField;

@FXML

private TextField addressField;

@FXML

private TextField employeesField;

@FXML

private TextField revenueField;

private ObservableList<Enterprise> enterprises = FXCollections.observableArrayList();

public void initialize() {

// Настройка колонок таблицы

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

addressColumn.setCellValueFactory(new PropertyValueFactory<>("address"));

employeesColumn.setCellValueFactory(new PropertyValueFactory<>("numberOfEmployees"));

revenueColumn.setCellValueFactory(new PropertyValueFactory<>("annualRevenue"));

// Добавление тестовых данных

enterprises.addAll(

new Enterprise("Предприятие А", "Москва, ул. Ленина, д. 10", 100, 50\_000\_000),

new Enterprise("Предприятие Б", "Санкт-Петербург, пр-т Невский, д. 15", 200, 70\_000\_000)

);

// Установка данных в таблицу

enterpriseTable.setItems(enterprises);

}

@FXML

public void handleAddButton(ActionEvent event) {

String name = nameField.getText();

String address = addressField.getText();

int employees = Integer.parseInt(employeesField.getText());

double revenue = Double.parseDouble(revenueField.getText());

Enterprise newEnterprise = new Enterprise(name, address, employees, revenue);

enterprises.add(newEnterprise);

clearFields();

}

private void clearFields() {

nameField.clear();

addressField.clear();

employeesField.clear();

revenueField.clear();

}

}

ПРИЛОЖЕНИЕ 2

Скриншоты рабочих окон программы в различных режимах работы.

