МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» им.В.И.УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

**Пояснительная записка к курсовой работе**

по дисциплине «Объектно-ориентированное программирование»

Выполнила студентка гр. 1308 \_\_\_\_\_\_\_\_\_\_\_\_\_ Черникова П. В.

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_ Гречухин М. Н.

Санкт-Петербург

2023

Содержание

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc137486335)

[ОПИСАНИЕ КЛАССОВ 4](#_Toc137486336)

[КОД КЛАССОВ-СУЩНОСТЕЙ 10](#_Toc137486337)

[МОДЕЛЬ БАЗЫ ДАННЫХ 11](#_Toc137486338)

[ПРИМЕР ИНТЕРФЕЙСА ПРОГРАММЫ 12](#_Toc137486339)

[ФОРМИРОВАНИЕ ОТЧЕТА ПО ТАБЛИЦАМ ПРОГРАММЫ 13](#_Toc137486340)

[РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 15](#_Toc137486341)

[ПОЛНЫЙ КОД ПРОГРАММЫ 25](#_Toc137486342)

[Приложение А 27](#_Toc137486343)

[Приложение Б 35](#_Toc137486344)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Задание 6. Разработать ПК для диспетчера автобусного парка. В ПК

должна храниться информация о водителях, маршрутах и графике движения

автобусов. Диспетчер автобусного парка может добавлять, изменять и

удалять эту информацию. Ему могут потребоваться следующие сведения:

* список водителей, их стаж работы и класс;
* когда начинается или заканчивается движение автобуса на всех или
* отдельных маршрутах;
* справка о графике движения автобусов и отчет о его нарушениях.

# ОПИСАНИЕ КЛАССОВ

**Диаграмма программных классов**

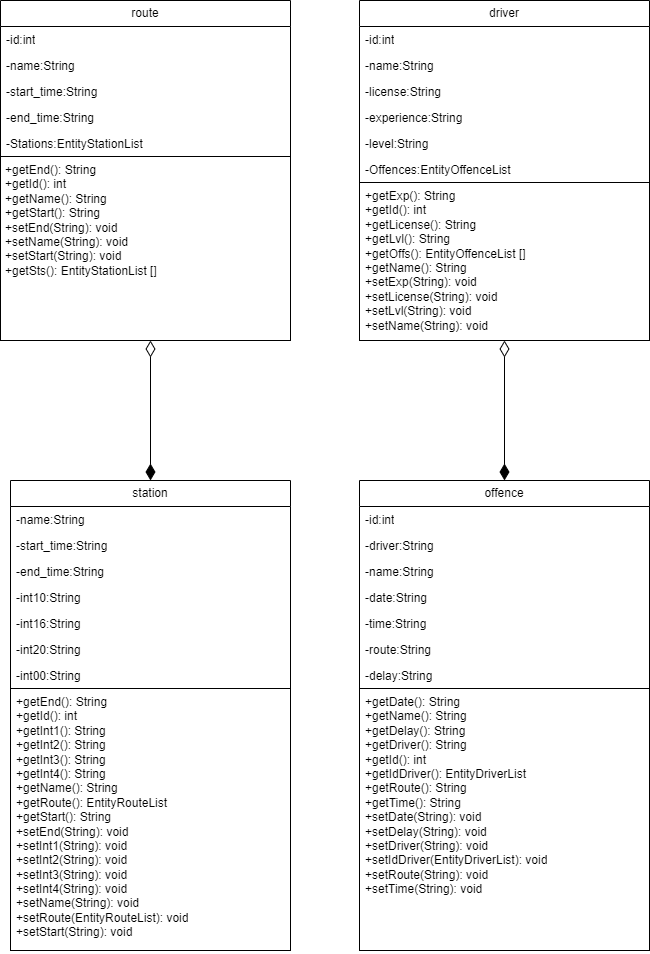


Рисунок 1. UML-диаграмма классов.

**Описание реализации классов:**

**EntityDriverList.java**

Класс, описывающий сущность «Список водителей»

Таблица 1 Свойства класса EntityDriverList.java

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Семантика** |
| driver\_id | int | ID водителя |
| name | String | Фамилия и имя |
| license | String | Номер прав |
| experience | String | Стаж |
| level | String | Класс вождения |
| Offences | EntityOffenceList [] | Нарушения |

Таблица 2 Методы класса EntityDriverList.java

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Возвращаемый тип** | **Параметры** | **Назначение** |
| getId | int |  | Получение ID |
| getName | String |  | Получение фамилии и имени |
| getLicense | String |  | Получение номера прав |
| getExp | String |  | Получение стажа |
| getLvl | String |  | Получение класса |
| getOffs | EntityOffenceList [] |  | Получение нарушений |
| setLicense | void | String new\_license | Установка номера прав |
| setName | void | String new\_name | Установка фамилии и имени |
| getExp | void | String new\_exp | Установка стажа |
| setLvl | void | String new\_lvl | Установка класса |

**EntityOffenceList.java**

Класс, описывающий сущность «Список нарушений графика движения»

Таблица 3 Свойства класса EntityOffenceList.java

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Семантика** |
| offence\_id | int | ID нарушения |
| driver | int | Номер прав |
| date | int | Дата нарушения |
| time | int | Время нарушения |
| route | int | Маршрут |
| delay | String | Время задержки |
| id\_driver | EntityDriverList | Ссылка на соответствующего водителя |

Таблица 4 Методы класса EntityOffenceList.java

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Возвращаемый тип** | **Параметры** | **Назначение** |
| getId | int |  | Получение Id |
| getDriver | String |  | Получение прав |
| getDate | String |  | Получение даты нарушения |
| getTime | String |  | Получение времени нарушения |
| getRoute | String |  | Получение маршрута |
| getDelay | String |  | Получение времени задержки |
| getIdDriver | EntityDriverList |  | Получение соответствующего водителя |
| setDriver | void | String new\_driver | Установка номера прав |
| setDate | void | String new\_date | Установка даты |
| setTime | void | String new\_time | Установка времени |
| setRoute | void | String new\_route | Установка маршрута |
| setDelay | void | String new\_delay | Установка времени здержки |
| setIdDriver | void | EntityDriverList new\_id\_driver | Установка нового водителя |

**EntityRouteList.java**

Класс, описывающий сущность «Список маршрутов»

Таблица 5 Свойства класса EntityRouteList.java

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Семантика** |
| route\_id | int | ID маршрута |
| name | String | Название |
| time1 | String | Время начала движения |
| time2 | String | Время завершения движения |
| Stations | EntityStationList [] | Список остановок |

Таблица 6 Методы класса EntityRouteList.java

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Возвращаемый тип** | **Параметры** | **Назначение** |
| getId | int |  | Получение id маршрута |
| getName | String |  | Получение названия маршрута |
| setStart | String |  | Получение времени начала движения |
| getEnd | String |  | Получение времени завершения движения |
| setName | void | String new\_name | Установка названия |
| setStart | void | String new\_time1 | Установка времени начала движения |
| setEnd | void | String new\_time2 | Установка времени завершения движения |
| getSts | EntityStationList [] |  | Получение списка остановок |

**EntityStationList.java**

Класс, описывающий сущность «Список остановок»

Таблица 7 Свойства класса EntityStationList.java

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Семантика** |
| station\_id | int | ID остановки |
| name | String | Название |
| time1 | String | Время начала движения |
| time2 | String | Время завершения движения |
| int10 | String | Интервал движения до 10:00 |
| int16 | String | Интервал движения до 16:00 |
| int20 | String | Интервал движения до 20:00 |
| int00 | String | Интервал движения после 20:00 |
| id\_route | EntityRouteList | Ссылка на соответствующий маршрут |

Таблица 8 Методы класса EntityStationList.java

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Возвращаемый тип** | **Параметры** | **Назначение** |
| getId | void |  | Получение Id остановки |
| getName | void |  | Получение названия |
| getStart | void |  | Получение времени начала движения |
| getEnd | void |  | Получение времени конца движения |
| getInt1 | void |  | Получение интервала движения до 10:00 |
| getInt2 | void |  | Получение интервала движения до 16:00 |
| getInt3 | void |  | Получение интервала движения до 20:00 |
| getInt4 | void |  | Получение интервала движения после 20:00 |
| getRoute | void |  | Получение соответствующего маршрута |
| setName | String | new\_name | Установка названия |
| setStart | String | new\_time1 | Установка времени начала движения |
| setEnd | String | new\_time2 | Установка времени конца движения |
| setInt1 | String | new\_int1 | Установка интервала движения до 10:00 |
| setInt2 | String | new\_int2 | Установка интервала движения до 16:00 |
| setInt3 | String | new\_int3 | Установка интервала движения до 20:00 |
| setInt4 | String | new\_int4 | Установка интервала движения после 20:00 |
| setRoute | EntityRouteList | new\_route | Установка нового маршрута |

# КОД КЛАССОВ-СУЩНОСТЕЙ

См. приложение А.

# МОДЕЛЬ БАЗЫ ДАННЫХ

В курсовой работе была использована библиотека Java Hibernate для обращения к СУБД MySQL. Поля таблиц и их связи между ними изображены на рисунке ниже:

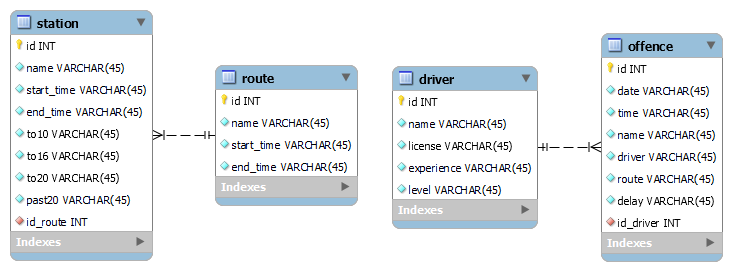


Рисунок 1. ER-диаграмма таблиц базы данных.

Отношениями OneToMany связаны таблица водителей c таблицей нарушений, и таблица маршрутов с таблицей остановок. Один маршрут может содержать в себе набор из нескольких остановок, так же как один водитель может иметь несколько штрафов. Соответственно отношениями ManyToOne связаны таблица нарушений c таблицей водителей, и таблица остановок с таблицей маршрутов.

# ПРИМЕР ИНТЕРФЕЙСА ПРОГРАММЫ

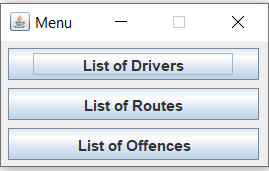


Рисунок 3. Экран с меню.

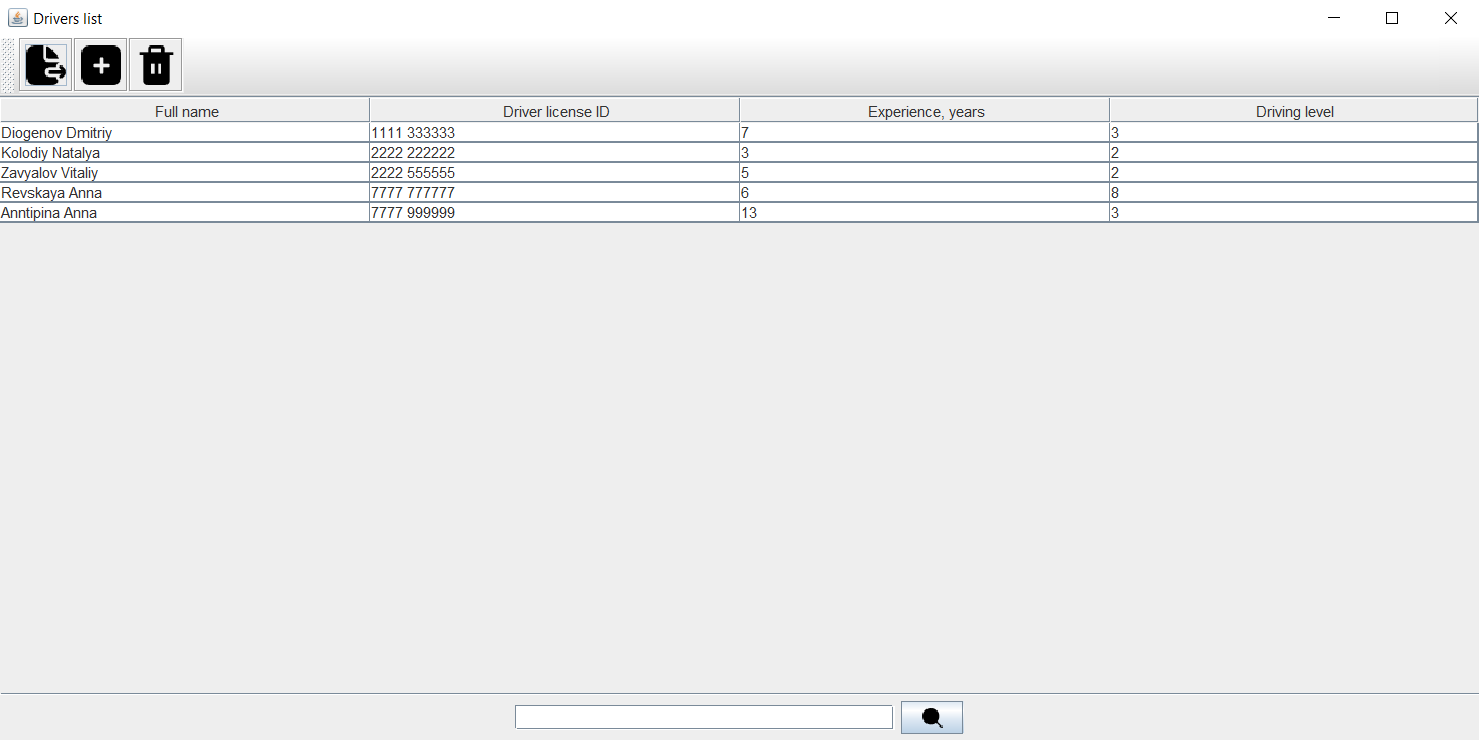


Рисунок 2. Экран со списком водителей.

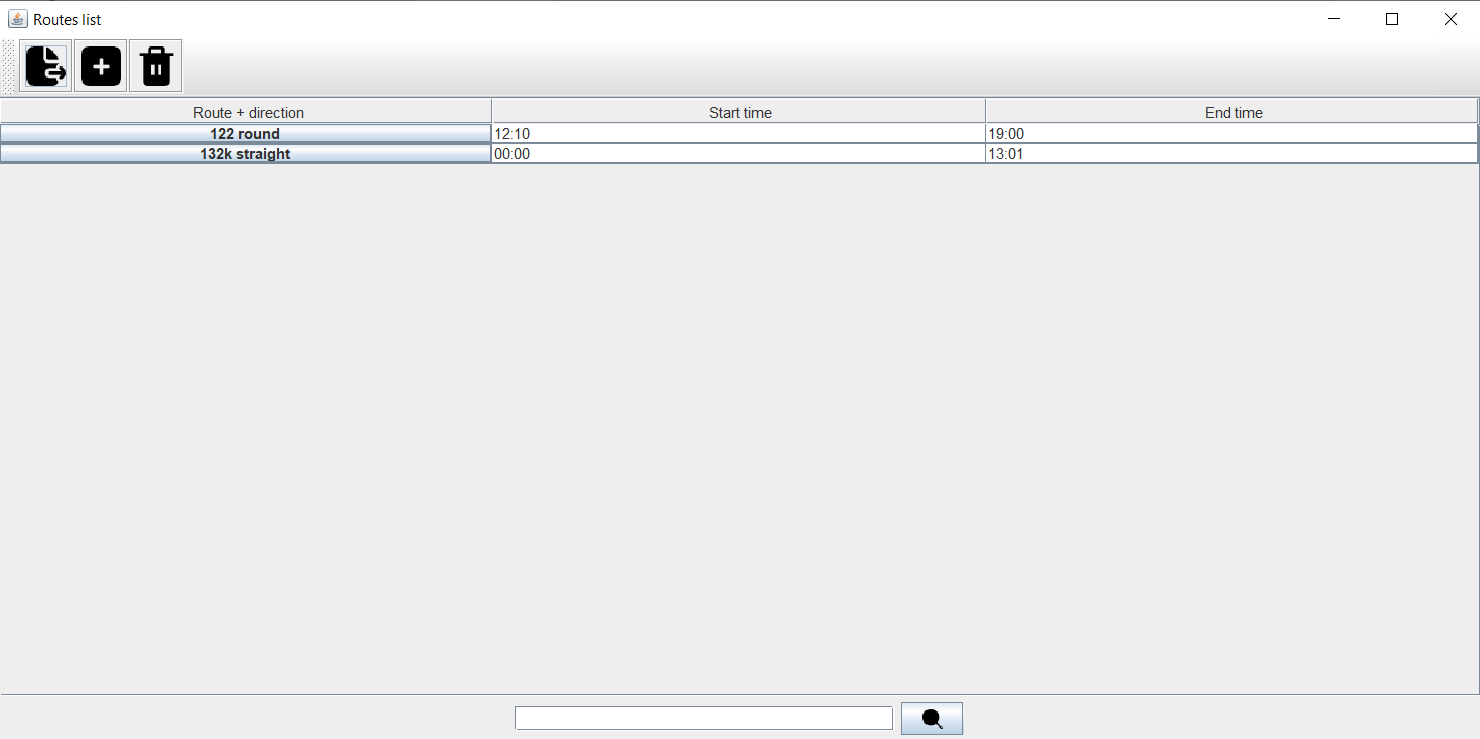


Рисунок 3. Экранная форма маршрутов.

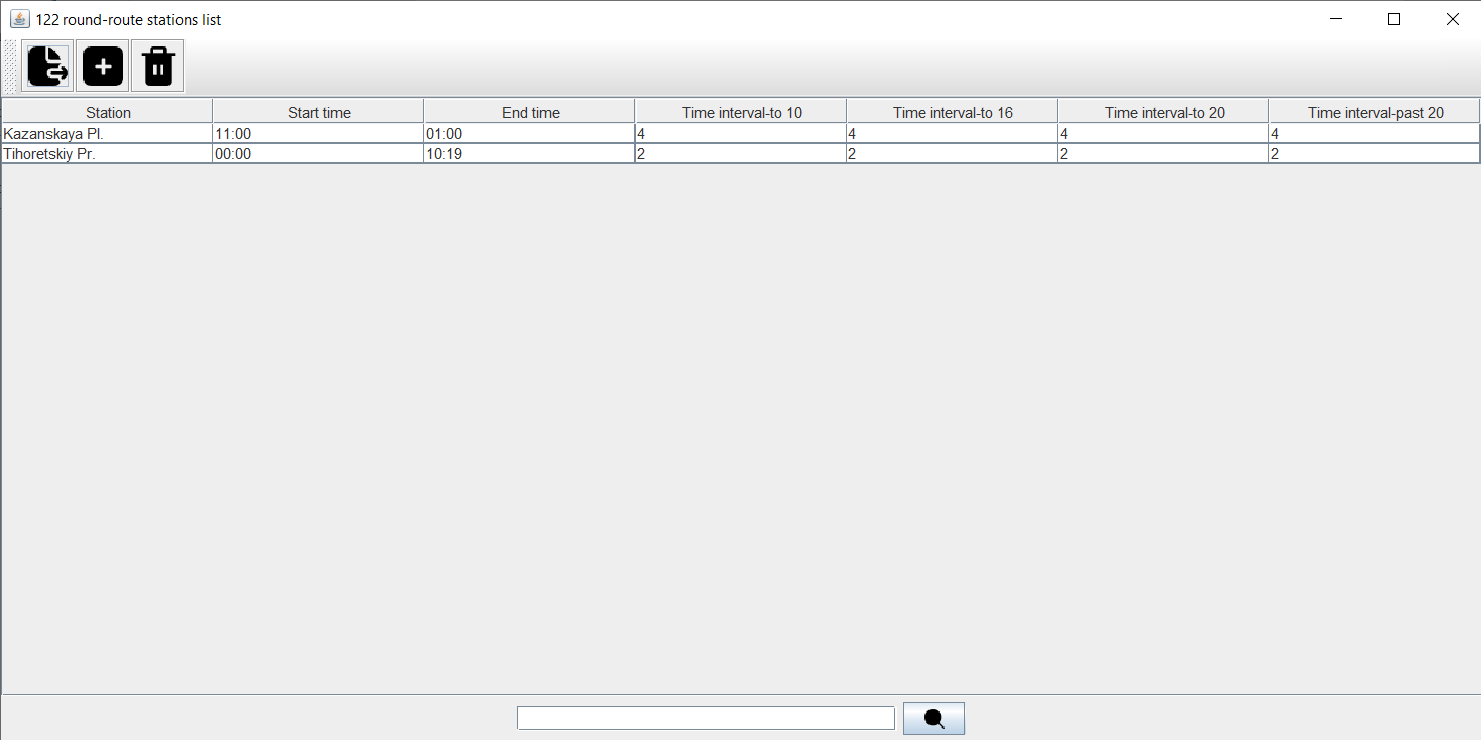


Рисунок 4. Экран остановок маршрута и графика движения на нем.

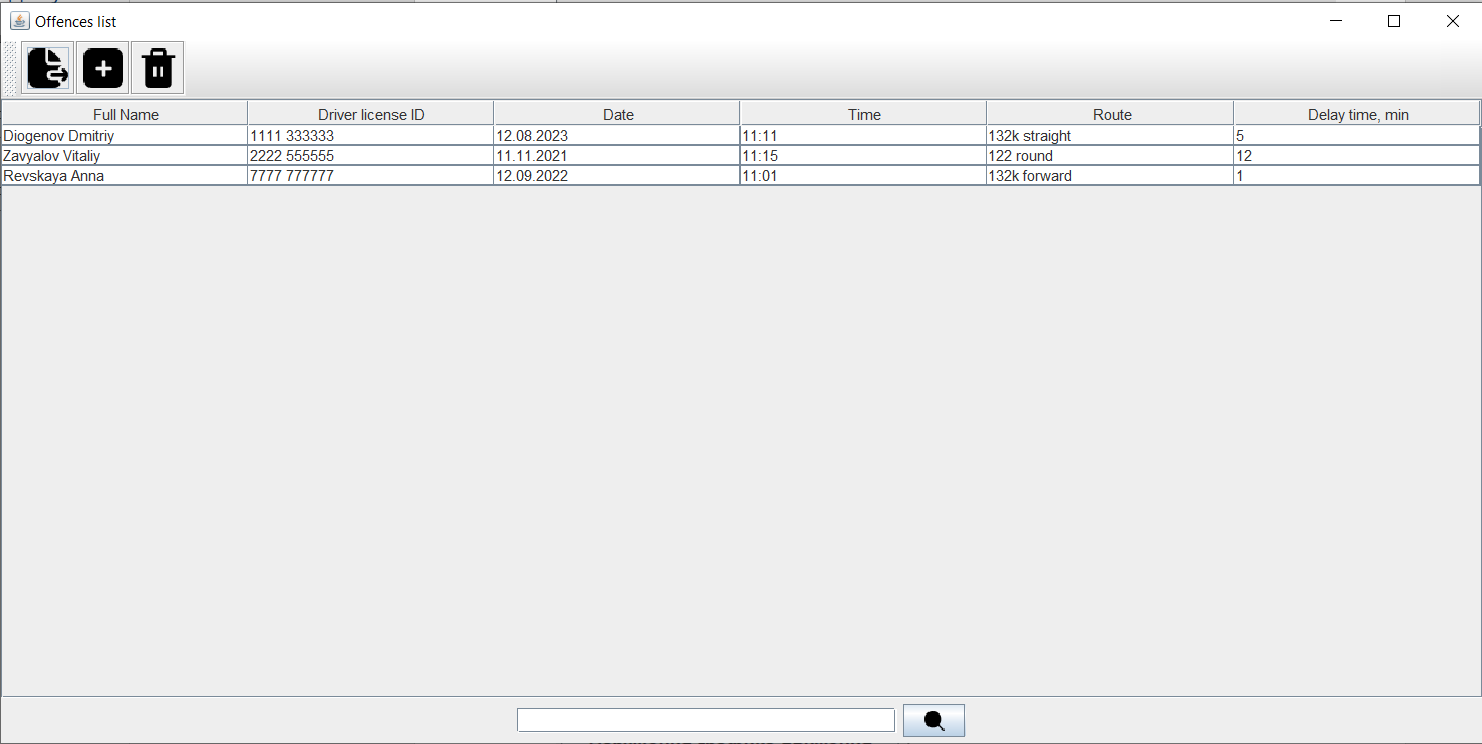


Рисунок 5. Экран со списком нарушений графика движения.

# ФОРМИРОВАНИЕ ОТЧЕТА ПО ТАБЛИЦАМ ПРОГРАММЫ

ПК предлагает формирование и выгрузку отчета в формате .pdf по каждой из таблиц, который будет выглядеть следующим образом:

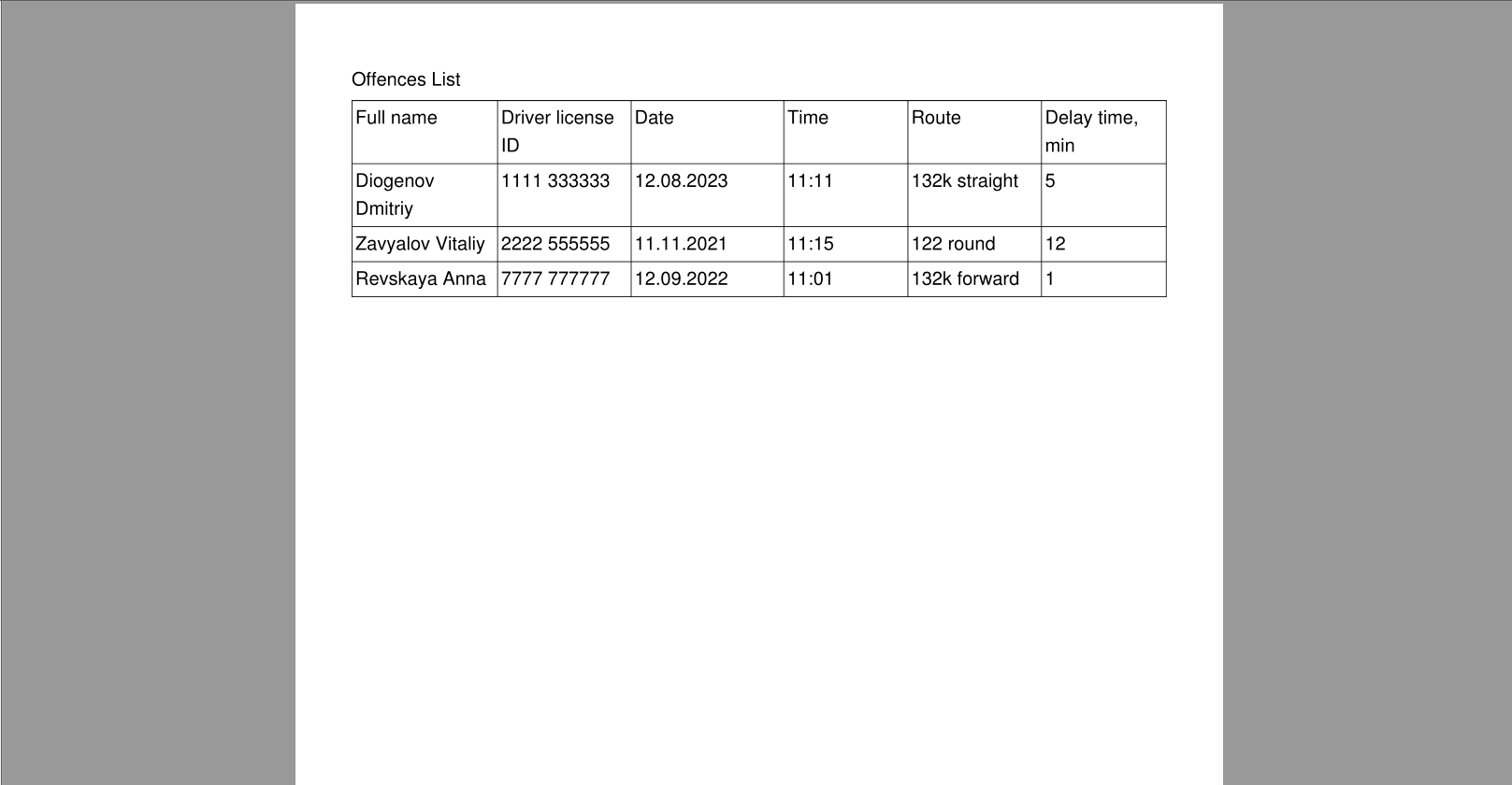


Рисунок 8. Пример отчета о нарушениях графика движения.

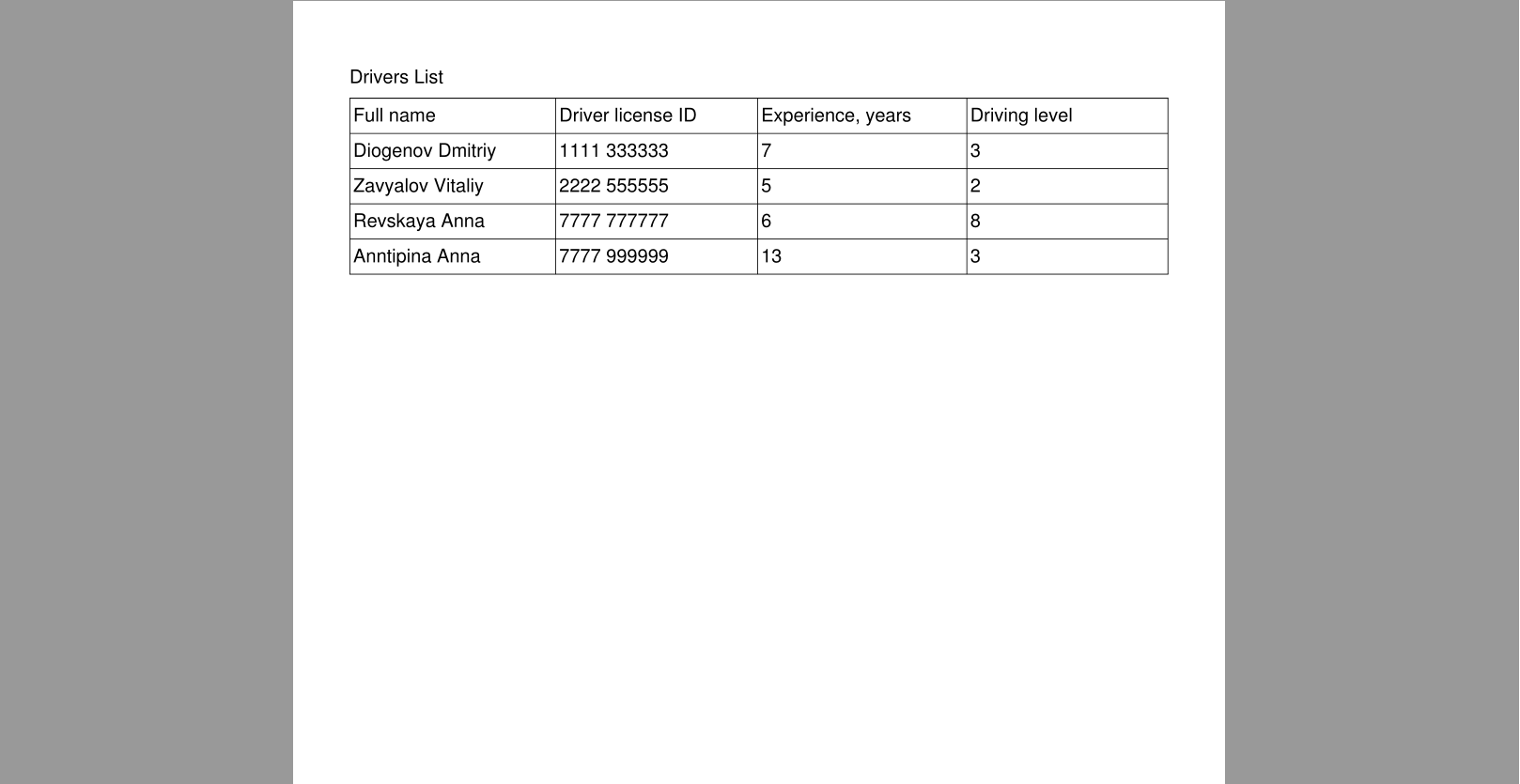


Рисунок 9. Пример отчета со списком водителей.

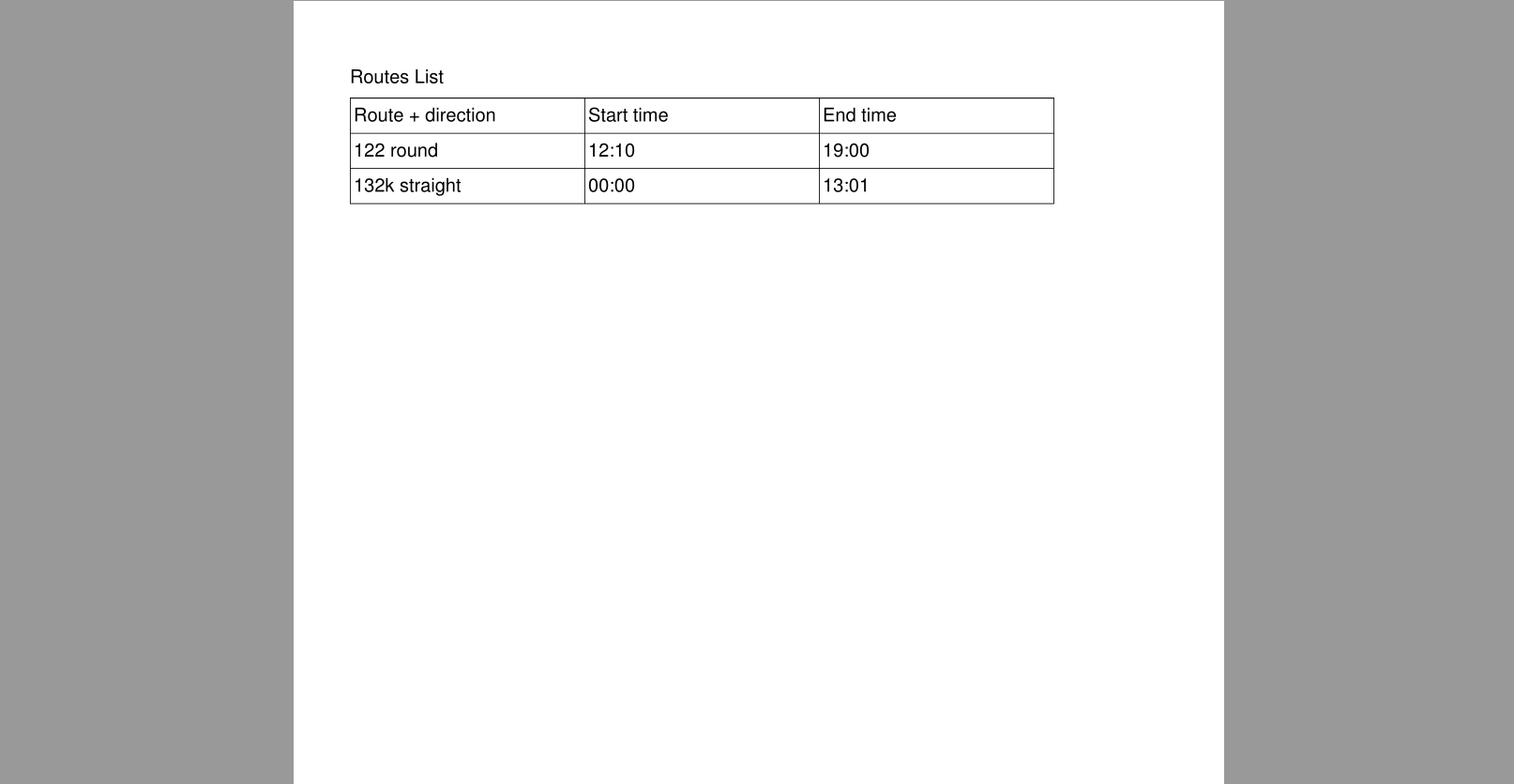


Рисунок 10. Пример отчета со списком маршрутов и их графиками.

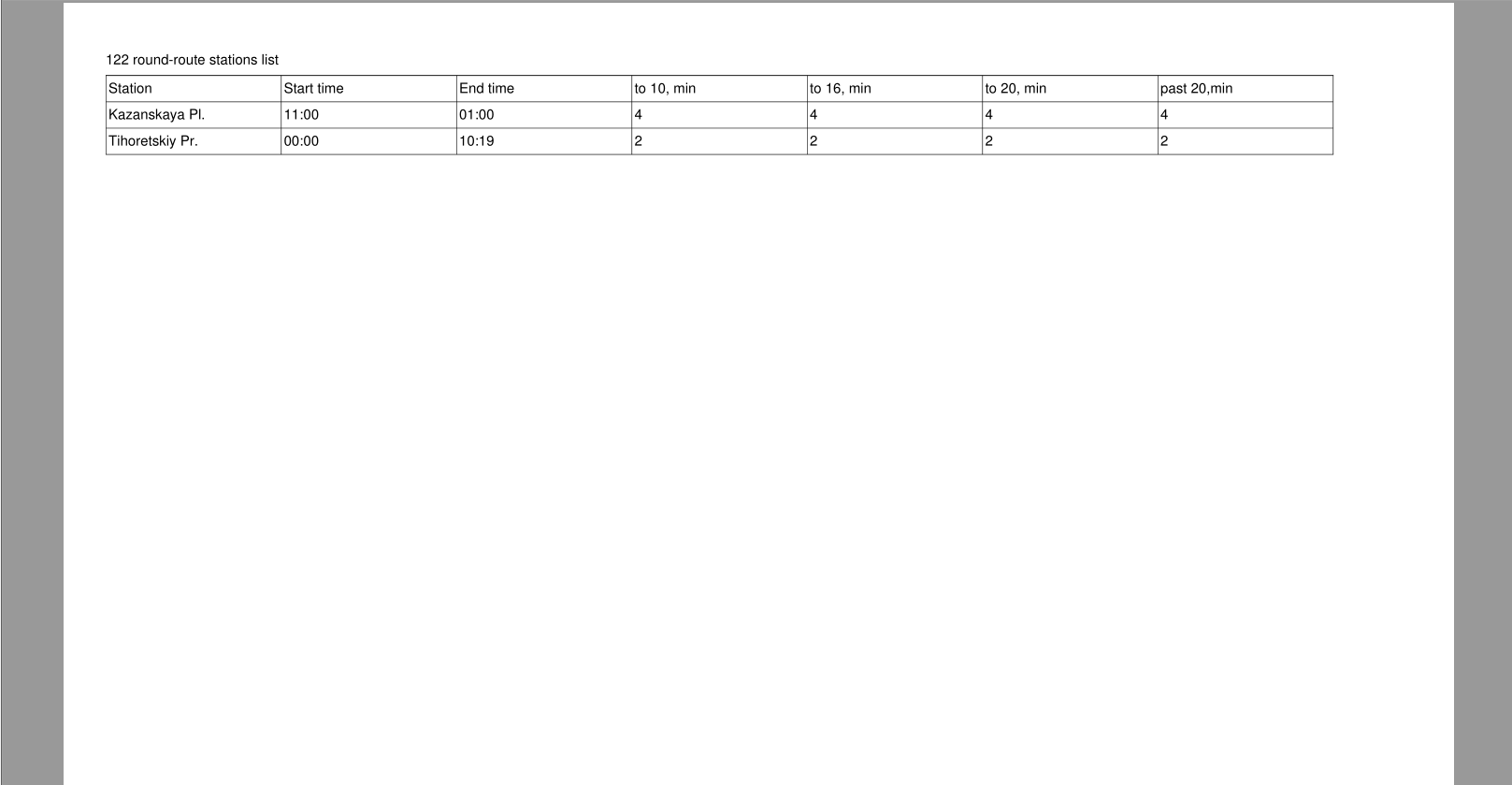


Рисунок 11. Пример отчета с графиком движения отдельного маршрута.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Далее будет представлена иллюстрированная инструкция для выполнения каждого из пунктов ТЗ, а также для некоторых других возможностей реализованной ПК.

1. Для формирования pdf-отчета

- выделить нужные строки таблицы

- нажать на кнопку для экспорта данных, изображенную с текстовой подсказкой

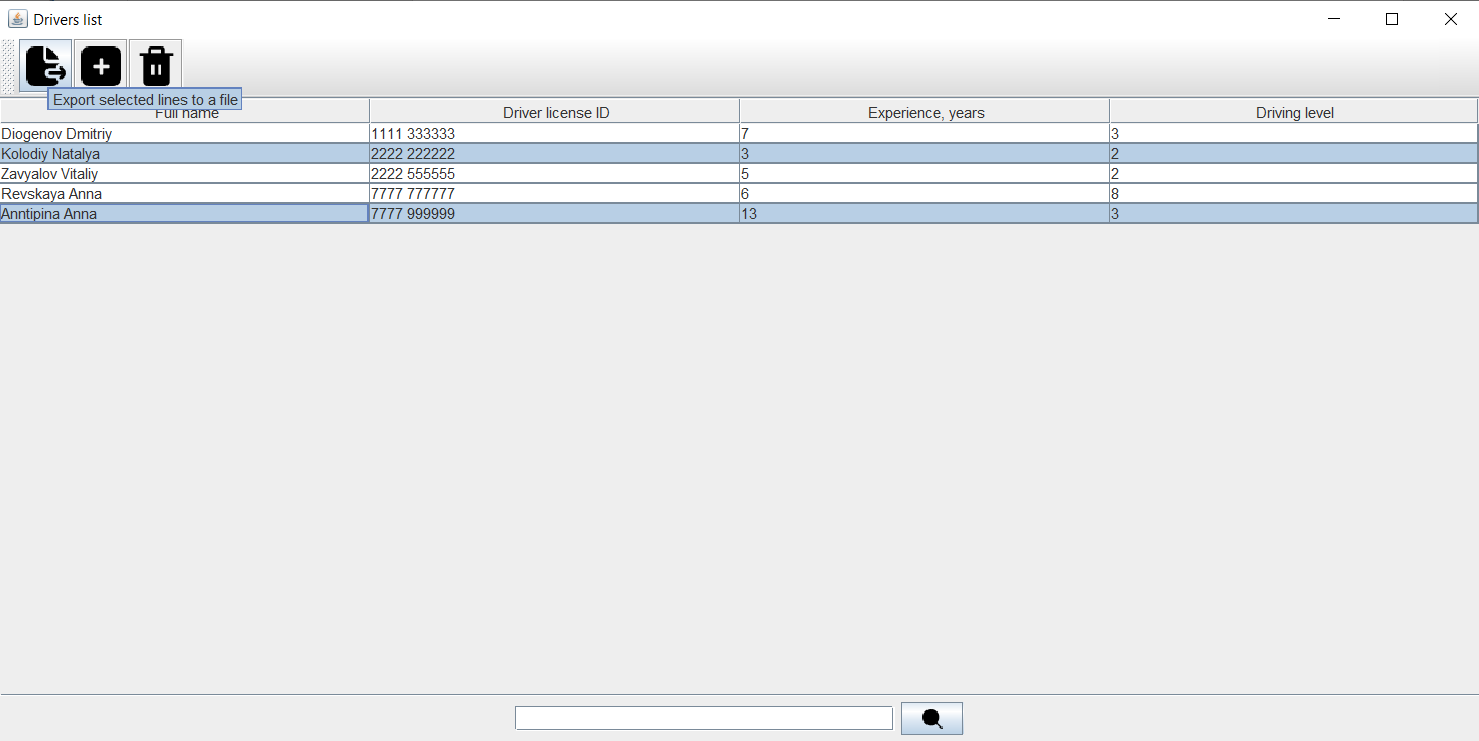


Рисунок 12. Выбор данных для выгрузки

- выбрать pdf-файл для загрузки или ввести новое название

- нажать на кнопку Save для сохранения выбора или Cancel для отмены выгрузки

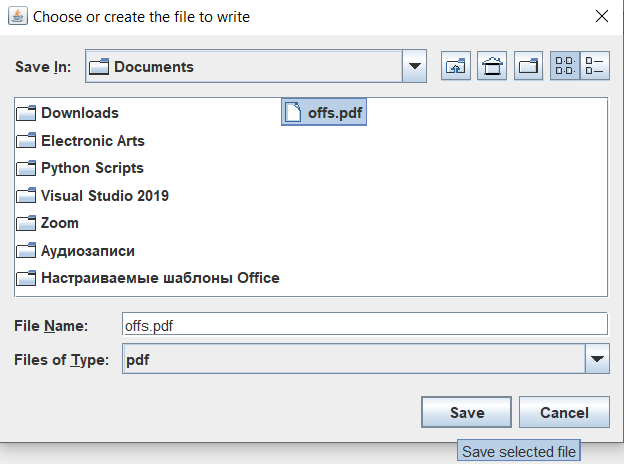


Рисунок 13. Выбор файла для выгрузки

- сформированный отчет будет доступен для просмотра по указанной директории

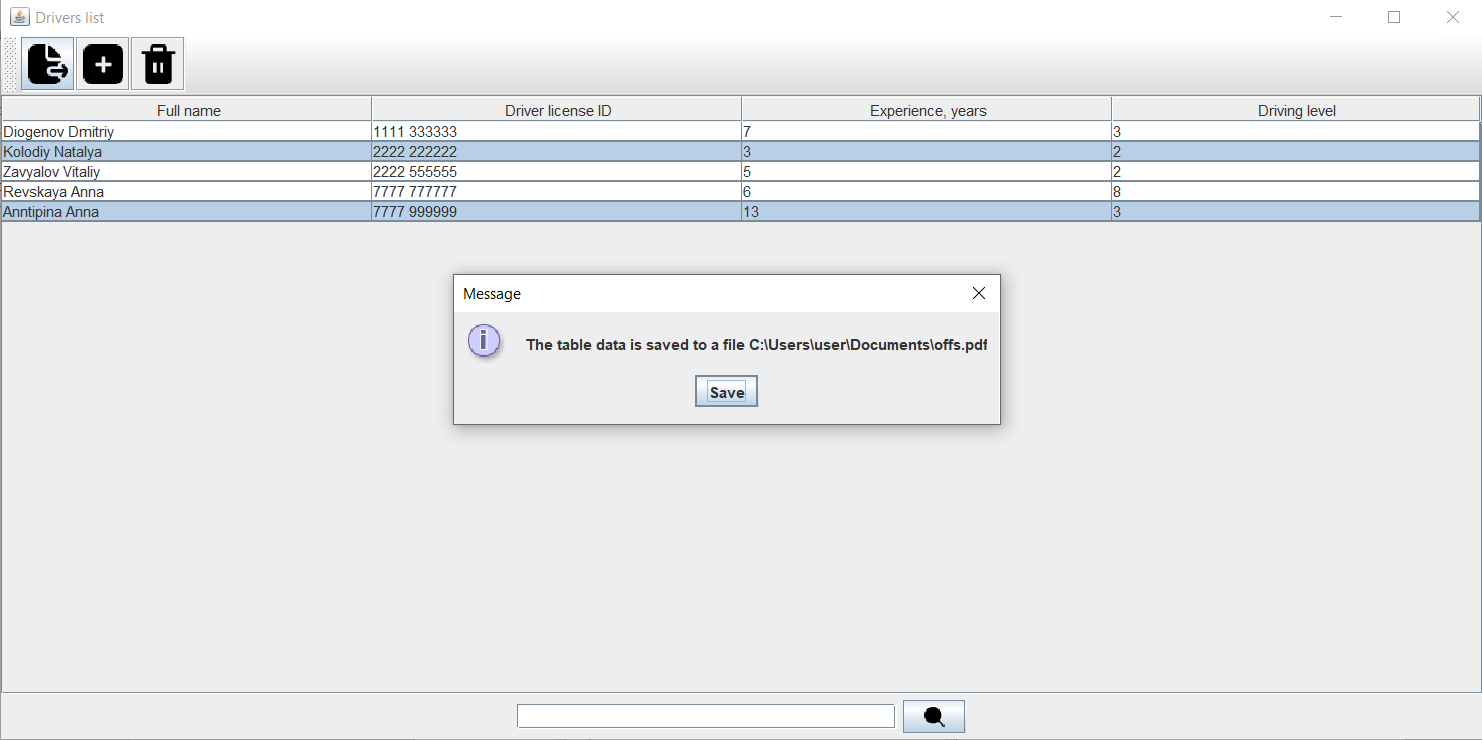


Рисунок 14. Результат выгрузки данных

Для остальных списков производятся аналогичные действия.

1. Для добавления новой строки в таблицу

- нажать на кнопку, изображенную с текстовой подсказкой

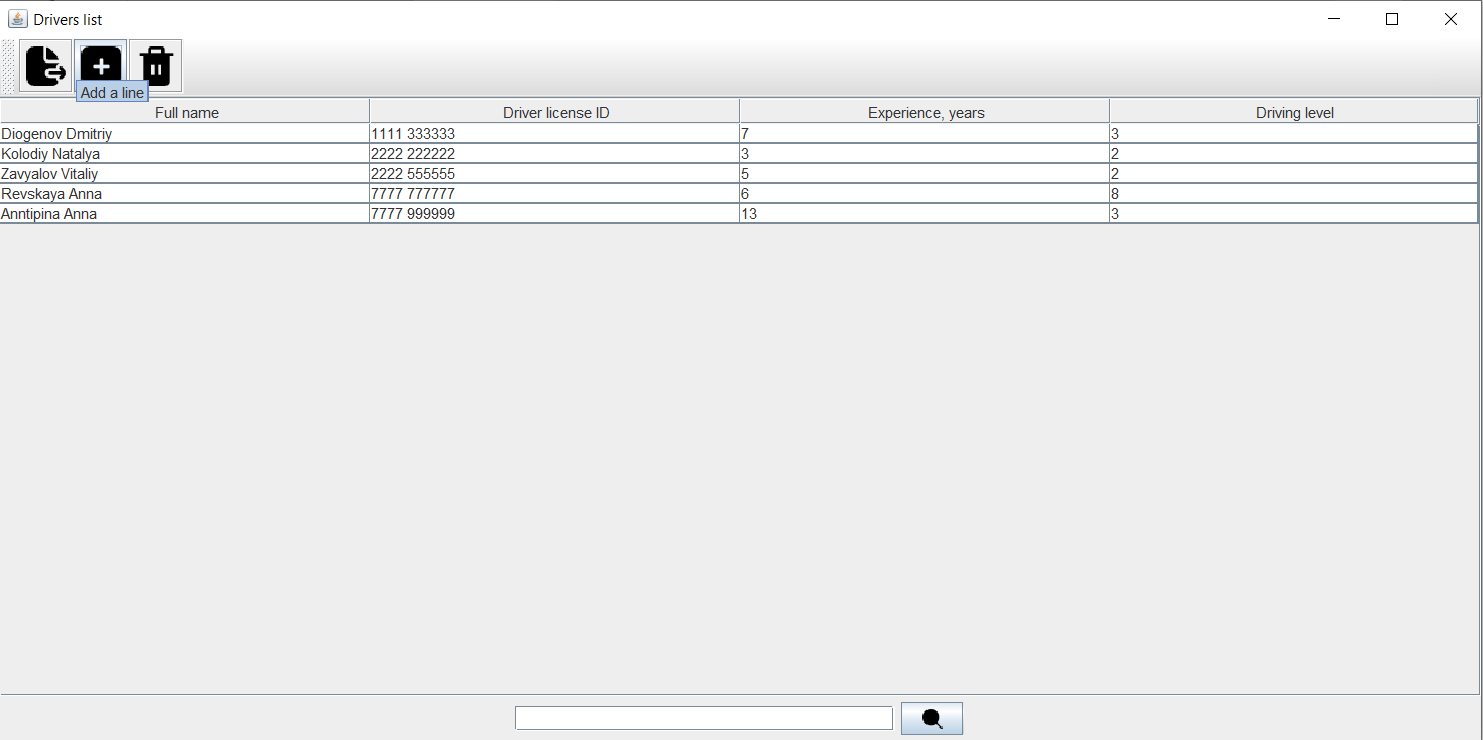


Рисунок 15. Выбор опции добавления новой строки

- заполнить поля, обращая внимание на допустимый формат, если он указан в скобках

- нажать на кнопку Save для сохранения или Cancel для отмены добавления

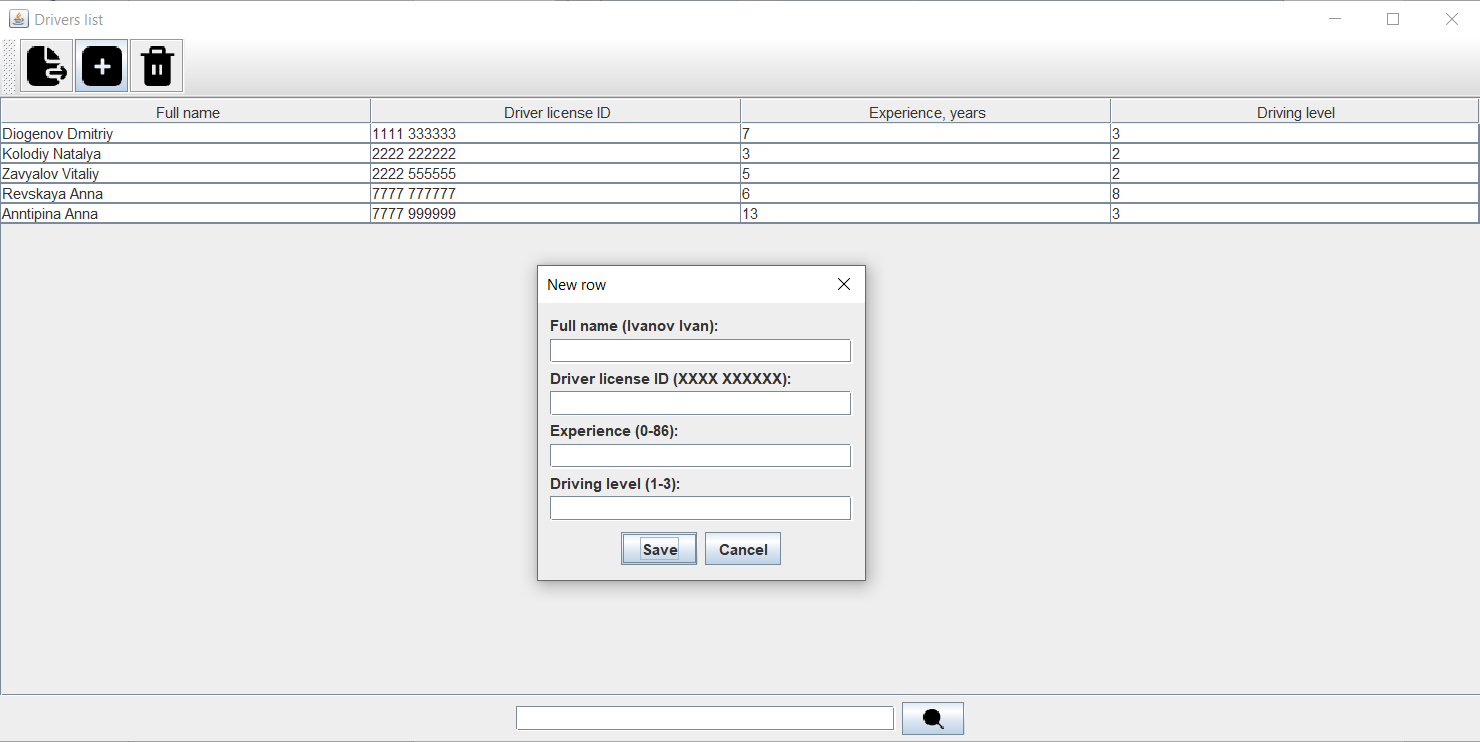


Рисунок 16. Заполнение полей

- чтобы добавить строку в список нарушений, номер прав водителя предлагается выбрать из уже существующих в списке водителей, как и номер маршрута предлагается выбрать из уже существующих в списке маршрутов

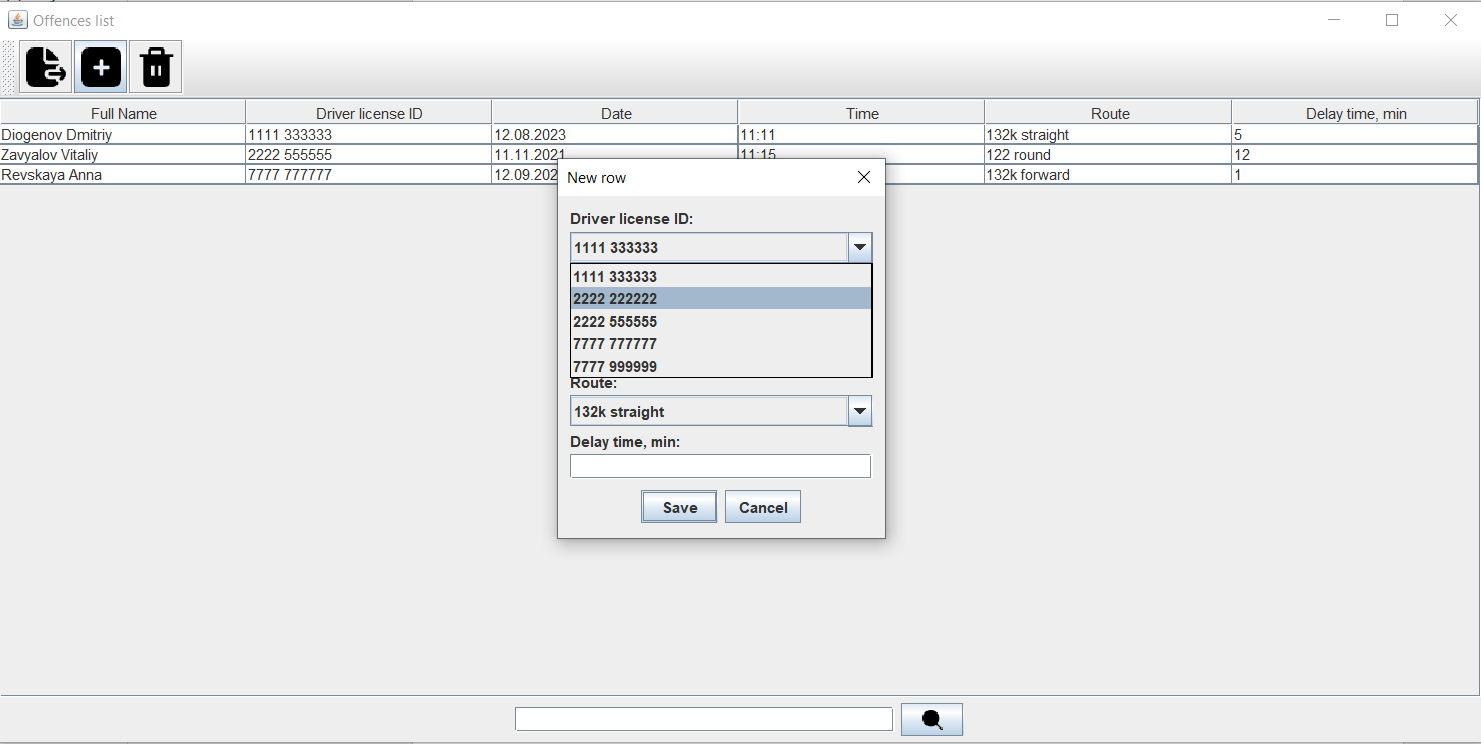


Рисунок 17. Выбор номера прав водителя из доступных.

- фамилия и имя после выбора прав водителя подставляются автоматически

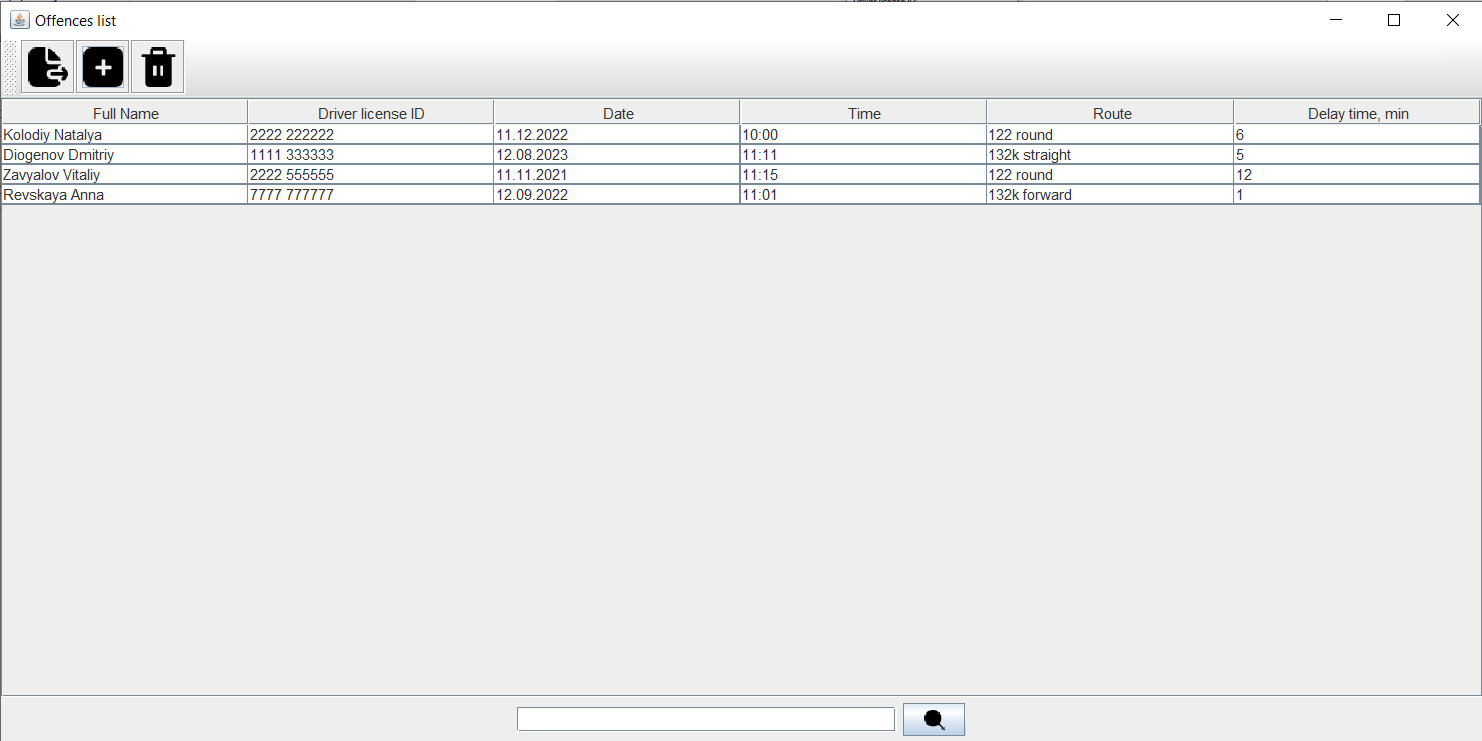


Рисунок 18. Резудьтат добавления новой строки с водительскими правами 2222 222222.

1. Для удаления строки из таблицы

- выделить нужные строки таблицы

- нажать на кнопку для удаления данных,изображенную с текстовой подсказкой

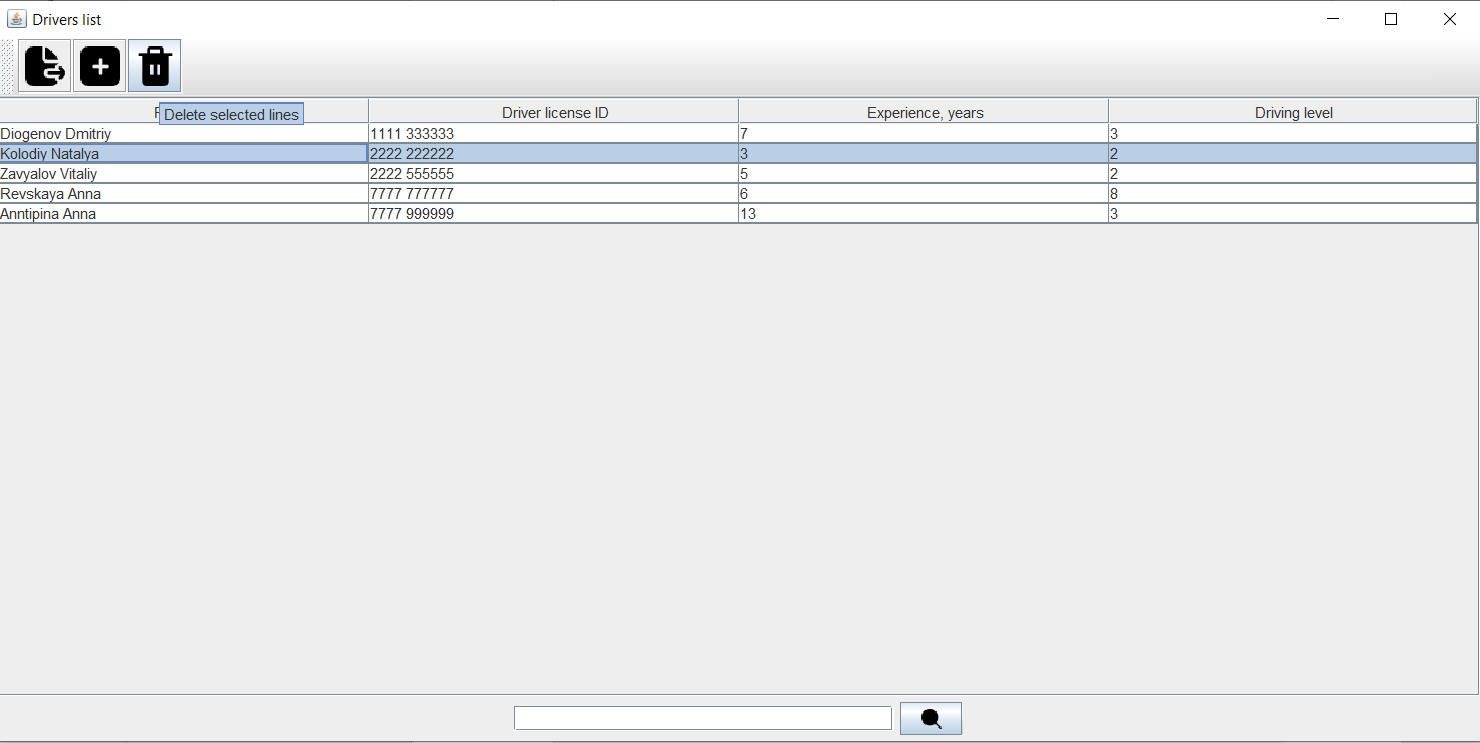


Рисунок 19. Выбор данных для удаления

- нажать на кнопку Confirm для подтверждения удаления или Cancel для отмены

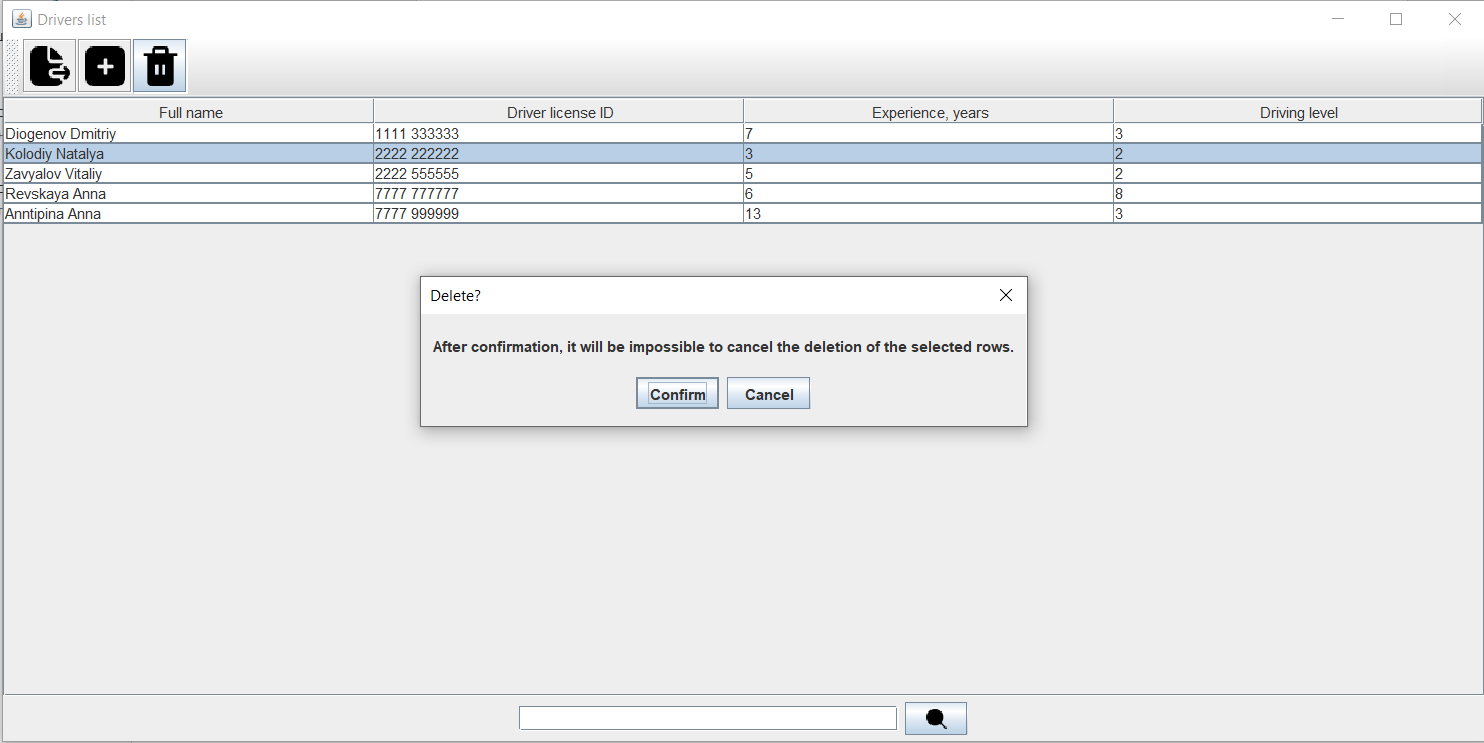


Рисунок 20. Подтверждение удаления

- выбранные строки будут удалены

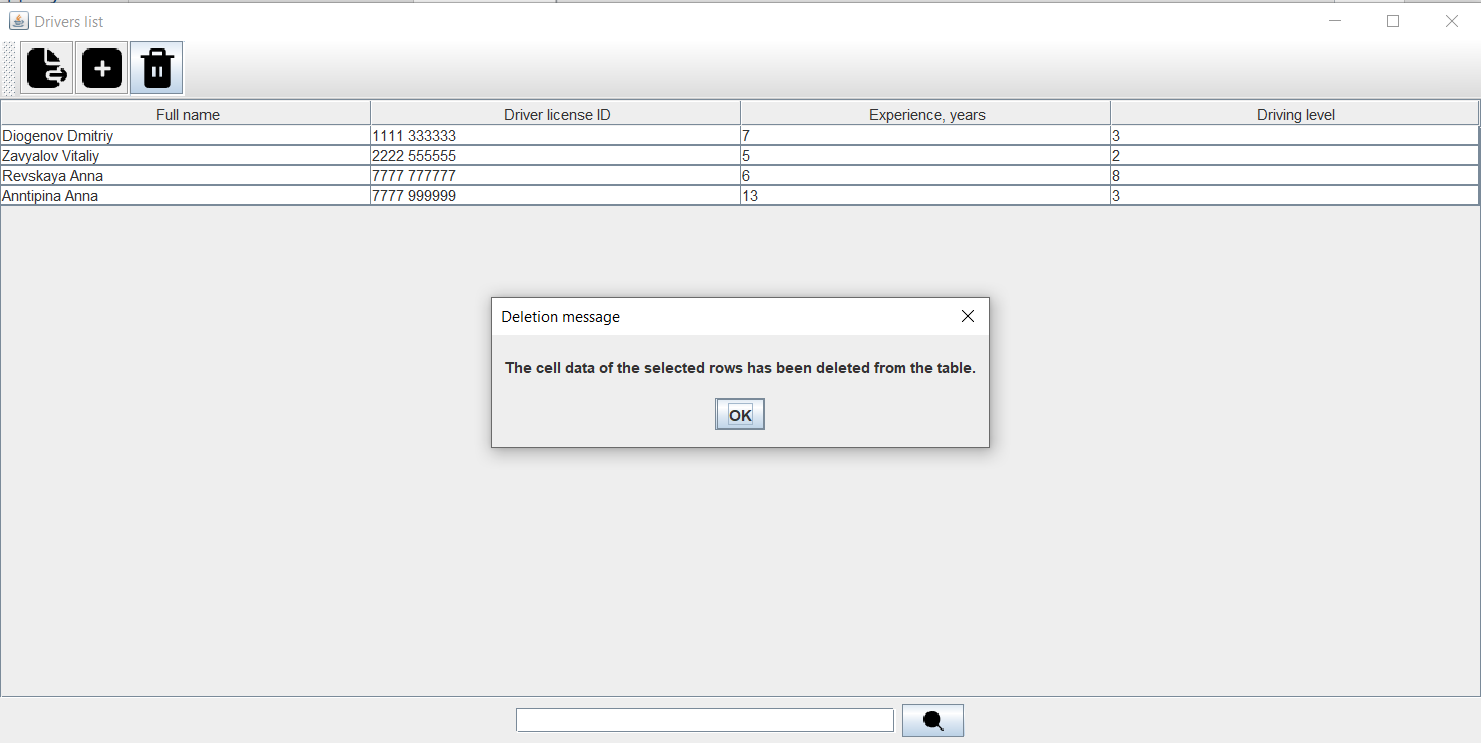


Рисунок 21. Результат удаления данных

- также будут удалены все строки в списке нарушений, принадлежащие соответствующему удаленному водителю

- при удалении маршрута из списка маршрутов связанные с ним остановки, содержащие график его движения, будут также удалены

4. Для редактирования ячеек таблицы

- выбрать строку, нажать на ячейку для появления всплывающей кнопки Edit the cell

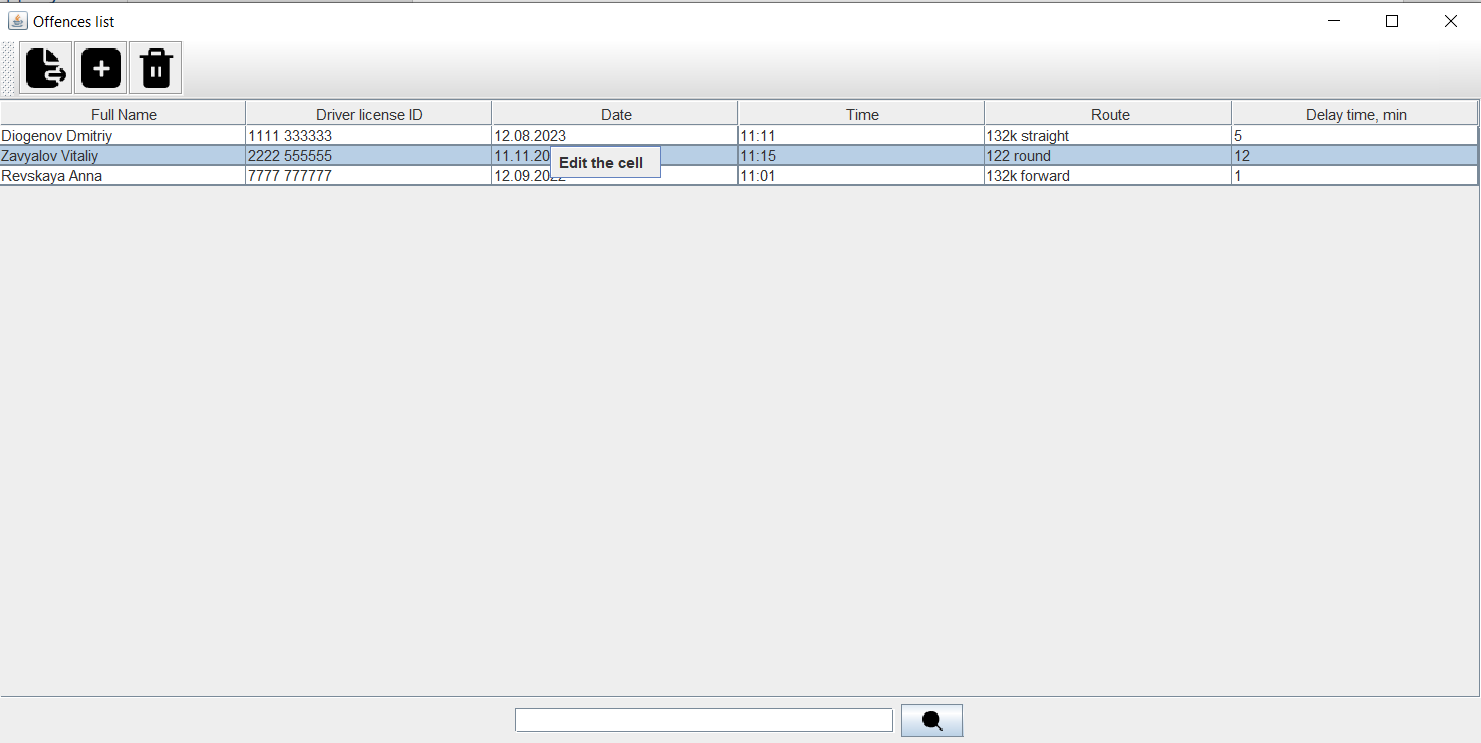


Рисунок 22. Выбор ячейки для редактирования

- нажать на кнопку Edit the cell и заполнить поле редактирования, обращая внимание на допустимый формат, если он указан в скобках

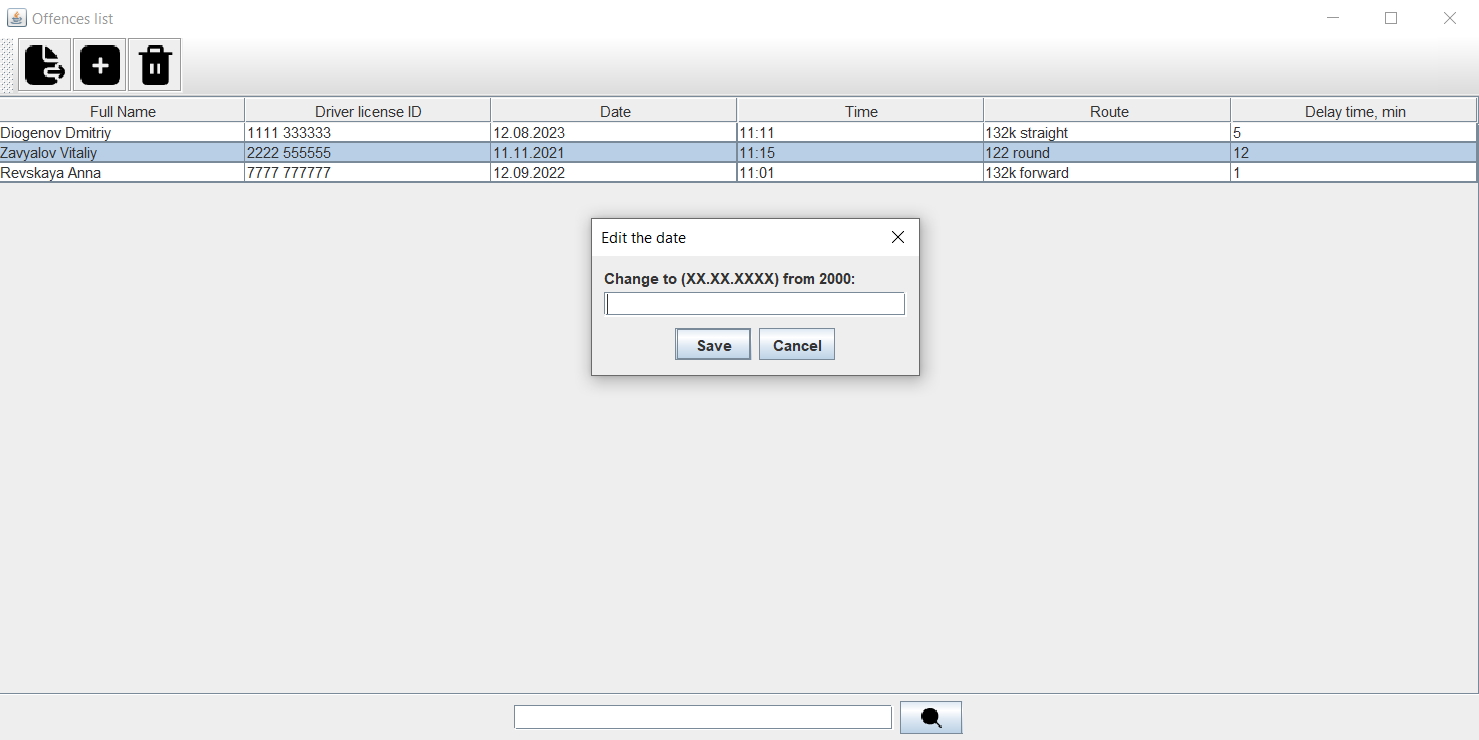


Рисунок 23. Заполнение поля ячейки при редактировании

- также доступно редактирование поля с кнопкой перехода к остановкам в таблице с маршрутами

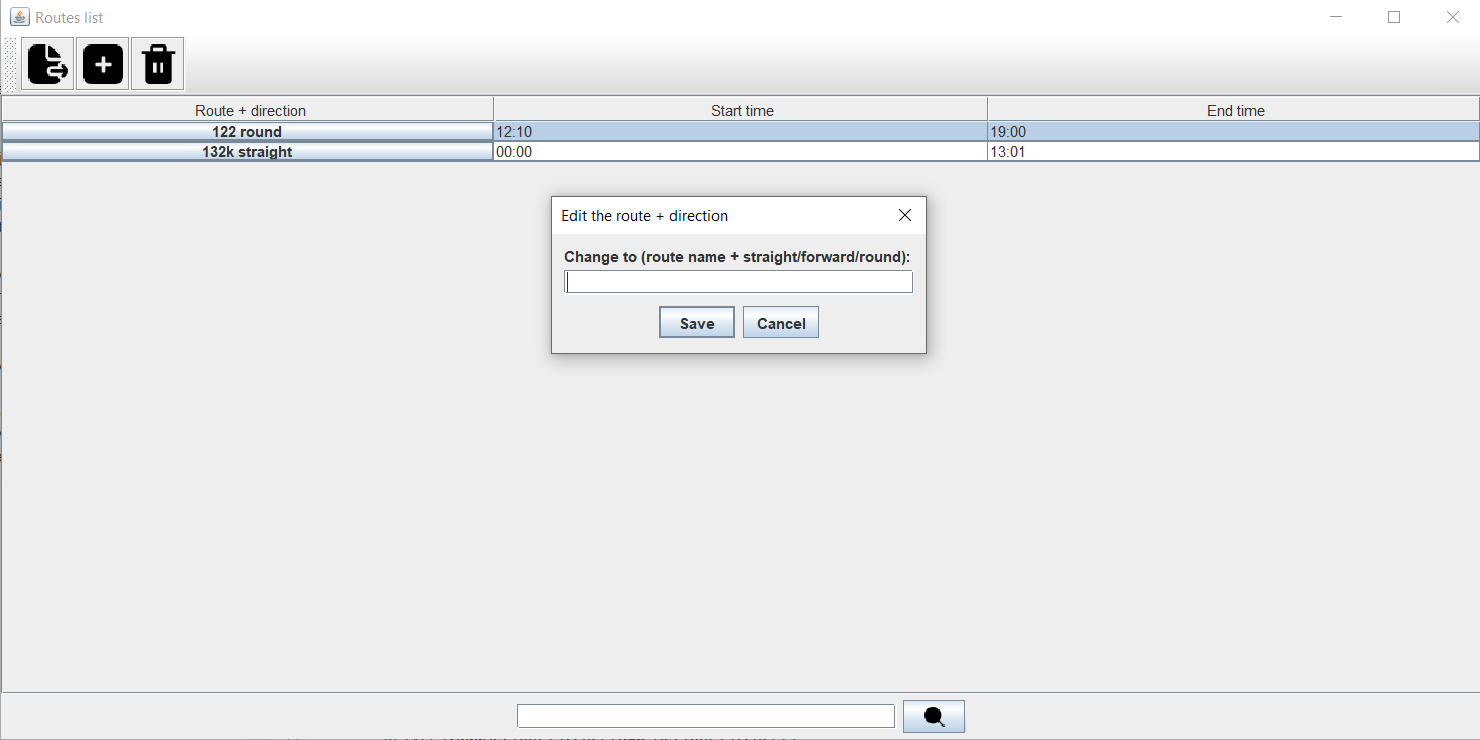


Рисунок 24. Редактировании кнопки перехода к остановкам

- в таблице маршрутов поле с фамилией и именем заполняется автоматически после выбора номера прав при редактировании ячейки Driver license ID

5. Для поиска по таблице

- ввести частично или полностью желаемое значение в строку поиска и нажать на кнопку поиска, изображенную с текстовой подсказкой

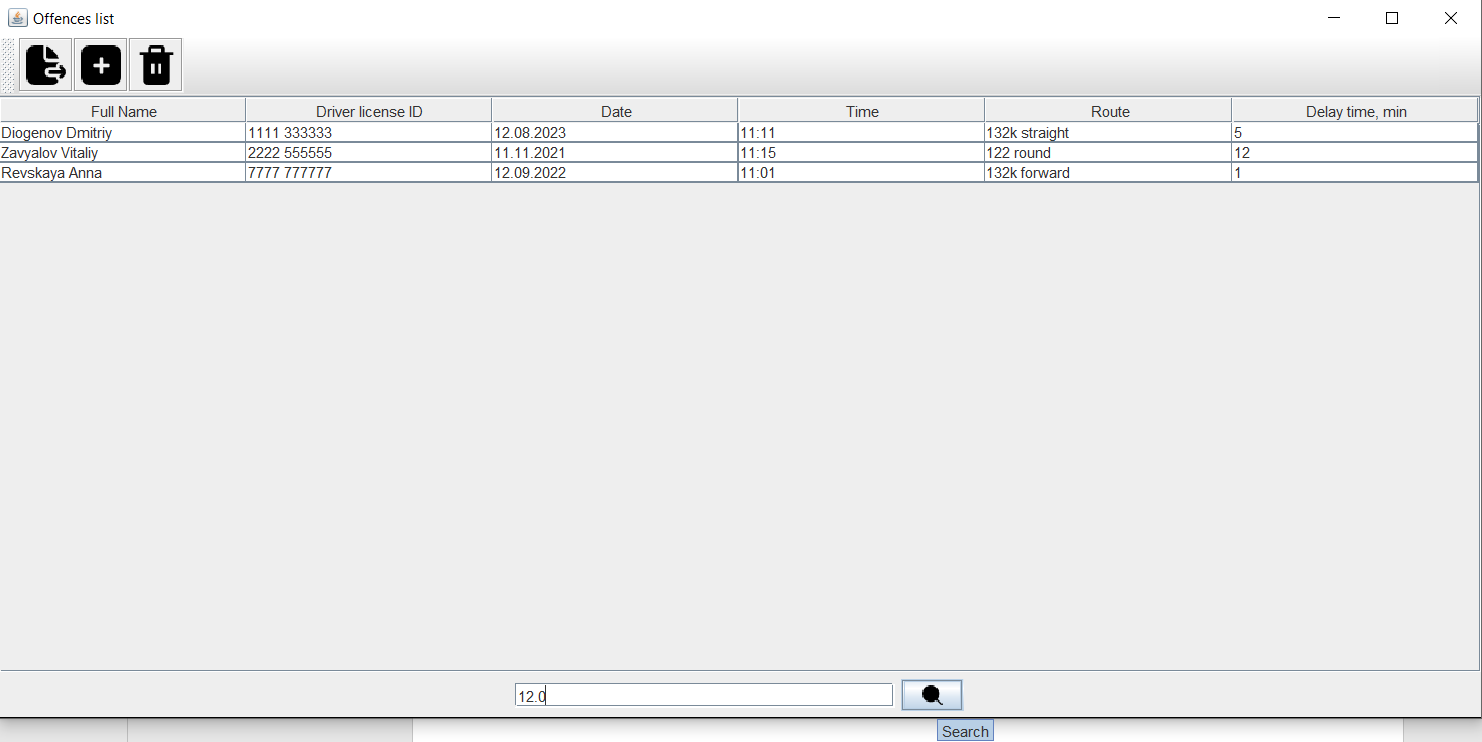


Рисунок 25. Ввод в строку поиска

- результат поиска будет представлен в следующем виде

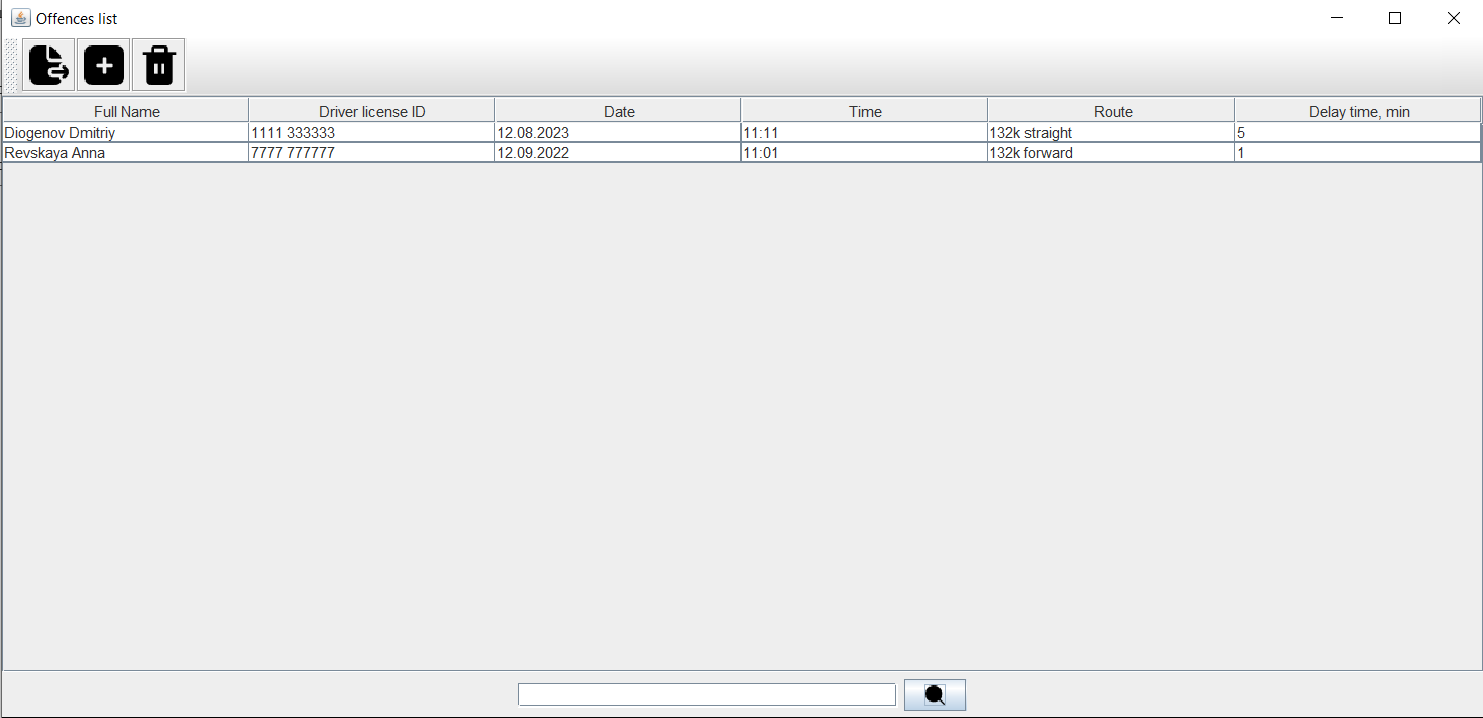


Рисунок 26. Результат поиска

- для возврата повторно нажмите на значок поиска и пропустите сообщение о пустой строке ввода

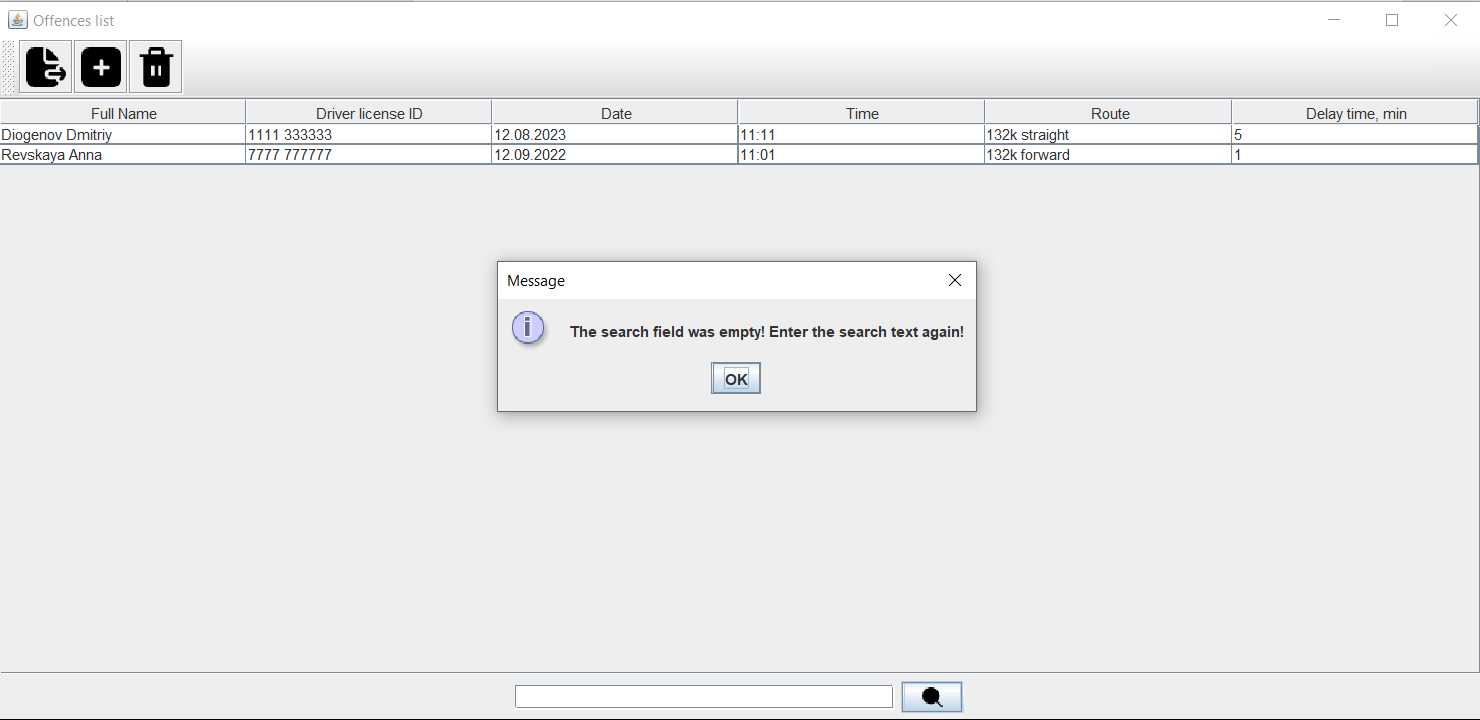


Рисунок 27. Поиск по пустой строке

- в таблице маршрутов поле с фамилией и именем заполняется автоматически после выбора номера прав при редактировании ячейки Driver license ID

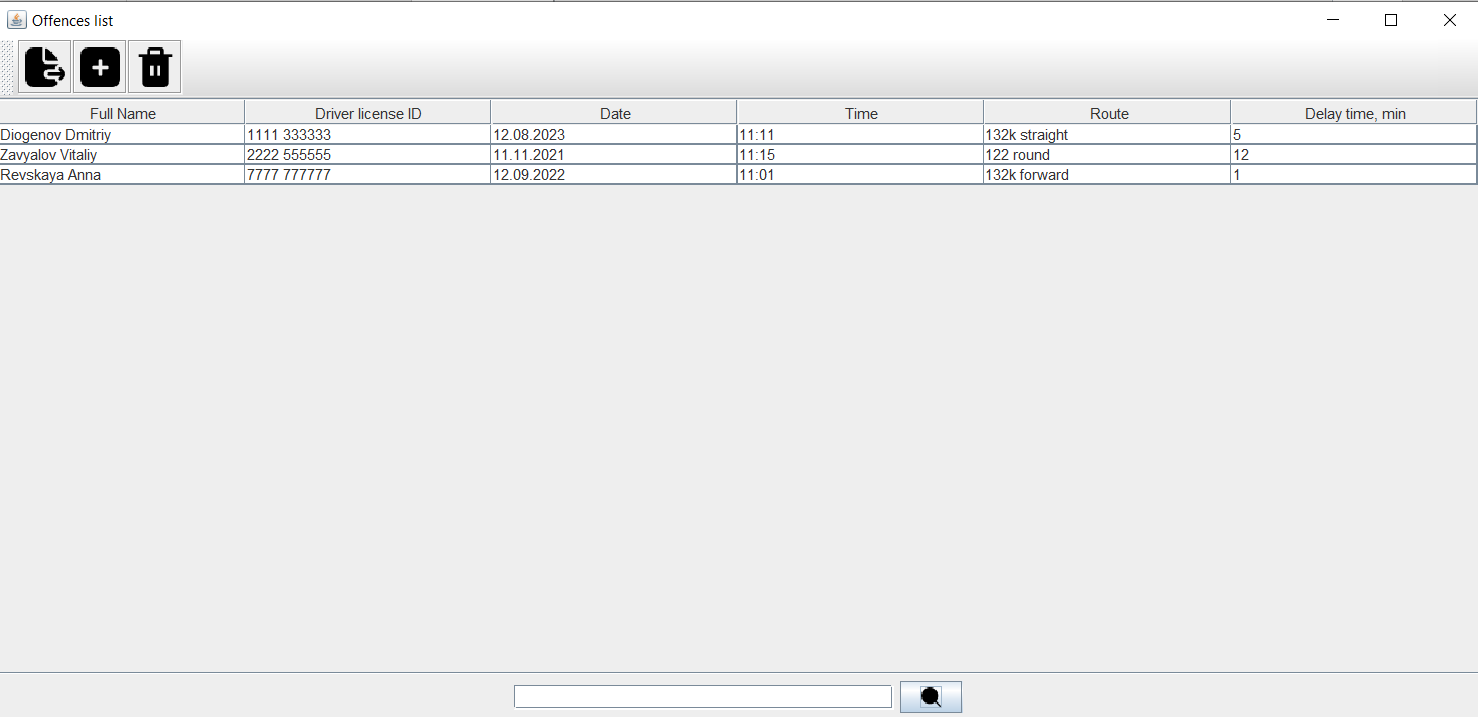


Рисунок 28. Результат возврата

6. Для перехода к списку остановок маршрута и графику его движения

- нажать на кнопку с маршрутом и его направлением в колонке Route + direction

- straight – в прямую сторону, forward – в обратную, round – в обе стороны одинаковый маршрут

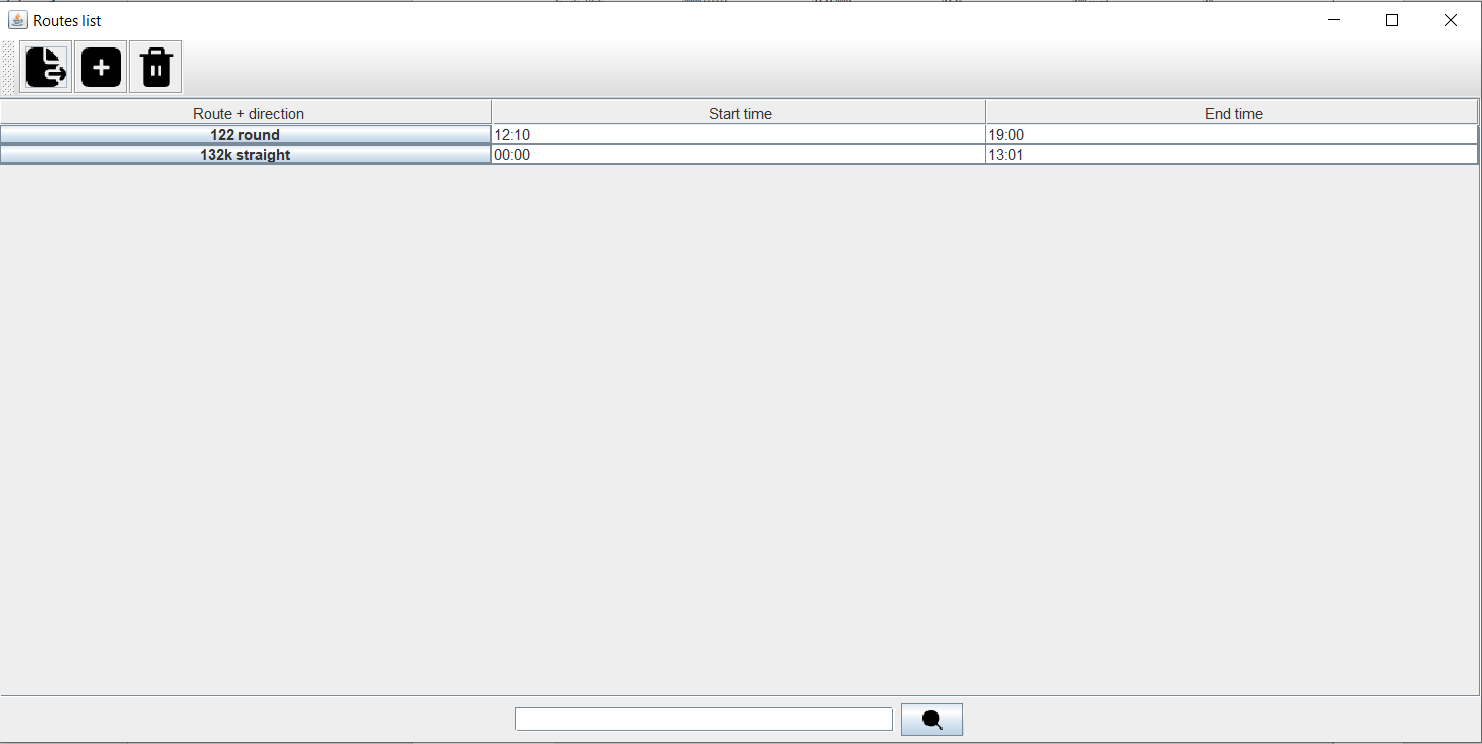


Рисунок 29. Список маршрутов и их направлений

- получение списка остановок и интервалов движения

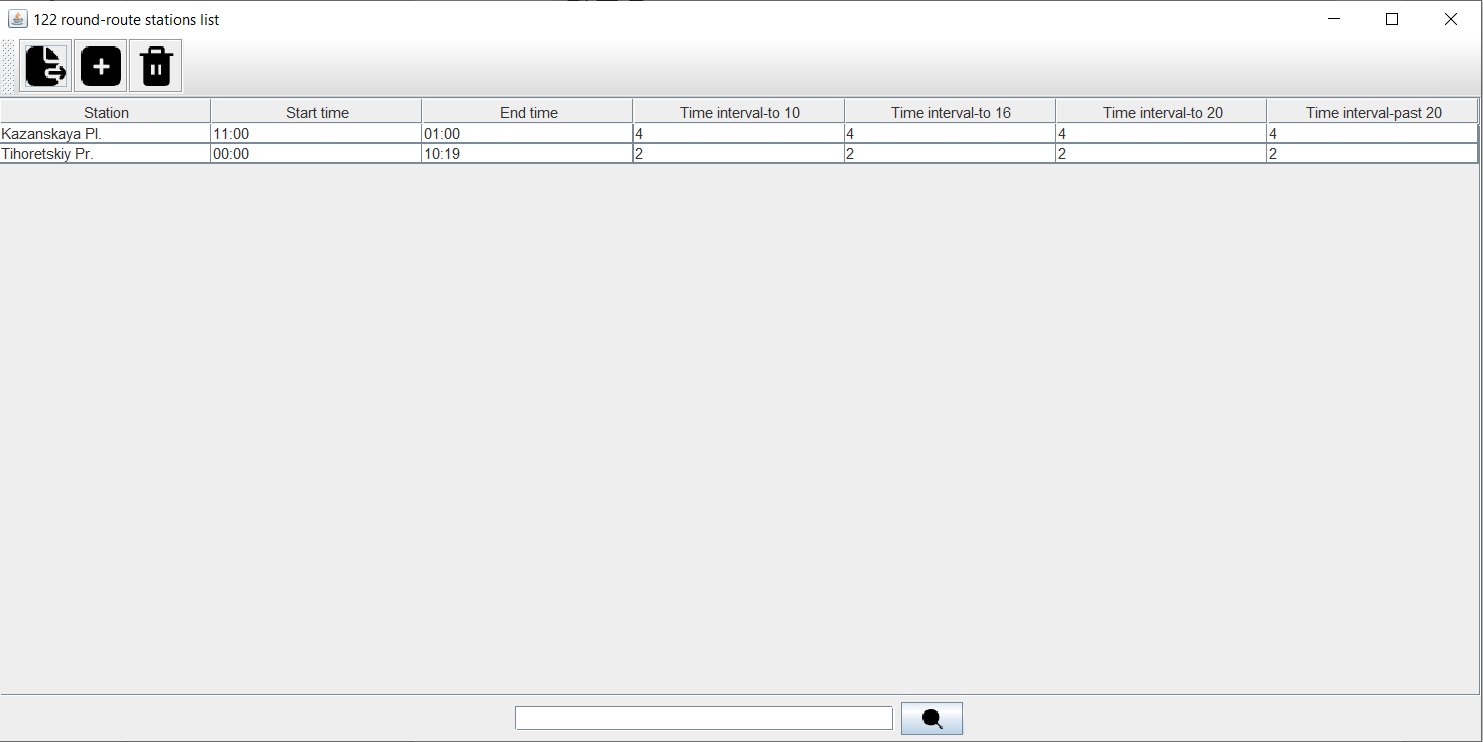


Рисунок 30. Список остановок после нажатия кнопки 122 round

7. Для закрытия окна

- нажать на крестик в правом верхнем углы любой таблицы и подтвердить закрытие

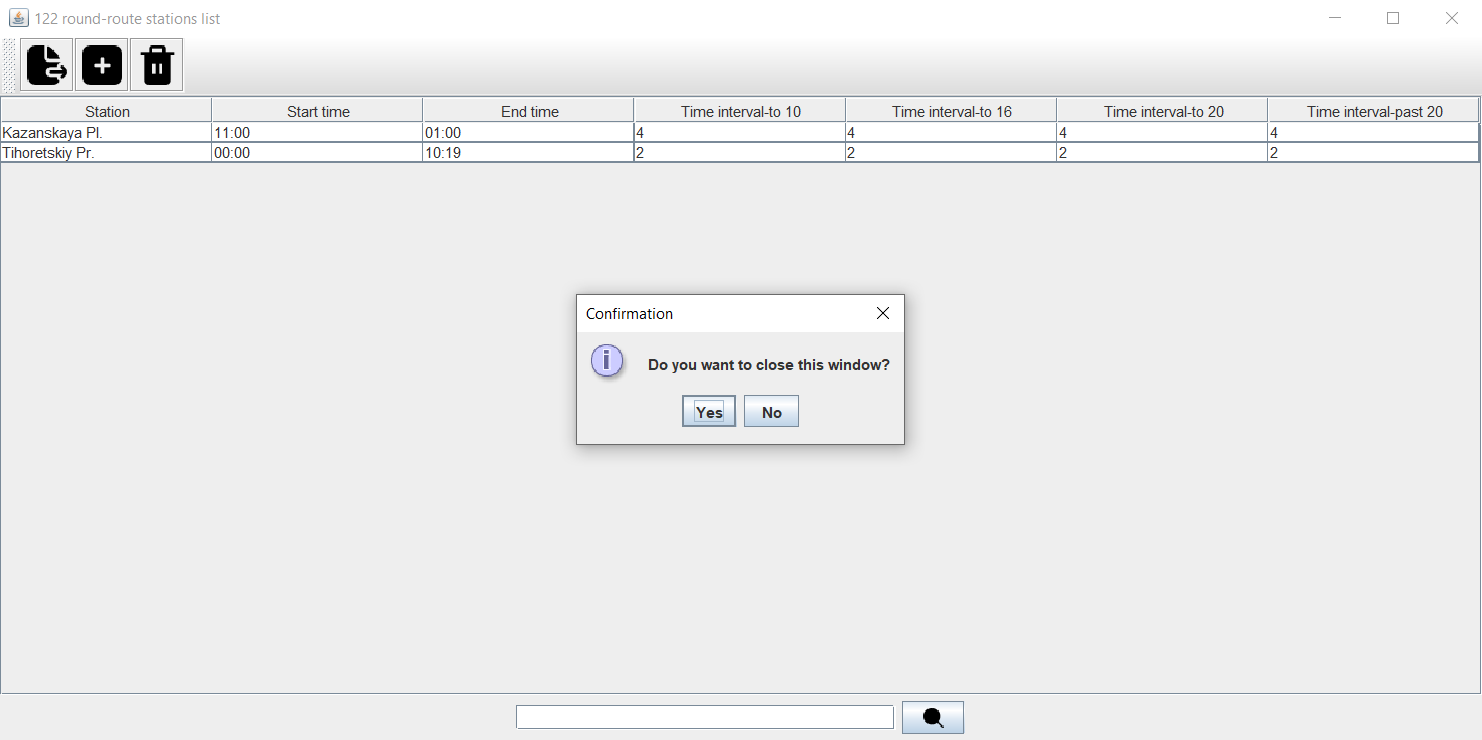


Рисунок 31. Подтверждение закрытия экрана

# ПОЛНЫЙ КОД ПРОГРАММЫ

См. приложение Б.

# Приложение А

**Код классов-сущностей**

1. **EntityDriverList**

package back\_entities;

import java.util.ArrayList;

import java.util.List;

import back\_database.DBOperations;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.FetchType;

import jakarta.persistence.Id;

import jakarta.persistence.OneToMany;

import jakarta.persistence.Table;

@Entity

@Table(name="driver")

public class EntityDriverList {

//id водителя = номер прав

@Id

@Column(name="id")

private int driver\_id;

//Ф.И.О

@Column(name="name")

private String name;

//права

@Column(name="license")

private String license;

//стаж работы

@Column(name="experience")

private String exp;

//класс

@Column(name="level")

private String lvl;

//массив штрафов

@OneToMany(fetch = FetchType.LAZY, mappedBy = "id\_driver", cascade = CascadeType.ALL)

private List<EntityOffenceList> Offences = new ArrayList<>();

//конструктор класса

public EntityDriverList(String name, String license, String exp, String lvl) {

this.name = name;

this.license = license;

this.exp = exp;

this.lvl = lvl;

}

// дефолтный конструктор класса

public EntityDriverList() {

this.name = "";

this.license = "";

this.exp = "";

this.lvl = "";

}

//геттеры

public int getId() {

return driver\_id;

}

public String getName() {

return name;

}

public String getLicense() {

return license;

}

public String getExp() {

return exp;

}

public String getLvl() {

return lvl;

}

// сеттеры

public void setName(String new\_name) {

this.license = new\_name;

}

public void setLicense(String new\_license) {

this.license = new\_license;

}

public void setExp(String new\_exp) {

this.exp = new\_exp;

}

public void setLvl(String new\_lvl) {

this.lvl = new\_lvl;

}

//геттер списка штрафов

public List<EntityOffenceList> getOffs() {

Offences = DBOperations.getOffences(license);

return Offences;

}

}

1. **EntityOffenceList**

package back\_entities;

import java.util.ArrayList;

import java.util.List;

import back\_database.DBOperations;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.FetchType;

import jakarta.persistence.Id;

import jakarta.persistence.OneToMany;

import jakarta.persistence.Table;

@Entity

@Table(name="driver")

public class EntityDriverList {

//id водителя = номер прав

@Id

@Column(name="id")

private int driver\_id;

//Ф.И.О

@Column(name="name")

private String name;

//права

@Column(name="license")

private String license;

//стаж работы

@Column(name="experience")

private String exp;

//класс

@Column(name="level")

private String lvl;

//массив штрафов

@OneToMany(fetch = FetchType.LAZY, mappedBy = "id\_driver", cascade = CascadeType.ALL)

private List<EntityOffenceList> Offences = new ArrayList<>();

//конструктор класса

public EntityDriverList(String name, String license, String exp, String lvl) {

this.name = name;

this.license = license;

this.exp = exp;

this.lvl = lvl;

}

// дефолтный конструктор класса

public EntityDriverList() {

this.name = "";

this.license = "";

this.exp = "";

this.lvl = "";

}

//геттеры

public int getId() {

return driver\_id;

}

public String getName() {

return name;

}

public String getLicense() {

return license;

}

public String getExp() {

return exp;

}

public String getLvl() {

return lvl;

}

// сеттеры

public void setName(String new\_name) {

this.license = new\_name;

}

public void setLicense(String new\_license) {

this.license = new\_license;

}

public void setExp(String new\_exp) {

this.exp = new\_exp;

}

public void setLvl(String new\_lvl) {

this.lvl = new\_lvl;

}

//геттер списка штрафов

public List<EntityOffenceList> getOffs() {

Offences = DBOperations.getOffences(license);

return Offences;

}

}

1. **EntityRouteList**

package back\_entities;

import java.util.ArrayList;

import java.util.List;

import back\_database.DBOperations;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.FetchType;

import jakarta.persistence.Id;

import jakarta.persistence.OneToMany;

import jakarta.persistence.Table;

@Entity

@Table(name="driver")

public class EntityDriverList {

//id водителя = номер прав

@Id

@Column(name="id")

private int driver\_id;

//Ф.И.О

@Column(name="name")

private String name;

//права

@Column(name="license")

private String license;

//стаж работы

@Column(name="experience")

private String exp;

//класс

@Column(name="level")

private String lvl;

//массив штрафов

@OneToMany(fetch = FetchType.LAZY, mappedBy = "id\_driver", cascade = CascadeType.ALL)

private List<EntityOffenceList> Offences = new ArrayList<>();

//конструктор класса

public EntityDriverList(String name, String license, String exp, String lvl) {

this.name = name;

this.license = license;

this.exp = exp;

this.lvl = lvl;

}

// дефолтный конструктор класса

public EntityDriverList() {

this.name = "";

this.license = "";

this.exp = "";

this.lvl = "";

}

//геттеры

public int getId() {

return driver\_id;

}

public String getName() {

return name;

}

public String getLicense() {

return license;

}

public String getExp() {

return exp;

}

public String getLvl() {

return lvl;

}

// сеттеры

public void setName(String new\_name) {

this.license = new\_name;

}

public void setLicense(String new\_license) {

this.license = new\_license;

}

public void setExp(String new\_exp) {

this.exp = new\_exp;

}

public void setLvl(String new\_lvl) {

this.lvl = new\_lvl;

}

//геттер списка штрафов

public List<EntityOffenceList> getOffs() {

Offences = DBOperations.getOffences(license);

return Offences;

}

}

1. **EntityStationrList**

package back\_entities;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.FetchType;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.Table;

@Entity

@Table(name="station")

public class EntityStationList {

//id остановки

@Id

@Column(name="id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int station\_id;

//название

@Column(name="name")

private String name;

//начало движения на остановке

@Column(name="start\_time")

private String time1;

//конец движения на остановке

@Column(name="end\_time")

private String time2;

//интервалы движения:

//до 10:00

@Column(name="to10")

private String int10;

//до 16:00

@Column(name="to16")

private String int16;

//до 20:00

@Column(name="to20")

private String int20;

//до 20:00

@Column(name="past20")

private String int00;

//id маршрута

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "id\_route")

private EntityRouteList id\_route;

//конструктор класса

public EntityStationList(String name, String time1, String time2, String int10, String int16, String int20, String int00, EntityRouteList route) {

this.name = name;

this.time1 = time1;

this.time2 = time2;

this.int10 = int10;

this.int16 = int16;

this.int20 = int20;

this.int00 = int00;

this.id\_route = route;

}

// дефолтный конструктор класса

public EntityStationList() {

this.name = "";

this.time1 = "";

this.time2 = "";

this.int10 = "";

this.int16 = "";

this.int20 = "";

this.int00 = "";

this.id\_route = null;

}

//геттеры

public int getId() {

return station\_id;

}

public String getName() {

return name;

}

public String getStart() {

return time1;

}

public String getEnd() {

return time2;

}

public String getInt1() {

return int10;

}

public String getInt2() {

return int16;

}

public String getInt3() {

return int20;

}

public String getInt4() {

return int00;

}

public EntityRouteList getRoute() {

return id\_route;

}

// сеттеры

public void setName(String new\_name) {

this.name = new\_name;

}

public void setStart(String new\_time1) {

this.time1 = new\_time1;

}

public void setEnd(String new\_time2) {

this.time2 = new\_time2;

}

public void setInt1(String new\_int1) {

this.int10 = new\_int1;

}

public void setInt2(String new\_int2) {

this.int16 = new\_int2;

}

public void setInt3(String new\_int3) {

this.int20 = new\_int3;

}

public void setInt4(String new\_int4) {

this.int00 = new\_int4;

}

public void setRoute(EntityRouteList new\_route) {

this.id\_route = new\_route;

}

}

# Приложение Б

**Полный код программы**

1. **GuiDriverList**

package front;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.List;

import javax.swing.DefaultRowSorter;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPopupMenu;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.JToolBar;

import javax.swing.RowFilter;

import javax.swing.UIManager;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import javax.swing.table.TableRowSorter;

import org.apache.log4j.Logger;

import com.itextpdf.kernel.PdfException;

import com.itextpdf.kernel.pdf.PdfDocument;

import com.itextpdf.kernel.pdf.PdfWriter;

import com.itextpdf.layout.Document;

import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.layout.element.Table;

import back\_database.DBOperations;

import back\_database.Singleton;

import back\_entities.EntityDriverList;

import jakarta.persistence.EntityManager;

public class GuiDriverList {

//фрейм

private JFrame frame;

//таблица и ее элементы

private DefaultTableModel model;

private String [] columns;

private String [][] data;

private JPopupMenu popupMenu;

private JMenuItem menuItemEdit;

private int [] selectedRows= {};

private TableRowSorter<TableModel> sorter;

//кнопки

private JButton Export;

private JButton Add;

private JButton Delete;

private JButton Search;

//панель инструментов

private JToolBar toolBar;

private JScrollPane scroll;

private JTable table;

private JTextField FilterText;

//заполнение ячейки текстом

String CellText;

private int row;

private int col;

//файловый экспорт

private JFrame exp\_frame;

private JFileChooser fileChooser;

private FileNameExtensionFilter filter;

//запись в pdf-файл

private String check\_pdf;

private PdfWriter writer;

private PdfDocument pdfDoc;

private Document doc;

private Table pdfTable;

//панель для строки поиска

private JPanel filterPanel;

//протоколирование

private static final Logger log = Logger.getLogger(GuiDriverList.class);

public GuiDriverList() {

log.info("Screen form creation.");

frame = new JFrame("Drivers list");

frame.setSize(1200, 600);

frame.setLocation(160, 100);

frame.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

Export = new JButton(new ImageIcon("./img/Export.png"));

Add = new JButton(new ImageIcon("./img/Add.png"));

Delete = new JButton(new ImageIcon("./img/Delete.png"));

Search = new JButton(new ImageIcon("./img/Search.png"));

Export.setToolTipText("Export selected lines to a file");

Add.setToolTipText("Add a line");

Delete.setToolTipText("Delete selected lines");

Search.setToolTipText("Search");

toolBar = new JToolBar("Toolbar");

toolBar.add(Export);

toolBar.add(Add);

toolBar.add(Delete);

frame.setLayout(new BorderLayout());

frame.add(toolBar, BorderLayout.NORTH);

//заполнение таблицы

columns = new String[] {"Full name", "Driver license ID","Experience, years","Driving level"};

getDBList();

model = new DefaultTableModel(data, columns);

//запрет на дефолтное редактирование ячеек

table = new JTable(model) {

public boolean isCellEditable(int row, int column) {

return false;

}

};

//всплывающее меню для редактирования ячейки в таблице

popupMenu = new JPopupMenu();

menuItemEdit = new JMenuItem("Edit the cell");

popupMenu.add(menuItemEdit);

//запрет на дефолтное перемещение столбцов таблицы

table.getTableHeader().setReorderingAllowed(false);

//сортировка по столбцам

sorter = new TableRowSorter<TableModel>(model);

table.setRowSorter(sorter);

scroll = new JScrollPane(table);

//скроллер

frame.add(scroll, BorderLayout.CENTER);

//строка поиска по столбцам

FilterText = new JTextField();

FilterText.setColumns(30);

filterPanel = new JPanel();

filterPanel.add(FilterText);

filterPanel.add(Search);

frame.add(filterPanel, BorderLayout.SOUTH);

// добавление слушателей

ActionListeners();

}

private void ActionListeners() {

//выделение строк

table.getSelectionModel().addListSelectionListener(new ListSelectionListener(){

public void valueChanged(ListSelectionEvent e) {

selectedRows = table.getSelectedRows();

table.setComponentPopupMenu(popupMenu);

}

});

//сохранение в файл

Export.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

try {

checkSelection(selectedRows);

log.info("Writing to a PDF file.");

log.info("Selecting a PDF file.");

exp\_frame = new JFrame();

fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Choose or create the file to write");

filter = new FileNameExtensionFilter("pdf","pdf");

fileChooser.setFileFilter(filter);

fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

int result = fileChooser.showSaveDialog(exp\_frame);

if (result == JFileChooser.APPROVE\_OPTION ) {

log.info("PDF file was selected.");

try {

check\_pdf = String.valueOf(fileChooser.getSelectedFile());

check\_pdf = remove\_substring(check\_pdf, ".pdf");

writer = new PdfWriter(check\_pdf + ".pdf");

pdfDoc = new PdfDocument (writer);

pdfDoc.addNewPage();

doc = new Document(pdfDoc);

float [] pointColumnWidths = {150F, 150F, 150F, 150F};

pdfTable = new Table(pointColumnWidths);

Paragraph paragraph = new Paragraph("Drivers List");

pdfTable.addCell("Full name");

pdfTable.addCell("Driver license ID");

pdfTable.addCell("Experience, years");

pdfTable.addCell("Driving level");

for (int rows = 0; rows < selectedRows.length; rows++) {

for (int cols = 0; cols < table.getColumnCount(); cols++) {

pdfTable.addCell(table.getModel().getValueAt(rows, cols).toString());

}

}

doc.add(paragraph);

doc.add(pdfTable);

doc.close();

JOptionPane.showMessageDialog(table,"The table data is saved to a file " + check\_pdf +".pdf");

log.info("The table data is saved to a file " + check\_pdf + ".pdf");

} catch (PdfException ex) {

ex.printStackTrace();

} catch (FileNotFoundException e1) {

JOptionPane.showMessageDialog(table,"File was not found.");

e1.printStackTrace();

}

}

else {

log.warn("To write data to a new file, you must enter its name and click the save button , otherwise writing to the file will not happen.");

log.info("Writing to the file has been canceled.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

myEx.printStackTrace();

}

}

});

//закрытие окна

frame.addWindowListener(new WindowAdapter()

{

public void windowClosing(WindowEvent event) {

Object[] options = { "Yes", "No" };

int option = JOptionPane.showOptionDialog(event.getWindow(), "Do you want to close this window?",

"Confirmation", JOptionPane.YES\_NO\_OPTION,

JOptionPane.INFORMATION\_MESSAGE, null, options,

options[0]);

if (option == 0) {

event.getWindow().setVisible(false);

new GuiMenu().show();

}

}

});

//поиск

Search.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

log.info("Table search.");

String text;

text = FilterText.getText();

try {

CheckTextField(text, 10);

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(text));

log.info("The elements were found successfully.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

log.info("Error - an empty input field in the search bar.", myEx);

JOptionPane.showMessageDialog(table, myEx.getMessage());

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(""));

}

finally {

FilterText.setText("");

}

}});

//редактирование ячеек

menuItemEdit.addActionListener(new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Editing a cell in a table row. ");

row = table.getSelectedRow();

col = table.getSelectedColumn();

String prevname = model.getValueAt(row, 0).toString();

String prevdriver = model.getValueAt(row, 1).toString();

String prevexp = model.getValueAt(row, 2).toString();

String prevlvl = model.getValueAt(row, 3).toString();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

switch (col) {

case(0):

CellText = JOptionPane.showInputDialog(table, "Change to (Ivanov Ivan): ","Edit the full name", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityDriverList Driver = DBOperations.searchDriverWithLicense(prevdriver);

Driver.setLicense(CellText);

em.merge(Driver);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevname, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(1):

CellText = JOptionPane.showInputDialog(table, "Change to (XXXX XXXXXX): ","Edit the license id", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

try {

checkRepeat(CellText);

EntityManager em = Singleton.createEMandTrans();

EntityDriverList Driver = DBOperations.searchDriverWithLicense(prevdriver);

Driver.setLicense(CellText);

em.merge(Driver);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The driver license id field reapeats.", myEx);

log.warn("The new cell will not be added.", myEx);

}

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevdriver, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(2):

CellText = JOptionPane.showInputDialog(table, "Change to (0-86 years): ","Edit the experience", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityDriverList Driver = DBOperations.searchDriverWithLicense(prevdriver);

Driver.setExp(CellText);

em.merge(Driver);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevexp, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(3):

CellText = JOptionPane.showInputDialog(table, "Change to (1-3): ","Edit the level", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityDriverList Driver = DBOperations.searchDriverWithLicense(prevdriver);

Driver.setLvl(CellText);

em.merge(Driver);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevlvl, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

}

}});

//добавление новой строки после выделенной

Add.addActionListener (new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Adding a new row to the table after the selected one.");

row = table.getSelectedRow() + 1;

JTextField add\_name = new JTextField();

JTextField add\_driver = new JTextField();

JTextField add\_exp = new JTextField();

JTextField add\_lvl = new JTextField();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

Object[] message = {

"Full name (Ivanov Ivan): ", add\_name,

"Driver license ID (XXXX XXXXXX): ", add\_driver,

"Experience (0-86): ", add\_exp,

"Driving level (1-3): ", add\_lvl

};

int option = JOptionPane.showConfirmDialog(table, message, "New row", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

if(option == JOptionPane.OK\_OPTION) {

int exeption\_sum = 0;

try{

log.info(row +"," + 0 + "-cell is being filled in.");

CheckTextField(add\_name.getText(), 0);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 0 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 1 + "-cell is being filled in.");

CheckTextField(add\_driver.getText(), 1);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 1 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 2 + "-cell is being filled in.");

CheckTextField(add\_exp.getText(), 2);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 2 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 3 + "-cell is being filled in.");

CheckTextField(add\_lvl.getText(), 3);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 3 + "-cell has been filled in wrong.");

}

try{

if(exeption\_sum > 0) throw new WrongAddException();

try {

checkRepeat(add\_driver.getText());

model.insertRow(row, new String [] {add\_name.getText(), add\_driver.getText(), add\_exp.getText(), add\_lvl.getText()});

EntityManager em = Singleton.createEMandTrans();

EntityDriverList newDriver = new EntityDriverList(add\_name.getText(), add\_driver.getText(), add\_exp.getText(), add\_lvl.getText());

em.persist(newDriver);

Singleton.finishEMandTrans(em);

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The driver license id field reapeats.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

catch(WrongAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The new line fields were filled in incorrectly.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

}

});

//удаление строк

Delete.addActionListener(new ActionListener(){

public void actionPerformed (ActionEvent event){

log.info("Deletion of selected rows from the table.");

try {

checkSelection(selectedRows);

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Confirm");

int option = JOptionPane.showConfirmDialog(table, "After confirmation, it will be impossible to cancel the deletion of the selected rows.", "Delete?", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

log.warn("After confirmation, it will be impossible to cancel the deletion of the selected rows.");

if (option == JOptionPane.OK\_OPTION) {

UIManager.put("OptionPane.okButtonText", "OK");

for(int i = selectedRows.length-1; i > -1; i--){

EntityManager em = Singleton.createEMandTrans();

EntityDriverList Driver = DBOperations.searchDriverWithLicense(String.valueOf(table.getModel().getValueAt(selectedRows[i], 1)));

em.merge(Driver);

Singleton.finishEMandTrans(em);

model.removeRow(selectedRows[i]);

DBOperations.deleteDriver(Driver.getId());

}

JOptionPane.showMessageDialog (frame, "The cell data of the selected rows has been deleted from the table.","Deletion message", JOptionPane.PLAIN\_MESSAGE);

log.info("The selected rows are removed from the table.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("Исключительная ситуация - нет выделенных строк для удаления. ", myEx);

log.warn("Необходимо выделить строки для удаления, иначе удаление будет недоступно. ");

}

}

});

}

//получение данных из БД для таблицы

private void getDBList() {

// получение списка из БД

List<EntityDriverList> Drivers = DBOperations.getSortedDrivers();

ArrayList<String[]> values = new ArrayList<String[]>(Drivers.size() + 1);

for (int i = 0; i < Drivers.size(); i++) {

EntityDriverList entity = Drivers.get(i);

values.add(new String[]{entity.getName(), entity.getLicense(),entity.getExp(),entity.getLvl()});

}

// преобразование в формат для заполнения таблицы данными.

data = new String[values.size()][4];

for (int i = 0; i < values.size(); i++) data[i] = values.get(i);

}

//подсчет подстрок в строке

public static int count\_substring(String str, String target) {

return (str.length() - str.replace(target, "").length()) / target.length();

}

//удаление подстрок в строке

public static String remove\_substring(String str, String substr) {

int count = count\_substring(str, substr);

if (count > 0) {

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

str = str.replace(substr, "");

}

return str;

}

//проверка введенного стажа

public int CheckExpNumber (String text){

int flag = 0;

if (Integer.valueOf(text) >= 0 && Integer.valueOf(text) <= 86 ) {

flag = 1;

}

return flag;

}

//проверка введенного класса вождения

public int CheckLvlNumber (String text){

int flag = 0;

if (Integer.valueOf(text) > 0 && Integer.valueOf(text) <= 3 ) {

flag = 1;

}

return flag;

}

//проверка повтора строки

private void checkRepeat(String str) throws RepeatAddException{

for (int i = 0; i < table.getRowCount(); i++) {

if (table.getModel().getValueAt(i, 1).equals(str)) {

throw new RepeatAddException();

}

}

}

public class RepeatAddException extends Exception {

public RepeatAddException() {

super("It is restricted to add a row with the same driver license id! Please, try again and change this field.");

}

}

//обработка заполнения текстового поля

public void CheckTextField (String text, int col) throws EmptyFieldException, WrongFieldException, WrongDateException, WrongTimeException, EmptySearchFieldException {

if (col == 0 && !text.matches("^[A-Z-][a-z-]\* [A-Z-][a-z-]\*$")) throw new WrongFieldException(col);

if (col == 1 && !text.matches("^\\d{4}\\s\\d{6}$")) throw new WrongFieldException(col);

if (col != 10 && (text.trim().isEmpty() || text == null) ) throw new EmptyFieldException(col);

if (col == 10 && (text.trim().isEmpty() || text == null)) throw new EmptySearchFieldException();

if (col == 2 && (!text.matches("^[1-9]\\d\*$") || CheckExpNumber(text) == 0)) throw new WrongFieldException(col);

if (col == 3 && (!text.matches("^[1-9]\\d\*$") || CheckLvlNumber(text) == 0)) throw new WrongFieldException(col);

}

//поле поиска пустое

public class EmptySearchFieldException extends Exception {

public EmptySearchFieldException() {

super("The search field was empty! Enter the search text again!");

}

}

//поле пустое

public class EmptyFieldException extends Exception {

public EmptyFieldException(int col) {

super("The field in the column " + col + " was empty! Fill in the cell again!");

}

}

//поле заполнено неверно

public class WrongFieldException extends Exception {

public WrongFieldException(int col) {

super("The field in the column " + col + " was filled in wrong!");

}

}

//время введено неверно

public class WrongTimeException extends Exception {

public WrongTimeException() {

super("The filled in time is wrong! Fill in the cell again and enter an existing time!");

}

}

//дата введена неверно

public class WrongDateException extends Exception {

public WrongDateException() {

super("The filled in date is wrong! Fill in the cell again and enter an existing date!");

}

}

//обработка удаления строк

public void checkSelection (int[] arr) throws MyExceptionDelete{

int len = arr.length;

if (len == 0) {

throw new MyExceptionDelete();

}

}

public class MyExceptionDelete extends Exception {

public MyExceptionDelete() {

super ("No row selected for deletion or export.");

}

}

//поля для добавления строки были введены неправильно

public class WrongAddException extends Exception {

public WrongAddException() {

super ("The new line fields were filled in incorrectly!");

}

}

//демонстрация окна

public void show() {

log.info("Demonstration of a window with a list of drivers.");

frame.setVisible(true);

}

public static void main(String[] args) {

log.info("Demonstration of a window with a list of drivers.");

new GuiDriverList().show();

}

}

1. **GuiMenueList**

package front;

import javax.swing.\*;

import org.apache.log4j.Logger;

import java.awt.event.\*;

import back\_gui.SpringUtilities;

public class GuiMenu {

//экранная форма

private JFrame frame;

//кнопки

private JButton Drivers;

private JButton Offences;

private JButton Routes;

//протоколирование

private static final Logger log = Logger.getLogger(GuiRouteList.class);

public void show() {

log.info("Screen form creation.");

frame = new JFrame("Menu");

frame.setSize(228, 138);

frame.setResizable(false);

frame.setLocation(650, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Drivers = new JButton(" List of Drivers ");

Routes = new JButton(" List of Routes ");

Offences = new JButton(" List of Offences ");

JPanel Panel = new JPanel(new SpringLayout());

Panel.add(Drivers);

Panel.add(Routes);

Panel.add(Offences);

SpringUtilities.makeCompactGrid(Panel,3,1,6, 6, 6, 6);

frame.add(Panel);

log.info("Button actions' creation.");

Drivers.addActionListener(new ActionListener()

{

public void actionPerformed (ActionEvent event)

{

new GuiDriverList().show();

frame.setVisible(false);

}

} );

Routes.addActionListener(new ActionListener()

{

public void actionPerformed (ActionEvent event)

{

new GuiRouteList().show();

frame.setVisible(false);

}

} );

Offences.addActionListener(new ActionListener()

{

public void actionPerformed (ActionEvent event)

{

new GuiOffenceList().show();

frame.setVisible(false);

}

} );

frame.setVisible(true);

}

public static void main(String[] args) {

new GuiMenu().show();

}

}

1. **GuiOffenceList**

package front;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.List;

import javax.swing.DefaultRowSorter;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPopupMenu;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.JToolBar;

import javax.swing.RowFilter;

import javax.swing.UIManager;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import javax.swing.table.TableRowSorter;

import org.apache.log4j.Logger;

import com.itextpdf.kernel.PdfException;

import com.itextpdf.kernel.pdf.PdfDocument;

import com.itextpdf.kernel.pdf.PdfWriter;

import com.itextpdf.layout.Document;

import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.layout.element.Table;

import back\_database.DBOperations;

import back\_database.Singleton;

import back\_entities.EntityDriverList;

import back\_entities.EntityOffenceList;

import jakarta.persistence.EntityManager;

public class GuiOffenceList {

//фрейм

private JFrame frame;

//таблица и ее элементы

private DefaultTableModel model;

private String [] columns;

private String [][] data;

private JPopupMenu popupMenu;

private JMenuItem menuItemEdit;

private int [] selectedRows= {};

private TableRowSorter<TableModel> sorter;

//кнопки

private JButton Export;

private JButton Add;

private JButton Delete;

private JButton Search;

//панель инструментов

private JToolBar toolBar;

private JScrollPane scroll;

private JTable table;

private JTextField FilterText;

//выпадающие списки водителей и маршрутов

private String[] alldrivers;

private String[] allroutes;

private JComboBox<String>driver\_choice;

private JComboBox<String>route\_choice;

//заполнение ячейки текстом

String CellText;

private int row;

private int col;

//файловый экспор

private JFrame exp\_frame;

private JFileChooser fileChooser;

private FileNameExtensionFilter filter;

//запись в pdf-файл

private String check\_pdf;

private PdfWriter writer;

private PdfDocument pdfDoc;

private Document doc;

private Table pdfTable;

//панель для строки поиска

private JPanel filterPanel;

//протоколирование

private static final Logger log = Logger.getLogger(GuiOffenceList.class);

public GuiOffenceList() {

log.info("Screen form creation.");

frame = new JFrame("Offences list");

frame.setSize(1200, 600);

frame.setLocation(160, 100);

frame.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

Export = new JButton(new ImageIcon("./img/Export.png"));

Add = new JButton(new ImageIcon("./img/Add.png"));

Delete = new JButton(new ImageIcon("./img/Delete.png"));

Search = new JButton(new ImageIcon("./img/Search.png"));

Export.setToolTipText("Export selected lines to a file");

Add.setToolTipText("Add a line");

Delete.setToolTipText("Delete selected lines");

Search.setToolTipText("Search");

toolBar = new JToolBar("Toolbar");

toolBar.add(Export);

toolBar.add(Add);

toolBar.add(Delete);

frame.setLayout(new BorderLayout());

frame.add(toolBar, BorderLayout.NORTH);

//заполнение таблицы

columns = new String[] {"Full Name","Driver license ID","Date","Time","Route","Delay time, min"};

getDBList();

model = new DefaultTableModel(data, columns);

//запрет на дефолтное редактирование ячеек

table = new JTable(model) {

public boolean isCellEditable(int row, int column) {

return false;

}

};

//всплывающее меню для редактирования ячейки в таблице

popupMenu = new JPopupMenu();

menuItemEdit = new JMenuItem("Edit the cell");

popupMenu.add(menuItemEdit);

//запрет на дефолтное перемещение столбцов таблицы

table.getTableHeader().setReorderingAllowed(false);

//сортировка по столбцам

sorter = new TableRowSorter<TableModel>(model);

table.setRowSorter(sorter);

scroll = new JScrollPane(table);

//скроллер

frame.add(scroll, BorderLayout.CENTER);

//строка поиска по столбцам

FilterText = new JTextField();

FilterText.setColumns(30);

filterPanel = new JPanel();

filterPanel.add(FilterText);

filterPanel.add(Search);

frame.add(filterPanel, BorderLayout.SOUTH);

//заполнение выпадающих списков

alldrivers = DBOperations.getSortedLicences();

allroutes = DBOperations.getSortedRouteNames();

// добавление слушателей

ActionListeners();

}

private void ActionListeners() {

//выделение строк

table.getSelectionModel().addListSelectionListener(new ListSelectionListener(){

public void valueChanged(ListSelectionEvent e) {

selectedRows = table.getSelectedRows();

if (table.getSelectedColumn() != 0 )table.setComponentPopupMenu(popupMenu);

}

});

//сохранение в файл

Export.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

try {

checkSelection(selectedRows);

log.info("Writing to a PDF file.");

log.info("Selecting a PDF file.");

exp\_frame = new JFrame();

fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Choose or create the file to write");

filter = new FileNameExtensionFilter("pdf","pdf");

fileChooser.setFileFilter(filter);

fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

int result = fileChooser.showSaveDialog(exp\_frame);

if (result == JFileChooser.APPROVE\_OPTION ) {

log.info("PDF file was selected.");

try {

check\_pdf = String.valueOf(fileChooser.getSelectedFile());

check\_pdf = remove\_substring(check\_pdf, ".pdf");

writer = new PdfWriter(check\_pdf + ".pdf");

pdfDoc = new PdfDocument (writer);

pdfDoc.addNewPage();

doc = new Document(pdfDoc);

float [] pointColumnWidths = {150F, 150F, 150F, 150F, 150F, 150F};

pdfTable = new Table(pointColumnWidths);

Paragraph paragraph = new Paragraph("Offences List");

pdfTable.addCell("Full name");

pdfTable.addCell("Driver license ID");

pdfTable.addCell("Date");

pdfTable.addCell("Time");

pdfTable.addCell("Route");

pdfTable.addCell("Delay time, min");

for (int rows = 0; rows < selectedRows.length; rows++) {

for (int cols = 0; cols < table.getColumnCount(); cols++) {

pdfTable.addCell(table.getModel().getValueAt(rows, cols).toString());

}

}

doc.add(paragraph);

doc.add(pdfTable);

doc.close();

JOptionPane.showMessageDialog(table,"The table data is saved to a file " + check\_pdf +".pdf");

log.info("The table data is saved to a file " + check\_pdf + ".pdf");

} catch (PdfException ex) {

ex.printStackTrace();

} catch (FileNotFoundException e1) {

JOptionPane.showMessageDialog(table,"File was not found.");

e1.printStackTrace();

}

}

else {

log.warn("To write data to a new file, you must enter its name and click the save button , otherwise writing to the file will not happen.");

log.info("Writing to the file has been canceled.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

myEx.printStackTrace();

}

}

});

//закрытие окна

frame.addWindowListener(new WindowAdapter()

{

public void windowClosing(WindowEvent event) {

Object[] options = { "Yes", "No" };

int option = JOptionPane.showOptionDialog(event.getWindow(), "Do you want to close this window?",

"Confirmation", JOptionPane.YES\_NO\_OPTION,

JOptionPane.INFORMATION\_MESSAGE, null, options,

options[0]);

if (option == 0) {

event.getWindow().setVisible(false);

new GuiMenu().show();

}

}

});

//поиск

Search.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

log.info("Table search.");

String text;

text = FilterText.getText();

try {

CheckTextField(text, 10);

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(text));

log.info("The elements were found successfully.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

log.info("Error - an empty input field in the search bar.", myEx);

JOptionPane.showMessageDialog(table, myEx.getMessage());

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(""));

}

finally {

FilterText.setText("");

}

}});

//редактирование ячеек

menuItemEdit.addActionListener(new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Editing a cell in a table row. ");

row = table.getSelectedRow();

col = table.getSelectedColumn();

String prevdriver = model.getValueAt(row, 1).toString();

String prevdate = model.getValueAt(row, 2).toString();

String prevtime = model.getValueAt(row, 3).toString();

String prevdelay = model.getValueAt(row, 5).toString();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

switch (col) {

case(1):

driver\_choice = new JComboBox<String>(alldrivers);

driver\_choice.setSelectedItem(table.getValueAt(row, col));

int result = JOptionPane.showOptionDialog(frame, driver\_choice, "Choose the driver license id: ",

JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.INFORMATION\_MESSAGE, null, null, null);

if (result == JOptionPane.OK\_OPTION) {

try {

checkRepeat(driver\_choice.getSelectedItem().toString(), prevdate, prevtime);

EntityManager em = Singleton.createEMandTrans();

EntityOffenceList Offence = DBOperations.searchOffence(prevdate, prevtime, prevdriver);

Offence.setDriver(driver\_choice.getSelectedItem().toString());

EntityDriverList Driver = DBOperations.searchDriverWithLicense(driver\_choice.getSelectedItem().toString());

Offence.setName(Driver.getName());

em.merge(Offence);

em.merge(Driver);

Singleton.finishEMandTrans(em);

table.setValueAt(driver\_choice.getSelectedItem().toString(), row, col);

table.setValueAt(Driver.getName(), row, 0);

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The driver license id, date and time fields reapeat.", myEx);

log.warn("The new cell will not be added.", myEx);

}

}

break;

case(2):

CellText = JOptionPane.showInputDialog(table, "Change to (XX.XX.XXXX) from 2000: ","Edit the date", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

try {

checkRepeat(prevdriver, CellText, prevtime);

EntityManager em = Singleton.createEMandTrans();

EntityOffenceList Offence = DBOperations.searchOffence(prevdate, prevtime, prevdriver);

Offence.setDate(CellText);

em.merge(Offence);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The driver license id, date and time fields reapeat.", myEx);

log.warn("The new cell will not be added.", myEx);

}

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevdate, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(3):

CellText = JOptionPane.showInputDialog(table, "Change to (XX:XX): ","Edit the time", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

try {

checkRepeat(prevdriver, prevdate, CellText);

EntityManager em = Singleton.createEMandTrans();

EntityOffenceList Offence = DBOperations.searchOffence(prevdate, prevtime, prevdriver);

Offence.setTime(CellText);

em.merge(Offence);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The driver license id, date and time fields reapeat.", myEx);

log.warn("The new cell will not be added.", myEx);

}

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevtime, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(4):

route\_choice = new JComboBox<String>(allroutes);

route\_choice.setSelectedItem(table.getValueAt(row, col));

result = JOptionPane.showOptionDialog(frame, route\_choice, "Choose the route: ",

JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.INFORMATION\_MESSAGE, null, null, null);

if (result == JOptionPane.OK\_OPTION) {

EntityManager em = Singleton.createEMandTrans();

EntityOffenceList Offence = DBOperations.searchOffence(prevdate, prevtime, prevdriver);

Offence.setRoute(route\_choice.getSelectedItem().toString());

em.merge(Offence);

Singleton.finishEMandTrans(em);

table.setValueAt(route\_choice.getSelectedItem().toString(), row, col);

}

break;

case(5):

CellText = JOptionPane.showInputDialog(table, "Change to (number of minutes, up to 300 ): ","Edit the delay time", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityOffenceList Offence = DBOperations.searchOffence(prevdate, prevtime, prevdriver);

Offence.setDelay(CellText);

em.merge(Offence);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongFieldException | EmptyFieldException | WrongTimeException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevdelay, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

}

}});

//добавление новой строки

Add.addActionListener (new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Adding a new row to the table.");

row = table.getSelectedRow() + 1;

driver\_choice = new JComboBox<String>(alldrivers);

driver\_choice.setSelectedItem(table.getValueAt(row, 1));

JTextField add\_date = new JTextField();

JTextField add\_time = new JTextField();

route\_choice = new JComboBox<String>(allroutes);

route\_choice.setSelectedItem(table.getValueAt(row, 4));

JTextField add\_delay = new JTextField();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

Object[] message = {

"Driver license ID: ", driver\_choice,

"Date (XX.XX.XXXX) from 2000: ", add\_date,

"Time (XX:XX): ", add\_time,

"Route: ", route\_choice,

"Delay time, min: ", add\_delay,

};

int option = JOptionPane.showConfirmDialog(table, message, "New row", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

if(option == JOptionPane.OK\_OPTION) {

int exeption\_sum = 0;

try{

log.info(row +"," + 2 + "-cell is being filled in.");

CheckTextField(add\_date.getText(), 2);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 2 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 3 + "-cell is being filled in.");

CheckTextField(add\_time.getText(), 3);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 3 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 5 + "-cell is being filled in.");

CheckTextField(add\_delay.getText(), 5);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 5 + "-cell has been filled in wrong.");

}

try{

if(exeption\_sum > 0) throw new WrongAddException();

try {

checkRepeat(driver\_choice.getSelectedItem().toString(), add\_date.getText(), add\_time.getText());

EntityManager em = Singleton.createEMandTrans();

EntityDriverList Driver = DBOperations.searchDriverWithLicense(driver\_choice.getSelectedItem().toString());

em.merge(Driver);

Singleton.finishEMandTrans(em);

model.insertRow(row, new String [] {Driver.getName(), driver\_choice.getSelectedItem().toString(), add\_date.getText(),

add\_time.getText(), route\_choice.getSelectedItem().toString(), add\_delay.getText()});

em = Singleton.createEMandTrans();

EntityDriverList OneDriver = DBOperations.searchDriverWithLicense(driver\_choice.getSelectedItem().toString());

EntityOffenceList newOffence = new EntityOffenceList(Driver.getName(), driver\_choice.getSelectedItem().toString(), add\_date.getText(),

add\_time.getText(), route\_choice.getSelectedItem().toString(), add\_delay.getText(), OneDriver);

em.persist(newOffence);

Singleton.finishEMandTrans(em);

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The driver license id, date and time fields reapeat.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

catch(WrongAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The new line fields were filled in incorrectly.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

}

});

//удаление строк

Delete.addActionListener(new ActionListener(){

public void actionPerformed (ActionEvent event){

log.info("Deletion of selected rows from the table.");

try {

checkSelection(selectedRows);

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Confirm");

int option = JOptionPane.showConfirmDialog(table, "After confirmation, it will be impossible to cancel the deletion of the selected rows.", "Delete?", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

log.warn("After confirmation, it will be impossible to cancel the deletion of the selected rows.");

if (option == JOptionPane.OK\_OPTION) {

UIManager.put("OptionPane.okButtonText", "OK");

for(int i = selectedRows.length-1; i > -1; i--){

//System.out.println(i);

EntityManager em = Singleton.createEMandTrans();

EntityOffenceList Offence = DBOperations.searchOffence(String.valueOf(table.getModel().getValueAt(selectedRows[i], 1)), String.valueOf(table.getModel().getValueAt(selectedRows[i], 2)), String.valueOf(table.getModel().getValueAt(selectedRows[i], 0)));

em.merge(Offence);

Singleton.finishEMandTrans(em);

model.removeRow(selectedRows[i]);

DBOperations.deleteOffence(Offence.getId());

}

JOptionPane.showMessageDialog (frame, "The cell data of the selected rows has been deleted from the table.","Deletion message", JOptionPane.PLAIN\_MESSAGE);

log.info("The selected rows are removed from the table.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("Исключительная ситуация - нет выделенных строк для удаления. ", myEx);

log.warn("Необходимо выделить строки для удаления, иначе удаление будет недоступно. ");

}

}

});

}

//получение данных из БД для таблицы

private void getDBList() {

// получение списка из БД

List<EntityOffenceList> Offences = DBOperations.getSortedOffences();

ArrayList<String[]> values = new ArrayList<String[]>(Offences.size() + 1);

for (int i = 0; i < Offences.size(); i++) {

EntityOffenceList entity = Offences.get(i);

values.add(new String[]{entity.getName(), entity.getDriver(),entity.getDate(),entity.getTime(),entity.getRoute(), entity.getDelay()});

}

// преобразование в формат для заполнения таблицы данными.

data = new String[values.size()][6];

for (int i = 0; i < values.size(); i++) data[i] = values.get(i);

}

//подсчет подстрок в строке

public static int count\_substring(String str, String target) {

return (str.length() - str.replace(target, "").length()) / target.length();

}

//удаление подстрок в строке

public static String remove\_substring(String str, String substr) {

int count = count\_substring(str, substr);

if (count > 0) {

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

str = str.replace(substr, "");

}

return str;

}

//проверка введенного кол-ва минут

public int CheckMinNumber (String text){

int flag = 0;

if (Integer.valueOf(text) > 0 && Integer.valueOf(text) <= 300 ) {

flag = 1;

}

return flag;

}

//проверка времени

public int CheckTime (String text){

String[] premSplitString = text.split("\\:");

int[] premSplit = new int[] {Integer.valueOf(premSplitString[0]), Integer.valueOf(premSplitString[1])};

int flag = 0;

if (premSplit[0] >= 0 && premSplit[0] < 24 && premSplit[1] >= 0 && premSplit[1] < 60 ) {

flag = 1;

}

return flag;

}

//проверка даты

public int CheckDate (String text){

String[] premSplitString = text.split("\\.");

int[] premSplit = new int[] {Integer.valueOf(premSplitString[0]), Integer.valueOf(premSplitString[1]), Integer.valueOf(premSplitString[2])};

int flag = 0;

if (premSplit[1] > 0 && premSplit[1] <= 12 && premSplit[0] > 0 && premSplit[0] <= 31 && premSplit[2] >= 2000) {

if (premSplit[1] == 2 && (premSplit[0] < 29 || ((((premSplit[2] % 4) == 0 &&

(premSplit[2] % 100) != 0) || (premSplit[2] % 400) == 0) && premSplit[0] < 30)))

flag = 1;

else if ((premSplit[1] == 4 || premSplit[1] == 6 || premSplit[1] == 9 || premSplit[1] == 11)

&& premSplit[0] < 31)

flag = 1;

else if (premSplit[1] == 1 || premSplit[1] == 3 || premSplit[1] == 5 || premSplit[1] == 7 || premSplit[1] == 8 || premSplit[1] == 10 || premSplit[1] == 12)

flag = 1;

}

return flag;

}

//проверка повтора строки

private void checkRepeat(String driver, String date, String time) throws RepeatAddException{

for (int i = 0; i < table.getRowCount(); i++) {

if (table.getModel().getValueAt(i, 0).equals(driver) && table.getModel().getValueAt(i, 1).equals(date) && table.getModel().getValueAt(i, 2).equals(time)) {

throw new RepeatAddException();

}

}

}

public class RepeatAddException extends Exception {

public RepeatAddException() {

super("It is restricted to add a row with the same driver license id, date and time! Please, try again and change one of these fields.");

}

}

//обработка заполнения текстового поля

public void CheckTextField (String text, int col) throws EmptyFieldException, WrongFieldException, WrongDateException, WrongTimeException, EmptySearchFieldException {

if (col != 10 && (text.trim().isEmpty() || text == null)) throw new EmptyFieldException(col);

if (col == 10 && (text.trim().isEmpty() || text == null)) throw new EmptySearchFieldException();

if (col == 5 && (!text.matches("^[1-9]\\d\*$") || CheckMinNumber(text) == 0)) throw new WrongFieldException(col);

if (col == 3 && !text.matches("^\\d{2}:\\d{2}$")) throw new WrongFieldException(col);

if (col == 3 && text.matches("^\\d{2}:\\d{2}$") && CheckTime(text) == 0) throw new WrongTimeException();

if (col == 2 && !text.matches("^\\d{2}.\\d{2}.\\d{4}$")) throw new WrongFieldException(col);

if (col == 2 && text.matches("^\\d{2}.\\d{2}.\\d{4}$") && CheckDate(text) == 0) throw new WrongDateException();

}

//поле поиска пустое

public class EmptySearchFieldException extends Exception {

public EmptySearchFieldException() {

super("The search field was empty! Enter the search text again!");

}

}

//поле пустое

public class EmptyFieldException extends Exception {

public EmptyFieldException(int col) {

super("The field in the column " + col + " was empty! Fill in the cell again!");

}

}

//поле заполнено неверно

public class WrongFieldException extends Exception {

public WrongFieldException(int col) {

super("The field in the column " + col + " was filled in wrong!");

}

}

//время введено неверно

public class WrongTimeException extends Exception {

public WrongTimeException() {

super("The filled in time is wrong! Fill in the cell again and enter an existing time!");

}

}

//дата введена неверно

public class WrongDateException extends Exception {

public WrongDateException() {

super("The filled in date is wrong! Fill in the cell again and enter an existing date!");

}

}

//обработка удаления строк

public void checkSelection (int[] arr) throws MyExceptionDelete{

int len = arr.length;

if (len == 0) {

throw new MyExceptionDelete();

}

}

public class MyExceptionDelete extends Exception {

public MyExceptionDelete() {

super ("No row selected for deletion or export.");

}

}

//поля для добовления строки были введены неправильно

public class WrongAddException extends Exception {

public WrongAddException() {

super ("The new line fields were filled in incorrectly!");

}

}

//демонстрация окна

public void show() {

log.info("Demonstration of a window with a list of offences.");

frame.setVisible(true);

}

public static void main(String[] args) {

log.info("Demonstration of a window with a list of offences.");

new GuiOffenceList().show();

}

}

1. **GuiRouteList**

package front;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.List;

import javax.swing.DefaultRowSorter;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JCheckBox;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPopupMenu;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.JToolBar;

import javax.swing.RowFilter;

import javax.swing.UIManager;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import javax.swing.table.TableRowSorter;

import org.apache.log4j.Logger;

import com.itextpdf.kernel.PdfException;

import com.itextpdf.kernel.pdf.PdfDocument;

import com.itextpdf.kernel.pdf.PdfWriter;

import com.itextpdf.layout.Document;

import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.layout.element.Table;

import back\_database.DBOperations;

import back\_database.Singleton;

import back\_entities.EntityRouteList;

import back\_gui.ButtonEditor;

import back\_gui.ButtonRenderer;

import jakarta.persistence.EntityManager;

public class GuiRouteList {

//фрейм

private JFrame frame;

//таблица и ее элементы

private DefaultTableModel model;

private String [] columns;

private String [][] data;

private JPopupMenu popupMenu;

private JMenuItem menuItemEdit;

private int [] selectedRows= {};

private TableRowSorter<TableModel> sorter;

//кнопки

private JButton Export;

private JButton Add;

private JButton Delete;

private JButton Search;

//панель инструментов

private JToolBar toolBar;

private JScrollPane scroll;

private JTable table;

private JTextField FilterText;

//заполнение ячейки текстом

String CellText;

private int row;

private int col;

//файловый экспорт

private JFrame exp\_frame;

private JFileChooser fileChooser;

private FileNameExtensionFilter filter;

//запись в pdf-файл

private String check\_pdf;

private PdfWriter writer;

private PdfDocument pdfDoc;

private Document doc;

private Table pdfTable;

//панель для строки поиска

private JPanel filterPanel;

//протоколирование

private static final Logger log = Logger.getLogger(GuiRouteList.class);

public GuiRouteList() {

log.info("Screen form creation.");

frame = new JFrame("Routes list");

frame.setSize(1200, 600);

frame.setLocation(160, 100);

frame.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

Export = new JButton(new ImageIcon("./img/Export.png"));

Add = new JButton(new ImageIcon("./img/Add.png"));

Delete = new JButton(new ImageIcon("./img/Delete.png"));

Search = new JButton(new ImageIcon("./img/Search.png"));

Export.setToolTipText("Export selected lines to a file");

Add.setToolTipText("Add a line");

Delete.setToolTipText("Delete selected lines");

Search.setToolTipText("Search");

toolBar = new JToolBar("Toolbar");

toolBar.add(Export);

toolBar.add(Add);

toolBar.add(Delete);

frame.setLayout(new BorderLayout());

frame.add(toolBar, BorderLayout.NORTH);

//заполнение таблицы

columns = new String[] {"Route + direction","Start time","End time"};

getDBList();

model = new DefaultTableModel(data, columns);

//запрет на дефолтное редактирование ячеек

table = new JTable(model) {

public boolean isCellEditable(int row, int column) {

if(col == 0) return true;

else return false;

}

};

table.getColumn("Route + direction").setCellRenderer(new ButtonRenderer());

table.getColumn("Route + direction").setCellEditor(new ButtonEditor(this,new JCheckBox()));

//всплывающее меню для редактирования ячейки в таблице

popupMenu = new JPopupMenu();

menuItemEdit = new JMenuItem("Edit the cell");

popupMenu.add(menuItemEdit);

//запрет на дефолтное перемещение столбцов таблицы

table.getTableHeader().setReorderingAllowed(false);

//сортировка по столбцам

sorter = new TableRowSorter<TableModel>(model);

table.setRowSorter(sorter);

scroll = new JScrollPane(table);

//скроллер

frame.add(scroll, BorderLayout.CENTER);

//строка поиска по столбцам

FilterText = new JTextField();

FilterText.setColumns(30);

filterPanel = new JPanel();

filterPanel.add(FilterText);

filterPanel.add(Search);

frame.add(filterPanel, BorderLayout.SOUTH);

// добавление слушателей

ActionListeners();

}

private void ActionListeners() {

//выделение строк

table.getSelectionModel().addListSelectionListener(new ListSelectionListener(){

public void valueChanged(ListSelectionEvent e) {

selectedRows = table.getSelectedRows();

table.setComponentPopupMenu(popupMenu);

}

});

//сохранение в файл

Export.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

try {

checkSelection(selectedRows);

log.info("Writing to a PDF file.");

log.info("Selecting a PDF file.");

exp\_frame = new JFrame();

fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Choose or create the file to write");

filter = new FileNameExtensionFilter("pdf","pdf");

fileChooser.setFileFilter(filter);

fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

int result = fileChooser.showSaveDialog(exp\_frame);

if (result == JFileChooser.APPROVE\_OPTION ) {

log.info("PDF file was selected.");

try {

check\_pdf = String.valueOf(fileChooser.getSelectedFile());

check\_pdf = remove\_substring(check\_pdf, ".pdf");

writer = new PdfWriter(check\_pdf + ".pdf");

pdfDoc = new PdfDocument (writer);

pdfDoc.addNewPage();

doc = new Document(pdfDoc);

float [] pointColumnWidths = {150F, 150F, 150F};

pdfTable = new Table(pointColumnWidths);

Paragraph paragraph = new Paragraph("Routes List");

pdfTable.addCell("Route + direction");

pdfTable.addCell("Start time");

pdfTable.addCell("End time");

for (int rows = 0; rows < selectedRows.length; rows++) {

for (int cols = 0; cols < table.getColumnCount(); cols++) {

pdfTable.addCell(table.getModel().getValueAt(rows, cols).toString());

}

}

doc.add(paragraph);

doc.add(pdfTable);

doc.close();

JOptionPane.showMessageDialog(table,"The table data is saved to a file " + check\_pdf +".pdf");

log.info("The table data is saved to a file " + check\_pdf + ".pdf");

} catch (PdfException ex) {

ex.printStackTrace();

} catch (FileNotFoundException e1) {

JOptionPane.showMessageDialog(table,"File was not found.");

e1.printStackTrace();

}

}

else {

log.warn("To write data to a new file, you must enter its name and click the save button , otherwise writing to the file will not happen.");

log.info("Writing to the file has been canceled.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

myEx.printStackTrace();

}

}

});

//закрытие окна

frame.addWindowListener(new WindowAdapter()

{

public void windowClosing(WindowEvent event) {

Object[] options = { "Yes", "No" };

int option = JOptionPane.showOptionDialog(event.getWindow(), "Do you want to close this window?",

"Confirmation", JOptionPane.YES\_NO\_OPTION,

JOptionPane.INFORMATION\_MESSAGE, null, options,

options[0]);

if (option == 0) {

event.getWindow().setVisible(false);

new GuiMenu().show();

}

}

});

//поиск

Search.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

log.info("Table search.");

String text;

text = FilterText.getText();

try {

CheckTextField(text, 10);

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(text));

log.info("The elements were found successfully.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

log.info("Error - an empty input field in the search bar.", myEx);

JOptionPane.showMessageDialog(table, myEx.getMessage());

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(""));

}

finally {

FilterText.setText("");

}

}});

//редактирование ячеек

menuItemEdit.addActionListener(new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Editing a cell in a table row. ");

row = table.getSelectedRow();

col = table.getSelectedColumn();

String prevroute = model.getValueAt(row, 0).toString();

String prevstart = model.getValueAt(row, 1).toString();

String prevend = model.getValueAt(row, 2).toString();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

switch (col) {

case(0):

CellText = JOptionPane.showInputDialog(table, "Change to (route name + straight/forward/round): ","Edit the route + direction", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

try {

checkRepeat(CellText);

EntityManager em = Singleton.createEMandTrans();

EntityRouteList Route = DBOperations.searchRouteWithName(prevroute);

Route.setName(CellText);

em.merge(Route);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The route field reapeats.", myEx);

log.warn("The new cell will not be added.", myEx);

}

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevroute, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(1):

CellText = JOptionPane.showInputDialog(table, "Change to (XX:XX): ","Edit the start time", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityRouteList Route = DBOperations.searchRouteWithName(prevroute);

Route.setStart(CellText);

em.merge(Route);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevstart, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(2):

CellText = JOptionPane.showInputDialog(table, "Change to (XX:XX): ","Edit the end time", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityRouteList Route = DBOperations.searchRouteWithName(prevroute);

Route.setEnd(CellText);

em.merge(Route);

Singleton.finishEMandTrans(em);

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevend, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

}

}});

//добавление новой строки

Add.addActionListener (new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Adding a new row to the table after the selected one.");

row = table.getSelectedRow() + 1;

JTextField add\_route = new JTextField();

JTextField add\_start = new JTextField();

JTextField add\_end = new JTextField();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

Object[] message = {

"Route + direction (route name + straight/forward/round): ", add\_route,

"Start time (XX:XX): ", add\_start,

"End time (XX:XX): ", add\_end

};

int option = JOptionPane.showConfirmDialog(table, message, "New row", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

if(option == JOptionPane.OK\_OPTION) {

int exeption\_sum = 0;

try{

log.info(row +"," + 0 + "-cell is being filled in.");

CheckTextField(add\_route.getText(), 0);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 0 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 1 + "-cell is being filled in.");

CheckTextField(add\_start.getText(), 1);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 1 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 2 + "-cell is being filled in.");

CheckTextField(add\_end.getText(), 2);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 2 + "-cell has been filled in wrong.");

}

try{

if(exeption\_sum > 0) throw new WrongAddException();

try {

checkRepeat(add\_route.getText());

model.insertRow(row, new String [] {add\_route.getText(), add\_start.getText(), add\_end.getText()});

EntityManager em = Singleton.createEMandTrans();

EntityRouteList newRoute = new EntityRouteList(add\_route.getText(), add\_start.getText(), add\_end.getText());

em.persist(newRoute);

Singleton.finishEMandTrans(em);

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The route field reapeats.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

catch(WrongAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The new line fields were filled in incorrectly.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

}

});

//удаление строк

Delete.addActionListener(new ActionListener(){

public void actionPerformed (ActionEvent event){

log.info("Deletion of selected rows from the table.");

try {

checkSelection(selectedRows);

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Confirm");

int option = JOptionPane.showConfirmDialog(table, "After confirmation, it will be impossible to cancel the deletion of the selected rows.", "Delete?", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

log.warn("After confirmation, it will be impossible to cancel the deletion of the selected rows.");

if (option == JOptionPane.OK\_OPTION) {

UIManager.put("OptionPane.okButtonText", "OK");

for(int i = selectedRows.length-1; i > -1; i--){

EntityManager em = Singleton.createEMandTrans();

EntityRouteList Route = DBOperations.searchRouteWithName(String.valueOf(table.getModel().getValueAt(selectedRows[i], 0)));

em.merge(Route);

Singleton.finishEMandTrans(em);

model.removeRow(selectedRows[i]);

DBOperations.deleteRoute(Route.getId());

}

JOptionPane.showMessageDialog (frame, "The cell data of the selected rows has been deleted from the table.","Deletion message", JOptionPane.PLAIN\_MESSAGE);

log.info("The selected rows are removed from the table.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("Исключительная ситуация - нет выделенных строк для удаления. ", myEx);

log.warn("Необходимо выделить строки для удаления, иначе удаление будет недоступно. ");

}

}

});

}

//получение данных из БД для таблицы

private void getDBList() {

// получение списка из БД

List<EntityRouteList> Routes = DBOperations.getSortedRoutes();

ArrayList<String[]> values = new ArrayList<String[]>(Routes.size() + 1);

for (int i = 0; i < Routes.size(); i++) {

EntityRouteList entity = Routes.get(i);

values.add(new String[]{entity.getName(),entity.getStart(),entity.getEnd()});

}

// преобразование в формат для заполнения таблицы данными.

data = new String[values.size()][3];

for (int i = 0; i < values.size(); i++) data[i] = values.get(i);

}

//подсчет подстрок в строке

public static int count\_substring(String str, String target) {

return (str.length() - str.replace(target, "").length()) / target.length();

}

//удаление подстрок в строке

public static String remove\_substring(String str, String substr) {

int count = count\_substring(str, substr);

if (count > 0) {

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

str = str.replace(substr, "");

}

return str;

}

//проверка времени

public int CheckTime (String text){

String[] premSplitString = text.split("\\:");

int[] premSplit = new int[] {Integer.valueOf(premSplitString[0]), Integer.valueOf(premSplitString[1])};

int flag = 0;

if (premSplit[0] >= 0 && premSplit[0] < 24 && premSplit[1] >= 0 && premSplit[1] < 60 ) {

flag = 1;

}

return flag;

}

//проверка повтора строки

private void checkRepeat(String str) throws RepeatAddException{

for (int i = 0; i < table.getRowCount(); i++) {

if (table.getModel().getValueAt(i, 0).equals(str)) {

throw new RepeatAddException();

}

}

}

public class RepeatAddException extends Exception {

public RepeatAddException() {

super("It is restricted to add a row with the same driver license id, date and time! Please, try again and change one of these fields.");

}

}

//обработка заполнения текстового поля

public void CheckTextField (String text, int col) throws EmptyFieldException, WrongFieldException, WrongDateException, WrongTimeException, EmptySearchFieldException {

String text0 = text;

if (count\_substring(text,"round") == 1) text0.replace(" round", "");

else if (count\_substring(text,"straight") == 1) text0.replace(" straight", "");

else if (count\_substring(text,"forward") == 1) text0.replace(" forward", "");

System.out.print(text);

if ( (col == 0 && !text0.matches("^[A-Z a-z0-9]+$")) || (col == 0 && count\_substring(text,"round")== 0 && count\_substring(text,"straight")== 0 && count\_substring(text,"forward") == 0) ) throw new WrongFieldException(col);

if (col != 10 && (text.trim().isEmpty() || text == null) ) throw new EmptyFieldException(col);

if (col == 10 && (text.trim().isEmpty() || text == null)) throw new EmptySearchFieldException();

if ((col == 1 || col == 2) && !text.matches("^\\d{2}:\\d{2}$") && CheckTime(text) == 0) throw new WrongFieldException(col);

}

//поле поиска пустое

public class EmptySearchFieldException extends Exception {

public EmptySearchFieldException() {

super("The search field was empty! Enter the search text again!");

}

}

//поле пустое

public class EmptyFieldException extends Exception {

public EmptyFieldException(int col) {

super("The field in the column " + col + " was empty! Fill in the cell again!");

}

}

//поле заполнено неверно

public class WrongFieldException extends Exception {

public WrongFieldException(int col) {

super("The field in the column " + col + " was filled in wrong!");

}

}

//время введено неверно

public class WrongTimeException extends Exception {

public WrongTimeException() {

super("The filled in time is wrong! Fill in the cell again and enter an existing time!");

}

}

//дата введена неверно

public class WrongDateException extends Exception {

public WrongDateException() {

super("The filled in date is wrong! Fill in the cell again and enter an existing date!");

}

}

//обработка удаления строк

public void checkSelection (int[] arr) throws MyExceptionDelete{

int len = arr.length;

if (len == 0) {

throw new MyExceptionDelete();

}

}

public class MyExceptionDelete extends Exception {

public MyExceptionDelete() {

super ("No row selected for deletion or export.");

}

}

//поля для добавления строки были введены неправильно

public class WrongAddException extends Exception {

public WrongAddException() {

super ("The new line fields were filled in incorrectly!");

}

}

//демонстрация окна

public void show() {

log.info("Demonstration of a window with a list of routes.");

frame.setVisible(true);

}

//закрытие окна для перехода к следующему со списком остановок

public void close() {

frame.setVisible(false);

}

public static void main(String[] args) {

log.info("Demonstration of a window with a list of routes.");

new GuiRouteList().show();

}

}

1. **GuiStationList**

package front;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.List;

import javax.swing.DefaultRowSorter;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPopupMenu;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.JToolBar;

import javax.swing.RowFilter;

import javax.swing.UIManager;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import javax.swing.table.TableRowSorter;

import org.apache.log4j.Logger;

import com.itextpdf.kernel.PdfException;

import com.itextpdf.kernel.geom.PageSize;

import com.itextpdf.kernel.pdf.PdfDocument;

import com.itextpdf.kernel.pdf.PdfWriter;

import com.itextpdf.layout.Document;

import com.itextpdf.layout.element.Paragraph;

import com.itextpdf.layout.element.Table;

import back\_database.DBOperations;

import back\_database.Singleton;

import back\_entities.EntityRouteList;

import back\_entities.EntityStationList;

import back\_gui.ButtonEditor;

import jakarta.persistence.EntityManager;

public class GuiStationList {

//фрейм

private JFrame frame;

//таблица и ее элементы

private DefaultTableModel model;

private String [] columns;

private String [][] data;

private JPopupMenu popupMenu;

private JMenuItem menuItemEdit;

private int [] selectedRows= {};

private TableRowSorter<TableModel> sorter;

//кнопки

private JButton Export;

private JButton Add;

private JButton Delete;

private JButton Search;

//панель инструментов

private JToolBar toolBar;

private JScrollPane scroll;

private JTable table;

private JTextField FilterText;

//заполнение ячейки текстом

String CellText;

private int row;

private int col;

//файловый экспорт

private JFrame exp\_frame;

private JFileChooser fileChooser;

private FileNameExtensionFilter filter;

//запись в pdf-файл

private String check\_pdf;

private PdfWriter writer;

private PdfDocument pdfDoc;

private Document doc;

private Table pdfTable;

//панель для строки поиска

private JPanel filterPanel;

//протоколирование

private static final Logger log = Logger.getLogger(GuiRouteList.class);

private String routename;

public GuiStationList(String route)

{

routename = route;

log.info("Screen form creation.");

frame = new JFrame(routename + "-route stations list");

frame.setSize(1200, 600);

frame.setLocation(160, 100);

frame.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

Export = new JButton(new ImageIcon("./img/Export.png"));

Add = new JButton(new ImageIcon("./img/Add.png"));

Delete = new JButton(new ImageIcon("./img/Delete.png"));

Search = new JButton(new ImageIcon("./img/Search.png"));

Export.setToolTipText("Экспортировать выделенные строки в файл");

Add.setToolTipText("Добавить строку после выделенной");

Delete.setToolTipText("Удалить выделенные строки");

Search.setToolTipText("Поиск");

toolBar = new JToolBar("Панель инструментов");

toolBar.add(Export);

toolBar.add(Add);

toolBar.add(Delete);

frame.setLayout(new BorderLayout());

frame.add(toolBar, BorderLayout.NORTH);

//заполнение таблицы

columns = new String[] {"Station","Start time","End time", "Time interval-to 10", "Time interval-to 16",

"Time interval-to 20", "Time interval-past 20"};

getDBList();

model = new DefaultTableModel(data, columns);

//запрет на дефолтное редактирование ячеек

table = new JTable(model) {

public boolean isCellEditable(int row, int column) {

return false;

}

};

//всплывающее меню для редактирования ячейки в таблице

popupMenu = new JPopupMenu();

menuItemEdit = new JMenuItem("Edit the cell");

popupMenu.add(menuItemEdit);

//запрет на дефолтное перемещение столбцов таблицы

table.getTableHeader().setReorderingAllowed(false);

//сортировка по столбцам

sorter = new TableRowSorter<TableModel>(model);

table.setRowSorter(sorter);

scroll = new JScrollPane(table);

//скроллер

frame.add(scroll, BorderLayout.CENTER);

//строка поиска по столбцам

FilterText = new JTextField();

FilterText.setColumns(30);

filterPanel = new JPanel();

filterPanel.add(FilterText);

filterPanel.add(Search);

frame.add(filterPanel, BorderLayout.SOUTH);

// добавление слушателей

ActionListeners();

}

private void ActionListeners() {

//выделение строк

table.getSelectionModel().addListSelectionListener(new ListSelectionListener(){

public void valueChanged(ListSelectionEvent e) {

selectedRows = table.getSelectedRows();

table.setComponentPopupMenu(popupMenu);

}

});

//сохранение в файл

Export.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

try {

checkSelection(selectedRows);

log.info("Writing to a PDF file.");

log.info("Selecting a PDF file.");

exp\_frame = new JFrame();

fileChooser = new JFileChooser();

fileChooser.setDialogTitle("Choose or create the file to write");

filter = new FileNameExtensionFilter("pdf","pdf");

fileChooser.setFileFilter(filter);

fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);

int result = fileChooser.showSaveDialog(exp\_frame);

if (result == JFileChooser.APPROVE\_OPTION ) {

log.info("PDF file was selected.");

try {

check\_pdf = String.valueOf(fileChooser.getSelectedFile());

check\_pdf = remove\_substring(check\_pdf, ".pdf");

writer = new PdfWriter(check\_pdf + ".pdf");

pdfDoc = new PdfDocument (writer);

pdfDoc.setDefaultPageSize(PageSize.A3.rotate());

pdfDoc.addNewPage();

doc = new Document(pdfDoc);

float [] pointColumnWidths = {150F, 150F, 150F,150F, 150F, 150F,150F};

pdfTable = new Table(pointColumnWidths);

Paragraph paragraph = new Paragraph(routename + "-route stations list");

pdfTable.addCell("Station");

pdfTable.addCell("Start time");

pdfTable.addCell("End time");

pdfTable.addCell("to 10, min");

pdfTable.addCell("to 16, min");

pdfTable.addCell("to 20, min");

pdfTable.addCell("past 20,min");

for (int rows = 0; rows < selectedRows.length; rows++) {

for (int cols = 0; cols < table.getColumnCount(); cols++) {

pdfTable.addCell(table.getModel().getValueAt(rows, cols).toString());

}

}

doc.add(paragraph);

doc.add(pdfTable);

doc.close();

JOptionPane.showMessageDialog(table,"The table data is saved to a file " + check\_pdf +".pdf");

log.info("The table data is saved to a file " + check\_pdf + ".pdf");

} catch (PdfException ex) {

ex.printStackTrace();

} catch (FileNotFoundException e1) {

JOptionPane.showMessageDialog(table,"File was not found.");

e1.printStackTrace();

}

}

else {

log.warn("To write data to a new file, you must enter its name and click the save button , otherwise writing to the file will not happen.");

log.info("Writing to the file has been canceled.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

myEx.printStackTrace();

}

}

});

//закрытие окна

frame.addWindowListener(new WindowAdapter()

{

public void windowClosing(WindowEvent event) {

Object[] options = { "Yes", "No" };

int option = JOptionPane.showOptionDialog(event.getWindow(), "Do you want to close this window?",

"Confirmation", JOptionPane.YES\_NO\_OPTION,

JOptionPane.INFORMATION\_MESSAGE, null, options,

options[0]);

if (option == 0) {

event.getWindow().setVisible(false);

new GuiRouteList().show();

}

}

});

//поиск

Search.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

log.info("Table search.");

String text;

text = FilterText.getText();

try {

CheckTextField(text, 10);

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(text));

log.info("The elements were found successfully.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

log.info("Error - an empty input field in the search bar.", myEx);

JOptionPane.showMessageDialog(table, myEx.getMessage());

((DefaultRowSorter<TableModel, Integer>) sorter).setRowFilter(RowFilter.regexFilter(""));

}

finally {

FilterText.setText("");

}

}});

//редактирование ячеек

menuItemEdit.addActionListener(new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Editing a cell in a table row. ");

row = table.getSelectedRow();

col = table.getSelectedColumn();

String prevname = model.getValueAt(row, 0).toString();

String prevstart = model.getValueAt(row, 1).toString();

String prevend = model.getValueAt(row, 2).toString();

String prevint10 = model.getValueAt(row, 3).toString();

String prevint16 = model.getValueAt(row, 4).toString();

String prevint20 = model.getValueAt(row, 5).toString();

String prevint00 = model.getValueAt(row, 6).toString();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

switch (col) {

case(0):

CellText = JOptionPane.showInputDialog(table, "Change to : ","Edit the station name", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

try {

checkRepeat(CellText);

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(prevname);

Station.setName(CellText);

em.merge(Station);

Singleton.finishEMandTrans(em);

ButtonEditor.label = CellText;

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The station name field reapeats.", myEx);

log.warn("The new cell will not be added.", myEx);

}

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevname, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(1):

CellText = JOptionPane.showInputDialog(table, "Change to (XX:XX): ","Edit the start time", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(prevname);

Station.setStart(CellText);

em.merge(Station);

Singleton.finishEMandTrans(em);

ButtonEditor.label = CellText;

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevstart, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(2):

CellText = JOptionPane.showInputDialog(table, "Change to (XX:XX): ","Edit the end time", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(prevname);

Station.setEnd(CellText);

em.merge(Station);

Singleton.finishEMandTrans(em);

ButtonEditor.label = CellText;

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevend, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(3):

CellText = JOptionPane.showInputDialog(table, "Change to (1-60 min): ","Edit the interval to 10", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(prevname);

Station.setInt1(CellText);

em.merge(Station);

Singleton.finishEMandTrans(em);

ButtonEditor.label = CellText;

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevint10, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(4):

CellText = JOptionPane.showInputDialog(table, "Change to (1-60 min): ","Edit the interval to 16", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(prevname);

Station.setInt2(CellText);

em.merge(Station);

Singleton.finishEMandTrans(em);

ButtonEditor.label = CellText;

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevint16, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(5):

CellText = JOptionPane.showInputDialog(table, "Change to (1-60 min): ","Edit the interval to 20", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(prevname);

Station.setInt3(CellText);

em.merge(Station);

Singleton.finishEMandTrans(em);

ButtonEditor.label = CellText;

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevint20, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

case(6):

CellText = JOptionPane.showInputDialog(table, "Change to (1-60 min): ","Edit the interval past 20", JOptionPane.PLAIN\_MESSAGE);

try{

CheckTextField(CellText, col);

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(prevname);

Station.setInt4(CellText);

em.merge(Station);

Singleton.finishEMandTrans(em);

ButtonEditor.label = CellText;

table.setValueAt(CellText, row, col);

log.info(row +"," + col + "-cell has been edited.");

}

catch(WrongTimeException | EmptyFieldException | WrongFieldException | WrongDateException | EmptySearchFieldException myEx) {

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

table.setValueAt(prevint00, row, col);

log.info("Error - incorrect filling of the cell.", myEx);

log.warn("The contents of the cell will remain the same.", myEx);

}

break;

}

}});

//добавление новой строки после выделенной

Add.addActionListener (new ActionListener() {

public void actionPerformed (ActionEvent event){

log.info("Adding a new row to the table after the selected one.");

row = table.getSelectedRow() + 1;

JTextField add\_name = new JTextField();

JTextField add\_start = new JTextField();

JTextField add\_end = new JTextField();

JTextField add\_int10 = new JTextField();

JTextField add\_int16 = new JTextField();

JTextField add\_int20 = new JTextField();

JTextField add\_int00 = new JTextField();

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Save");

Object[] message = {

"Station: ", add\_name,

"Start time (XX:XX): ", add\_start,

"End time (XX:XX): ", add\_end,

"Time interval to 10 (1-60 min): ", add\_int10,

"Time interval to 16 (1-60 min): ", add\_int16,

"Time interval to 20 (1-60 min): : ", add\_int20,

"Time interval past 20 (1-60 min): ", add\_int00

};

int option = JOptionPane.showConfirmDialog(table, message, "New row", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

if(option == JOptionPane.OK\_OPTION) {

int exeption\_sum = 0;

try{

log.info(row +"," + 0 + "-cell is being filled in.");

CheckTextField(add\_name.getText(), 0);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 0 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 1 + "-cell is being filled in.");

CheckTextField(add\_start.getText(), 1);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 1 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 2 + "-cell is being filled in.");

CheckTextField(add\_end.getText(), 2);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 2 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 3 + "-cell is being filled in.");

CheckTextField(add\_int10.getText(), 3);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 3 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 4 + "-cell is being filled in.");

CheckTextField(add\_int16.getText(), 4);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 4 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 5 + "-cell is being filled in.");

CheckTextField(add\_int20.getText(), 5);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 5 + "-cell has been filled in wrong.");

}

try{

log.info(row +"," + 6 + "-cell is being filled in.");

CheckTextField(add\_int00.getText(), 6);

}

catch( WrongDateException | WrongTimeException | EmptyFieldException | WrongFieldException | EmptySearchFieldException myEx ) {

exeption\_sum++;

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info(row +"," + 6 + "-cell has been filled in wrong.");

}

try{

if(exeption\_sum > 0) throw new WrongAddException();

try {

checkRepeat(add\_name.getText());

model.insertRow(row, new String [] {add\_name.getText(), add\_start.getText(), add\_end.getText(),

add\_int10.getText(), add\_int16.getText(), add\_int20.getText(), add\_int00.getText()});

EntityManager em = Singleton.createEMandTrans();

EntityRouteList OneRoute = DBOperations.searchRouteWithName(routename);

EntityStationList newStation = new EntityStationList(add\_name.getText(), add\_start.getText(), add\_end.getText(),

add\_int10.getText(), add\_int16.getText(), add\_int20.getText(), add\_int00.getText(), OneRoute);

em.persist(newStation);

Singleton.finishEMandTrans(em);

}

catch (RepeatAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The station field reapeats.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

catch(WrongAddException myEx){

UIManager.put("OptionPane.okButtonText", "OK");

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("The new line fields were filled in incorrectly.", myEx);

log.warn("The new row will not be added.", myEx);

}

}

}

});

//удаление строк

Delete.addActionListener(new ActionListener(){

public void actionPerformed (ActionEvent event){

log.info("Deletion of selected rows from the table.");

try {

checkSelection(selectedRows);

UIManager.put("OptionPane.cancelButtonText", "Cancel");

UIManager.put("OptionPane.okButtonText", "Confirm");

int option = JOptionPane.showConfirmDialog(table, "After confirmation, it will be impossible to cancel the deletion of the selected rows.", "Delete?", JOptionPane.OK\_CANCEL\_OPTION,JOptionPane.PLAIN\_MESSAGE);

log.warn("After confirmation, it will be impossible to cancel the deletion of the selected rows.");

if (option == JOptionPane.OK\_OPTION) {

UIManager.put("OptionPane.okButtonText", "OK");

for(int i = selectedRows.length-1; i > -1; i--){

EntityManager em = Singleton.createEMandTrans();

EntityStationList Station = DBOperations.searchStationWithName(String.valueOf(table.getModel().getValueAt(selectedRows[i], 0)));

em.merge(Station);

Singleton.finishEMandTrans(em);

model.removeRow(selectedRows[i]);

DBOperations.deleteStation(Station.getId());

}

JOptionPane.showMessageDialog (frame, "The cell data of the selected rows has been deleted from the table.","Deletion message", JOptionPane.PLAIN\_MESSAGE);

log.info("The selected rows are removed from the table.");

}

}

catch(MyExceptionDelete myEx) {

JOptionPane.showMessageDialog(table, myEx.getMessage());

log.info("Исключительная ситуация - нет выделенных строк для удаления. ", myEx);

log.warn("Необходимо выделить строки для удаления, иначе удаление будет недоступно. ");

}

}

});

}

//получение данных из БД для таблицы

private void getDBList() {

// получение списка из БД

EntityRouteList OneRoute = DBOperations.searchRouteWithName(routename);

List<EntityStationList> Stations = OneRoute.getSts();

ArrayList<String[]> values = new ArrayList<String[]>(Stations.size() + 1);

for (int i = 0; i < Stations.size(); i++) {

EntityStationList entity = Stations.get(i);

//if(entity.getRoute().getId() == OneRoute.getId()) {

values.add(new String[]{entity.getName(),entity.getStart(),entity.getEnd(),

entity.getInt1(), entity.getInt2(), entity.getInt3(), entity.getInt4()});

}

// преобразование в формат для заполнения таблицы данными.

data = new String[values.size()][7];

for (int i = 0; i < values.size(); i++) data[i] = values.get(i);

}

//подсчет подстрок в строке

public static int count\_substring(String str, String target) {

return (str.length() - str.replace(target, "").length()) / target.length();

}

//удаление подстрок в строке

public static String remove\_substring(String str, String substr) {

int count = count\_substring(str, substr);

if (count > 0) {

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

str = str.replace(substr, "");

}

return str;

}

//проверка введенного кол-ва минут

public int CheckMinNumber (String text){

int flag = 0;

if (Integer.valueOf(text) > 0 && Integer.valueOf(text) <= 300 ) {

flag = 1;

}

return flag;

}

//проверка времени

public int CheckTime (String text){

String[] premSplitString = text.split("\\:");

int[] premSplit = new int[] {Integer.valueOf(premSplitString[0]), Integer.valueOf(premSplitString[1])};

int flag = 0;

if (premSplit[0] >= 0 && premSplit[0] < 24 && premSplit[1] >= 0 && premSplit[1] < 60 ) {

flag = 1;

}

return flag;

}

//проверка повтора строки

private void checkRepeat(String str) throws RepeatAddException{

for (int i = 0; i < table.getRowCount(); i++) {

if (table.getModel().getValueAt(i, 0).equals(str)) {

throw new RepeatAddException();

}

}

}

public class RepeatAddException extends Exception {

public RepeatAddException() {

super("It is restricted to add a row with the same driver license id! Please, try again and change this field.");

}

}

//обработка заполнения текстового поля

public void CheckTextField (String text, int col) throws EmptyFieldException, WrongFieldException, WrongDateException, WrongTimeException, EmptySearchFieldException {

if (col == 0 && !text.matches("^[ .A-Z,a-z-]+$")) throw new WrongFieldException(col);

if (col != 10 && col!=0 && (text.trim().isEmpty() || text == null) ) throw new EmptyFieldException(col);

if (col == 10 && (text.trim().isEmpty() || text == null)) throw new EmptySearchFieldException();

if ((col == 3 || col == 4 || col == 5 || col == 6) &&

(!text.matches("^[1-9]\\d\*$") || CheckMinNumber(text) == 0)) throw new WrongFieldException(col);

if ((col == 1 || col == 2) && !text.matches("^\\d{2}:\\d{2}$") && CheckTime(text) == 0) throw new WrongFieldException(col);

}

//поле поиска пустое

public class EmptySearchFieldException extends Exception {

public EmptySearchFieldException() {

super("The search field was empty! Enter the search text again!");

}

}

//поле пустое

public class EmptyFieldException extends Exception {

public EmptyFieldException(int col) {

super("The field in the column " + col + " was empty! Fill in the cell again!");

}

}

//поле заполнено неверно

public class WrongFieldException extends Exception {

public WrongFieldException(int col) {

super("The field in the column " + col + " was filled in wrong!");

}

}

//время введено неверно

public class WrongTimeException extends Exception {

public WrongTimeException() {

super("The filled in time is wrong! Fill in the cell again and enter an existing time!");

}

}

//дата введена неверно

public class WrongDateException extends Exception {

public WrongDateException() {

super("The filled in date is wrong! Fill in the cell again and enter an existing date!");

}

}

//обработка удаления строк

public void checkSelection (int[] arr) throws MyExceptionDelete{

int len = arr.length;

if (len == 0) {

throw new MyExceptionDelete();

}

}

public class MyExceptionDelete extends Exception {

public MyExceptionDelete() {

super ("No row selected for deletion or export.");

}

}

//поля для добовления строки были введены неправильно

public class WrongAddException extends Exception {

public WrongAddException() {

super ("The new line fields were filled in incorrectly!");

}

}

//демонстрация окна

public void show() {

log.info("Demonstration of a window with a list of stations.");

frame.setVisible(true);

}

public static void main(String[] args) {

log.info("Demonstration of a window with a list of stations.");

new GuiStationList("122 round").show();

}

}

1. **DBOperations**

package back\_database;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashSet;

import java.util.List;

import java.util.stream.Collectors;

import back\_entities.EntityDriverList;

import back\_entities.EntityOffenceList;

import back\_entities.EntityRouteList;

import back\_entities.EntityStationList;

import jakarta.persistence.EntityManager;

import jakarta.persistence.Query;

import jakarta.persistence.criteria.CriteriaBuilder;

import jakarta.persistence.criteria.CriteriaQuery;

import jakarta.persistence.criteria.Root;

public class DBOperations {

//отсортированный список нарушений

//использование: выгрузка данных из БД в таблицу

public static List<EntityOffenceList> getSortedOffences() {

EntityManager em = Singleton.createEMandTrans(); // создание объекта

CriteriaBuilder cb = em.getCriteriaBuilder(); // создание объекта запроса, изменяется подробностями запроса

CriteriaQuery<EntityOffenceList> cq = cb.createQuery(EntityOffenceList.class);

Root<EntityOffenceList> root = cq.from(EntityOffenceList.class); // корень запроса

cq.orderBy(cb.asc(root.get("driver")));

List<EntityOffenceList> Offences = em.createQuery(cq).getResultList();

Singleton.finishEMandTrans(em);

return Offences;

}

//отсортированный список номеров прав водителей

//нарушения: заполнение выпадаюших списков

public static String[] getSortedLicences() {

List<EntityDriverList> Drivers = getSortedDrivers();

List<String> licencesList = new ArrayList<String>();

licencesList = Drivers.stream().map(EntityDriverList::getLicense).collect(Collectors.toList());

licencesList = new ArrayList<>(new HashSet<>(licencesList));

String[] licenses = licencesList.toArray(new String[licencesList.size()]);

Arrays.sort(licenses);

return licenses;

}

//отсортированный список маршрутов

//нарушения: заполнение выпадаюших списков

public static String[] getSortedRouteNames() {

List<EntityRouteList> routesList = getSortedRoutes();

List<String> rnamesList = new ArrayList<String>();

rnamesList = routesList.stream().map(EntityRouteList::getName).collect(Collectors.toList());

rnamesList = new ArrayList<>(new HashSet<>(rnamesList));

String[] rnames = rnamesList.toArray(new String[rnamesList.size()]);

Arrays.sort(rnames);

return rnames;

}

//получение водителя по номеру прав

//использование:

//нарушения: дабавление строки в БД

//водители: сохранение после редактирования ячейки, удаление

public static EntityDriverList searchDriverWithLicense(String license) {

EntityManager em = Singleton.createEMandTrans();

Query qSess = em.createQuery("SELECT u FROM EntityDriverList u WHERE u.license = :license ");

qSess.setParameter("license", license);

EntityDriverList result = (EntityDriverList) qSess.getResultList().get(0);

return result;

}

//получение нарушения по номеру прав, дате и времени

//использование: сохранение после редактирования ячейки, удаление

public static EntityOffenceList searchOffence(String date, String time, String driver) {

EntityManager em = Singleton.createEMandTrans();

Query qSess = em.createQuery("SELECT u FROM EntityOffenceList u WHERE u.date = :date "

+ "AND u.time = :time AND u.driver = :driver");

qSess.setParameter("date", date);

qSess.setParameter("time", time);

qSess.setParameter("driver", driver);

EntityOffenceList result = (EntityOffenceList) qSess.getResultList().get(0);

return result;

}

//удаление нарушений по id

public static void deleteOffence(int offence\_id){

EntityManager em = Singleton.createEMandTrans();

Query offences = em.createQuery("DELETE FROM EntityOffenceList e WHERE e.id = :offence\_id");

offences.setParameter("offence\_id", offence\_id);

offences.executeUpdate();

Singleton.finishEMandTrans(em);

}

//отсортированный список водителей

//использование: выгрузка данных из БД в таблицу

public static List<EntityDriverList> getSortedDrivers() {

EntityManager em = Singleton.createEMandTrans();

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<EntityDriverList> cq = cb.createQuery(EntityDriverList.class);

Root<EntityDriverList> root = cq.from(EntityDriverList.class);

cq.orderBy(cb.asc(root.get("license")));

List<EntityDriverList> Drivers = em.createQuery(cq).getResultList();

Singleton.finishEMandTrans(em);

return Drivers;

}

//удаление водителей по id

public static void deleteDriver(int driver\_id){

EntityManager em = Singleton.createEMandTrans();

Query drivers = em.createQuery("DELETE FROM EntityDriverList e WHERE e.id = :driver\_id");

drivers.setParameter("driver\_id", driver\_id);

drivers.executeUpdate();

Singleton.finishEMandTrans(em);

}

//отсортированный список маршрутов

//использование: выгрузка данных из БД в таблицу

public static List<EntityRouteList> getSortedRoutes() {

EntityManager em = Singleton.createEMandTrans();

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<EntityRouteList> cq = cb.createQuery(EntityRouteList.class);

Root<EntityRouteList> root = cq.from(EntityRouteList.class);

cq.orderBy(cb.asc(root.get("name")));

List<EntityRouteList> Routes = em.createQuery(cq).getResultList();

Singleton.finishEMandTrans(em);

return Routes;

}

//получение маршрута по названию

//использование:

//маршруты: сохранение после редактирования ячейки, удаление

//остановки: дабавление строки в БД

public static EntityRouteList searchRouteWithName(String name) {

EntityManager em = Singleton.createEMandTrans();

Query qSess = em.createQuery("SELECT u FROM EntityRouteList u WHERE u.name = :name ");

qSess.setParameter("name", name);

EntityRouteList result = (EntityRouteList) qSess.getResultList().get(0);

return result;

}

//удаление маршрутов по id

public static void deleteRoute(int route\_id){

EntityManager em = Singleton.createEMandTrans();

Query routes = em.createQuery("DELETE FROM EntityRouteList e WHERE e.id = :route\_id");

routes.setParameter("route\_id", route\_id);

routes.executeUpdate();

Singleton.finishEMandTrans(em);

}

//отсортированный список остановок

//использование: выгрузка данных из БД в таблицу

public static List<EntityStationList> getSortedStations() {

EntityManager em = Singleton.createEMandTrans();

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<EntityStationList> cq = cb.createQuery(EntityStationList.class);

Root<EntityStationList> root = cq.from(EntityStationList.class);

cq.orderBy(cb.asc(root.get("name")));

List<EntityStationList> Stations = em.createQuery(cq).getResultList();

Singleton.finishEMandTrans(em);

return Stations;

}

//получение остановок по названию

//использование: сохранение после редактирования ячейки, удаление

public static EntityStationList searchStationWithName(String name) {

EntityManager em = Singleton.createEMandTrans();

Query qSess = em.createQuery("SELECT u FROM EntityStationList u WHERE u.name = :name ");

qSess.setParameter("name", name);

EntityStationList result = (EntityStationList) qSess.getResultList().get(0);

return result;

}

//удаление остановок по id

public static void deleteStation(int station\_id){

EntityManager em = Singleton.createEMandTrans();

Query stations = em.createQuery("DELETE FROM EntityStationList e WHERE e.id = :station\_id");

stations.setParameter("station\_id", station\_id);

stations.executeUpdate();

Singleton.finishEMandTrans(em);

}

//получение списка остановок по названию маршрута

//использование: получение массива остановок в классе EntityRouteList

public static List<EntityStationList> getStations(String routename) {

EntityRouteList OneRoute = DBOperations.searchRouteWithName(routename);

List<EntityStationList> Stations = DBOperations.getSortedStations();

List<EntityStationList> arrStations = new ArrayList<>();

for (int i = 0; i < Stations.size(); i++) {

EntityStationList entity = Stations.get(i);

if(entity.getRoute().getId() == OneRoute.getId()) {

arrStations.add(entity);

}

}

return arrStations;

}

//получение списка нарушений по номеру прав водителя

//использование: получение массива нарушений в классе EntityDriverList

public static List<EntityOffenceList> getOffences(String license) {

EntityDriverList OneDriver = DBOperations.searchDriverWithLicense(license);

List<EntityOffenceList> Offences = DBOperations.getSortedOffences();

List<EntityOffenceList> arrOffences = new ArrayList<>();

for (int i = 0; i < Offences.size(); i++) {

EntityOffenceList entity = Offences.get(i);

if(entity.getIdDriver().getId() == OneDriver.getId()) {

arrOffences.add(entity);

}

}

return arrOffences;

}

}

1. **Singleton**

package back\_database;

import jakarta.persistence.EntityManager;

import jakarta.persistence.EntityManagerFactory;

import jakarta.persistence.Persistence;

//Синглтон для использования единого во всем проекте EntityManagerFactory,

//а также общей работе с EntityManager.

public class Singleton {

//Экземпляр класса

private static Singleton instance = null;

//Фабрика EntityManager для работы с БД

private EntityManagerFactory emf;

//Создание фабрики

private Singleton() {

emf = Persistence.createEntityManagerFactory("test\_persistence");

}

//Cоздание фабрики или получение экземпляра класса с уже созданной фабрикой

public static Singleton getInstance() {

if (instance == null) {

instance = new Singleton();

}

return instance;

}

//Получение фабрики

public EntityManagerFactory getEntityManagerFactory() {

return emf;

}

// Создание интерфейса для работы с БД и открытие транзакции

public static EntityManager createEMandTrans() {

EntityManager em = getInstance().getEntityManagerFactory().createEntityManager();

em.getTransaction().begin();

return em;

}

//Закрытие интерфейса и транзации

public static void finishEMandTrans(EntityManager em) {

em.getTransaction().commit();

em.close();

}

}

1. **ButtonEditor**

package back\_gui;

import java.awt.Component;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowEvent;

import java.util.EventObject;

import javax.swing.DefaultCellEditor;

import javax.swing.JButton;

import javax.swing.JCheckBox;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import javax.swing.event.CellEditorListener;

import javax.swing.table.TableCellEditor;

import front.GuiRouteList;

import front.GuiStationList;

public class ButtonEditor extends DefaultCellEditor {

protected JButton button;

public static String label;

private boolean isPushed;

private GuiRouteList guiRouteList1;

public ButtonEditor(GuiRouteList guiRouteList, JCheckBox checkBox) {

super(checkBox);

guiRouteList1 = guiRouteList;

button = new JButton();

button.setOpaque(true);

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

fireEditingStopped();

}

});

}

public Component getTableCellEditorComponent(JTable table, Object value,

boolean isSelected, int row, int column) {

if (isSelected) {

button.setForeground(table.getSelectionForeground());

button.setBackground(table.getSelectionBackground());

} else {

button.setForeground(table.getForeground());

button.setBackground(table.getBackground());

}

label = (value == null) ? "" : value.toString();

button.setText(label);

isPushed = true;

return button;

}

public void windowClosing(WindowEvent event) {

event.getWindow().setVisible(false);

}

public Object getCellEditorValue() {

if (isPushed) {

guiRouteList1.close();

GuiStationList guiStationList = new GuiStationList(label);

guiStationList.show();

}

isPushed = false;

return new String(label);

}

public boolean stopCellEditing() {

isPushed = false;

return super.stopCellEditing();

}

protected void fireEditingStopped() {

super.fireEditingStopped();

}

}

1. **ButtonRenderer**

package back\_gui;

import java.awt.Component;

import javax.swing.JButton;

import javax.swing.JTable;

import javax.swing.UIManager;

import javax.swing.table.TableCellRenderer;

public class ButtonRenderer extends JButton implements TableCellRenderer {

public ButtonRenderer() {

setOpaque(true);

}

public Component getTableCellRendererComponent(JTable table, Object value,

boolean isSelected, boolean hasFocus, int row, int column) {

if (isSelected) {

setForeground(table.getSelectionForeground());

setBackground(table.getSelectionBackground());

} else {

setForeground(table.getForeground());

setBackground(UIManager.getColor("Button.background"));

}

setText((value == null) ? "" : value.toString());

return this;

}

}

1. **SprinfUtilities**

package back\_gui;

import javax.swing.\*;

import javax.swing.SpringLayout;

import java.awt.\*;

/\*

\* Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* - Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* - Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\*

\* - Neither the name of Oracle or the names of its

\* contributors may be used to endorse or promote products derived

\* from this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS

\* IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

\* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR

\* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR

\* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

\* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

\* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR

\* PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF

\* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING

\* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

\* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

/\*\*

\* A 1.4 file that provides utility methods for

\* creating form- or grid-style layouts with SpringLayout.

\* These utilities are used by several programs, such as

\* SpringBox and SpringCompactGrid.

\*/

public class SpringUtilities {

/\*\*

\* A debugging utility that prints to stdout the component's

\* minimum, preferred, and maximum sizes.

\*/

public static void printSizes(Component c) {

System.out.println("minimumSize = " + c.getMinimumSize());

System.out.println("preferredSize = " + c.getPreferredSize());

System.out.println("maximumSize = " + c.getMaximumSize());

}

/\*\*

\* Aligns the first <code>rows</code> \* <code>cols</code>

\* components of <code>parent</code> in

\* a grid. Each component is as big as the maximum

\* preferred width and height of the components.

\* The parent is made just big enough to fit them all.

\*

\* @param rows number of rows

\* @param cols number of columns

\* @param initialX x location to start the grid at

\* @param initialY y location to start the grid at

\* @param xPad x padding between cells

\* @param yPad y padding between cells

\*/

public static void makeGrid(Container parent,

int rows, int cols,

int initialX, int initialY,

int xPad, int yPad) {

SpringLayout layout;

try {

layout = (SpringLayout)parent.getLayout();

} catch (ClassCastException exc) {

System.err.println("The first argument to makeGrid must use SpringLayout.");

return;

}

Spring xPadSpring = Spring.constant(xPad);

Spring yPadSpring = Spring.constant(yPad);

Spring initialXSpring = Spring.constant(initialX);

Spring initialYSpring = Spring.constant(initialY);

int max = rows \* cols;

//Calculate Springs that are the max of the width/height so that all

//cells have the same size.

Spring maxWidthSpring = layout.getConstraints(parent.getComponent(0)).

getWidth();

Spring maxHeightSpring = layout.getConstraints(parent.getComponent(0)).

getHeight();

for (int i = 1; i < max; i++) {

SpringLayout.Constraints cons = layout.getConstraints(

parent.getComponent(i));

maxWidthSpring = Spring.max(maxWidthSpring, cons.getWidth());

maxHeightSpring = Spring.max(maxHeightSpring, cons.getHeight());

}

//Apply the new width/height Spring. This forces all the

//components to have the same size.

for (int i = 0; i < max; i++) {

SpringLayout.Constraints cons = layout.getConstraints(

parent.getComponent(i));

cons.setWidth(maxWidthSpring);

cons.setHeight(maxHeightSpring);

}

//Then adjust the x/y constraints of all the cells so that they

//are aligned in a grid.

SpringLayout.Constraints lastCons = null;

SpringLayout.Constraints lastRowCons = null;

for (int i = 0; i < max; i++) {

SpringLayout.Constraints cons = layout.getConstraints(

parent.getComponent(i));

if (i % cols == 0) { //start of new row

lastRowCons = lastCons;

cons.setX(initialXSpring);

} else { //x position depends on previous component

cons.setX(Spring.sum(lastCons.getConstraint(SpringLayout.EAST),

xPadSpring));

}

if (i / cols == 0) { //first row

cons.setY(initialYSpring);

} else { //y position depends on previous row

cons.setY(Spring.sum(lastRowCons.getConstraint(SpringLayout.SOUTH),

yPadSpring));

}

lastCons = cons;

}

//Set the parent's size.

SpringLayout.Constraints pCons = layout.getConstraints(parent);

pCons.setConstraint(SpringLayout.SOUTH,

Spring.sum(

Spring.constant(yPad),

lastCons.getConstraint(SpringLayout.SOUTH)));

pCons.setConstraint(SpringLayout.EAST,

Spring.sum(

Spring.constant(xPad),

lastCons.getConstraint(SpringLayout.EAST)));

}

/\* Used by makeCompactGrid. \*/

private static SpringLayout.Constraints getConstraintsForCell(

int row, int col,

Container parent,

int cols) {

SpringLayout layout = (SpringLayout) parent.getLayout();

Component c = parent.getComponent(row \* cols + col);

return layout.getConstraints(c);

}

/\*\*

\* Aligns the first <code>rows</code> \* <code>cols</code>

\* components of <code>parent</code> in

\* a grid. Each component in a column is as wide as the maximum

\* preferred width of the components in that column;

\* height is similarly determined for each row.

\* The parent is made just big enough to fit them all.

\*

\* @param rows number of rows

\* @param cols number of columns

\* @param initialX x location to start the grid at

\* @param initialY y location to start the grid at

\* @param xPad x padding between cells

\* @param yPad y padding between cells

\*/

public static void makeCompactGrid(Container parent,

int rows, int cols,

int initialX, int initialY,

int xPad, int yPad) {

SpringLayout layout;

try {

layout = (SpringLayout)parent.getLayout();

} catch (ClassCastException exc) {

System.err.println("The first argument to makeCompactGrid must use SpringLayout.");

return;

}

//Align all cells in each column and make them the same width.

Spring x = Spring.constant(initialX);

for (int c = 0; c < cols; c++) {

Spring width = Spring.constant(0);

for (int r = 0; r < rows; r++) {

width = Spring.max(width,

getConstraintsForCell(r, c, parent, cols).

getWidth());

}

for (int r = 0; r < rows; r++) {

SpringLayout.Constraints constraints =

getConstraintsForCell(r, c, parent, cols);

constraints.setX(x);

constraints.setWidth(width);

}

x = Spring.sum(x, Spring.sum(width, Spring.constant(xPad)));

}

//Align all cells in each row and make them the same height.

Spring y = Spring.constant(initialY);

for (int r = 0; r < rows; r++) {

Spring height = Spring.constant(0);

for (int c = 0; c < cols; c++) {

height = Spring.max(height,

getConstraintsForCell(r, c, parent, cols).

getHeight());

}

for (int c = 0; c < cols; c++) {

SpringLayout.Constraints constraints =

getConstraintsForCell(r, c, parent, cols);

constraints.setY(y);

constraints.setHeight(height);

}

y = Spring.sum(y, Spring.sum(height, Spring.constant(yPad)));

}

//Set the parent's size.

SpringLayout.Constraints pCons = layout.getConstraints(parent);

pCons.setConstraint(SpringLayout.SOUTH, y);

pCons.setConstraint(SpringLayout.EAST, x);

}

}