**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

Курсовая РАБОТА

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Объектно-ориентированное программирование

|  |  |  |
| --- | --- | --- |
| **Студенты гр. 4891** |  | Рахимов Г.В. |
|  |  | Титинина.А.А. |
| **Преподаватель** |  | Кочетков А.В. |

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на курсовую работу**

|  |
| --- |
| **Студенты:** Рахимов Г.В., Титинина А.А. |
| **Группа:** 4891 |
| **Тема работы:** построение приложения с графическим пользовательским интерфейсом. |
| **Исходные данные:**  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 ведения:   1. Список водителей, их стаж работы и класс; 2. Время начала или окончания движения автобусов на всех или отдельных маршрутах; 3. Справка о графике движения автобусов и отчет о его нарушениях. |

**содержание**

[Введение 4](#_Toc510977628)

[Теоретическое обоснование 5](#_Toc510977629)

[1. Архитектура проекта 5](#_Toc510977630)

[Проектирование 7](#_Toc510977631)

[1. Разработка интерфейса пользователя 7](#_Toc510977632)

[2. Обработка событий 11](#_Toc510977633)

[3. Обработка исключений 12](#_Toc510977634)

[4. Протоколирование работы системы 12](#_Toc510977635)

[Заключение 13](#_Toc510977636)

[Список использованных источников 14](#_Toc510977637)

[Приложение А 15](#_Toc510977638)

[1. Основные классы 15](#_Toc510977639)

# Введение

**Цель курсовой работы:** приобрести навыки построения программных комплексов с графическим пользовательским интерфейсом. Закрепление и углубление теоретических знаний, приобретение практических навыков по проектированию и разработке программного обеспечения на объектно-ориентированном языке.

# Теоретическое обоснование

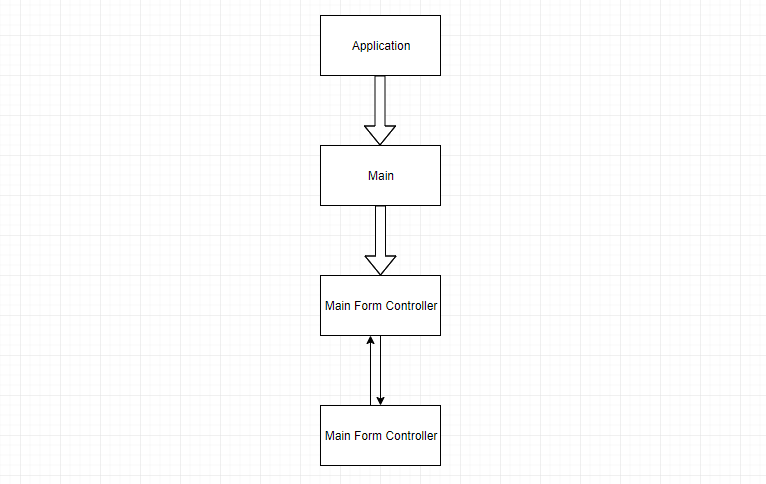
## Архитектура проекта

При построении программного комплекса использовался объектно-ориентированный язык Java версии 8. В качестве библиотеки для построения графического пользовательского интерфейса была использована JavaFX.

**JavaFX** — платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing, является более гибкой и позволяет моделировать сложные интерфейсы.

В JavaFX для построения приложения используется широко распространенный шаблон проектирования Model View Controller, что позволяет легко масштабировать приложение по мере необходимости.

Рисунок 1 – Схема архитектуры MVC



В качестве источника данных используется .xml файл

Схема 1 – Модель хранения данных

<?xml version="1.0" encoding="UTF-8"?>

<items>

<drivers>

<driver class="${driverClass}" experience="${driverExperience}" name="${driverName}" surname="${driverSurname}" />

</drivers>

<buses>

<bus busRouteNumber="${busRouteNumber}" governmentNumber="${governmentNumber}" />

</buses>

<routes>

<route busRouteNumber="${busRouteNumber}" endTime="${routeStartTime}" startTime="${routeEndTime}" />

</routes>

<violations>

<violation violatedDriver="${violatedDriver}" violatedRouteNumber="${busRouteNumber}" violationDate="${violationDate}" violationDescription="${violationDescription}" />

</violations>

</items>

Бизнес логика построена на основе контроллеров, которые взаимодействуя с моделью и хранилищем данных, преобразуют исходные данные в подходящий для отображения вид.

# Проектирование

## Разработка интерфейса пользователя

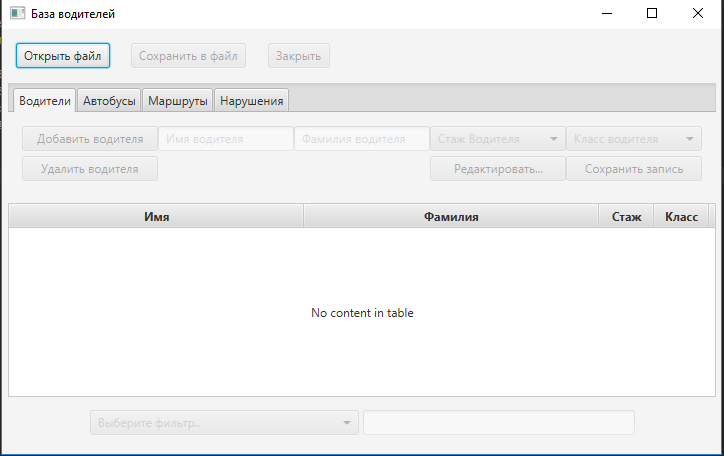
Графический интерфейс пользователя представляет собой совокупность визуальных средств, с помощью которых пользователь взаимодействует с программой. Графический интерфейс пользователя строится в несколько этапов.

JavaFX позволяет проектировать экранные формы в декларативном стиле, для этого используется язык разметки XML - eXtensible Markup Language, что значительно облегчается разработку и дальнейшее сопровождение программного обеспечения. В данном проекте использовалась платформа JavaFX версии 8.

MainForm.xml

На рисунке 3 изображена главная страница, она содержит в себе компоненты TabPane – вкладки, предназначенные для разделения приложения на логические области. Главная страница является шаблоном для последующих видов, подключаемых посредством XML из внешних файлов. Такое разделение удобно для разработчика т.к. позволяет сократить кол-во кода в одном файле, тем самым упрощая работу с представлением. Первая вкладка содержит информацию о водителях, вторая о автобусах, третья о маршрутах, четвертая о нарушениях.

Рисунок 3 – Основное окно при запуске



Как можно увидеть, все кнопки и поля недоступны для использования по умолчанию. Сразу как выполняется операция открытия файла, часть элементов становится доступна.

Рисунок 4 - Водители

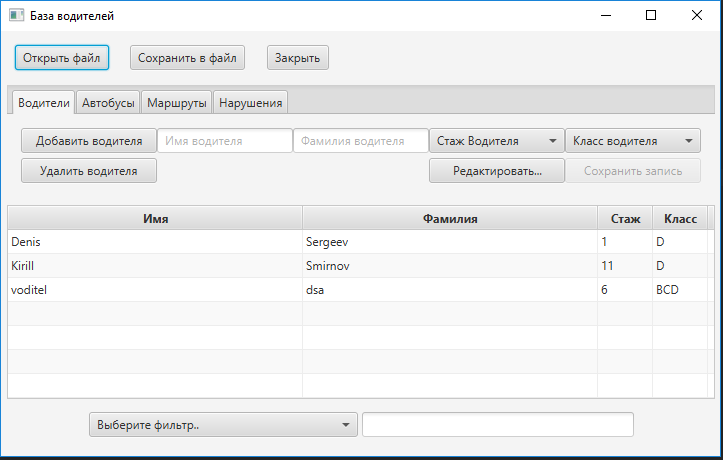


Рисунок 5 - Автобусы

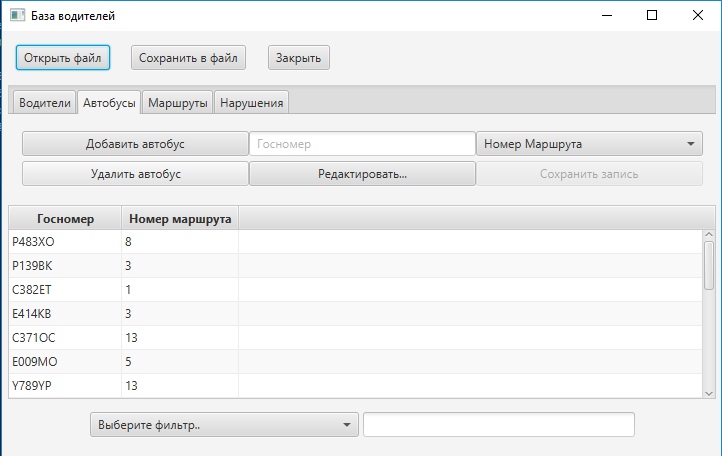


Рисунок 5 - Маршруты

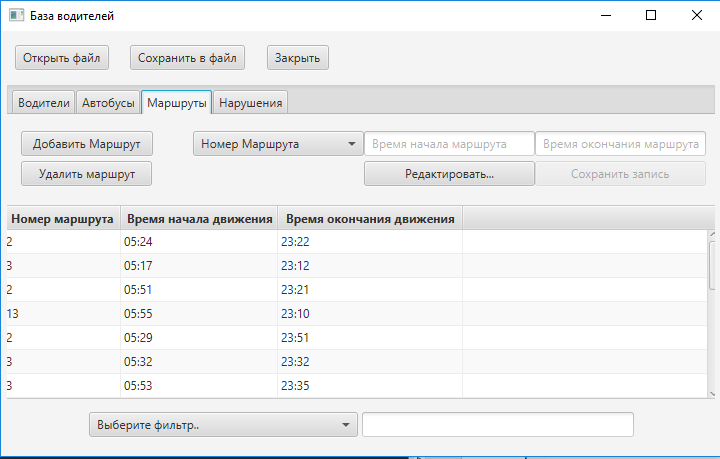
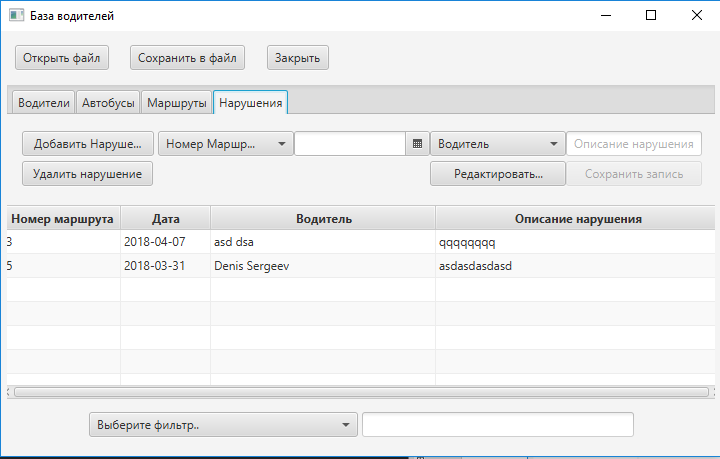


Рисунок 5 - Нарушения



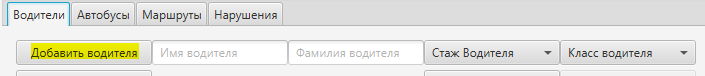
## Обработка событий

В JavaFX используется стандартная модель обработки событий, точно такая же, как и в библиотеке графических пользовательских интерфейсов Swing. Отличие JavaFX от Swing состоит в том, что JavaFX позволяет выбрать и другую модель обработки событий, основанную на аннотациях Java и взаимодействия их с представлениями в формате XML во время выполнения программы.

В методах контроллеров определяются т.н. слушатели, которые ожидают сигнала от пользовательского интерфейса о том, что пользователь совершил определенное действие**,** по результату совершенного действия контроллер определяет необходимость выполнения тех или иных команд. Для декларативного вида обработки событий, метод контроллера, который должен обработать то или иное событие, должен быть помечен аннотацией @FXML и его название должно соответствовать названию метода, который указан в файле представления.

Рассмотрим на примере добавления водителя

1. Для этого пользователю необходимо нажать на соответствующую кнопку в меню приложения.



1. Представление имеет информацию о методе обработчике данного события. В данном случае это метод createDriver (), который находится в классе контроллере, связанным с данным представлением.
2. Класс контроллер в свою очередь вызывает необходимый метод. Который подготавливает новую модель для хранения информации о новом сотруднике и вызывает соответствующую форму, необходимую пользователю для того чтобы ввести данные.

Данные, заполненные в поля рядом, преобразуются в поля класса Driver, после чего создается новый экземпляр данного класса.

1. Как результат выполнения, новая запись будет добавлена в таблицу.

## Обработка исключений

В ходе работы программы неминуемо возникают исключительные ситуации, которые должны быть обработаны определенным образом, для поддержания работоспособности программы. В Java имеются специальные классы, которые представляют собой такие исключительные ситуации. Класс **Exception** расширяющий класс **Throwable** представляет базовый тип так называемых перехватываемых исключений. На основе класса Exception мы можем создавать исключения, удовлетворяющие нашим нуждам.

Перехватывая исключения средствами Java, такими как блоком try catch и обрабатывая его должным образом мы предотвращаем поломку программы при неправильно введенных пользовательских данных.

## Протоколирование работы системы

На этапе разработки приложения программист обычно пользуется пошаговыми средствами отладки, встроенными в инструментальную среду.

В ряде случаев бывает неудобно или невозможно пользоваться такими средствами. Например, при работе с отладчиком виден лишь небольшой фрагмент программы, что не всегда дает возможность понять причины возникновения ошибок; кроме того, приложение может быть установлено в рабочую среду, где нет инструментов разработки и отладки программы. В этом случае для контроля и анализа работы приложения используют методы протоколирования. Этот способ является одним из основных для выявления ошибок в том случае, если приложение запускается на удаленном компьютере.

# Заключение

В ходе выполнения данного курсового проекта нами были освоены современные технологии построения приложений с графическим пользовательским интерфейсом на объектно-ориентрованном языке программирования Java. Были укреплены и углублены теоретические знания, полученные ранее.

# Список использованных источников

[1] Oracle Java Documentation

[2] StackOverflow.com

# Приложение А

## Основные классы

Bus.java

package Classes;

public class Bus {

private String governmentNumber;

public static final String possibleLettersAndDigits = "ABCEHKMOPTXY0123456789";

private static final int[] possibleBusNumbers = {1, 2, 3, 5, 8, 13, 21};

private String busRouteNumber;

public Bus(String governmentNumber, String busRouteNumber) {

this.governmentNumber = governmentNumber;

this.busRouteNumber = busRouteNumber;

}

public void setGovernmentNumber(String governmentNumber) {

this.governmentNumber = governmentNumber;

}

public String getBusRouteNumber() {

return busRouteNumber;

}

public String getGovernmentNumber() {

return governmentNumber;

}

public void setBusRouteNumber(String busRouteNumber) {

this.busRouteNumber = busRouteNumber;

}

}

Driver.java

package Classes;

public class Driver {

private String driverName;

private String driverSurname;

private String driverExperience;

private String driverClass;

public Driver(String driverName, String driverSurname, String driverExperience, String driverClass) {

this.driverName = driverName;

this.driverSurname = driverSurname;

this.driverExperience = driverExperience;

this.driverClass = driverClass;

}

public String getDriverName() {

return driverName;

}

public String getDriverSurname() {

return driverSurname;

}

public String getDriverExperience() {

return driverExperience;

}

public String getDriverClass() {

return driverClass;

}

public void setDriverExperience(String driverExperience) {

this.driverExperience = driverExperience;

}

public void setDriverName(String driverName) {

this.driverName = driverName;

}

public void setDriverSurname(String driverSurname) {

this.driverSurname = driverSurname;

}

public void setDriverClass(String driverClass) {

this.driverClass = driverClass;

}

}

Route.java

package Classes;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

public class Route {

public static final String[] possibleBusNumbers = {"1", "2", "3", "5", "8", "13", "21"};

DateFormat simpleDateFormat = new SimpleDateFormat("HH:mm");

private String routeNumber;

private String routeStartTime;

private String routeEndTime;

public Route(String routeNumber, String routeStartTime, String routeEndTime) {

this.routeNumber = routeNumber;

this.routeStartTime = routeStartTime;

this.routeEndTime = routeEndTime;

}

public String getRouteNumber() {

return routeNumber;

}

public void setRouteNumber(String routeNumber) {

this.routeNumber = routeNumber;

}

public String getRouteStartTime() {

return routeStartTime;

}

public void setRouteStartTime(String routeStartTime) {

this.routeStartTime = routeStartTime;

}

public String getRouteEndTime() {

return routeEndTime;

}

public void setRouteEndTime(String routeEndTime) {

this.routeEndTime = routeEndTime;

}

}

Violation.java

package Classes;

public class Violation {

private String violatedRouteNumber;

private String violationDate;

private String violatedDriver;

private String violationDescription;

public Violation(String violatedRouteNumber, String violationDate, String violatedDriver, String violationDescription) {

this.violatedRouteNumber = violatedRouteNumber;

this.violationDate = violationDate;

this.violatedDriver = violatedDriver;

this.violationDescription = violationDescription;

}

public String getViolatedRouteNumber() {

return violatedRouteNumber;

}

public void setViolatedRouteNumber(String violatedRouteNumber) {

this.violatedRouteNumber = violatedRouteNumber;

}

public String getViolationDate() {

return violationDate;

}

public void setViolationDate(String violationDate) {

this.violationDate = violationDate;

}

public String getViolatedDriver() {

return violatedDriver;

}

public void setViolatedDriver(String violatedDriver) {

this.violatedDriver = violatedDriver;

}

public String getViolationDescription() {

return violationDescription;

}

public void setViolationDescription(String violationDescription) {

this.violationDescription = violationDescription;

}

}

MainFormController.java

package Controllers;

import Classes.Bus;

import Classes.Driver;

import Classes.Route;

import Classes.Violation;

import Kurs.DriverClasses;

import Exceptions.FileCannotBeReadException;

import Exceptions.FileNameIsntDefinedException;

import Exceptions.ValueDoesntMatchPattern;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.collections.transformation.FilteredList;

import javafx.collections.transformation.SortedList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.\*;

import javafx.scene.control.TextField;

import javafx.scene.control.cell.ComboBoxTableCell;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.control.cell.TextFieldTableCell;

import javafx.scene.input.KeyEvent;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import javafx.stage.Window;

import org.w3c.dom.\*;

import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.awt.\*;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.time.LocalDate;

import java.util.LinkedList;

import java.util.List;

import java.util.Objects;

import java.util.regex.Pattern;

import java.util.stream.IntStream;

public class MainFormController extends Window {

private ObservableList<String> driverFilterValues = FXCollections.observableArrayList("Имя", "Фамилия", "Стаж", "Класс");

private ObservableList<String> busFilterValues = FXCollections.observableArrayList("Госномер", "Номер маршрута");

private ObservableList<String> routeFilterValues = FXCollections.observableArrayList("Номер маршрута", "Время начала движения", "Время окончания движения");

private ObservableList<String> violationFilterValues = FXCollections.observableArrayList("Номер маршрута", "Дата", "Водитель", "Описание нарушения");

private ObservableList<Driver> driversList;

private ObservableList<Bus> busesList;

private ObservableList<Route> routesList;

private ObservableList<Violation> violationsList;

@FXML

private TextField addFirstName;

@FXML

private TextField addSurname;

@FXML

private ComboBox<String> addDriverExperience;

@FXML

private ComboBox<String> addDriverClass;

@FXML

private TextField addGovernmentNumber;

@FXML

private ComboBox<String> addBusRouteNumber;

@FXML

private ComboBox<String> addRouteNumber;

@FXML

private TextField addRouteStartTime;

@FXML

private TextField addRouteEndTime;

@FXML

private ComboBox<String> addViolatedRouteNumber;

@FXML

private DatePicker addViolationDate;

@FXML

private ComboBox<String> addViolatedDriver;

@FXML

private TextField addViolationDescription;

private Desktop desktop = Desktop.getDesktop();

@FXML

private TableColumn<Driver, String> driverExperience;

@FXML

private TableColumn<Driver, String> driverClass;

@FXML

private TableColumn<Driver, String> driverSurname;

@FXML

private TableColumn<Driver, String> driverName;

@FXML

private TableView<Driver> driversTable;

@FXML

private TableView<Bus> busesTable;

@FXML

private TableView<Route> routesTable;

@FXML

private TableView<Violation> violationsTable;

@FXML

private TableColumn<Route, String> routeNumber;

@FXML

private TableColumn<Route, String> routeStartTime;

@FXML

private TableColumn<Route, String> routeEndTime;

@FXML

private TableColumn<Bus, String> governmentNumber;

@FXML

private TableColumn<Bus, String> busRouteNumber;

@FXML

private TableColumn<Violation, String> violatedRouteNumber;

@FXML

private TableColumn<Violation, String> violationDate;

@FXML

private TableColumn<Violation, String> violatedDriver;

@FXML

private TableColumn<Violation, String> violationDescription;

@FXML

private Button openFile;

@FXML

private TextField violationFilterField;

@FXML

private Button deleteViolation;

@FXML

private Button deleteDriver;

@FXML

private Button deleteBus;

@FXML

private TextField busFilterField;

@FXML

private Button createBus;

@FXML

private TextField driverFilterField;

@FXML

private Button createDriver;

@FXML

private ComboBox<String> routeFilter;

@FXML

private Button closeFile;

@FXML

private TextField routeFilterField;

@FXML

private ComboBox<String> violationFilter;

@FXML

private Button saveFile;

@FXML

private ComboBox<String> busFilter;

@FXML

private ComboBox<String> driverFilter;

@FXML

private Button createRoute;

@FXML

private Button deleteRoute;

@FXML

private Button createViolation;

@FXML

private Button saveEditedDriver;

@FXML

private Button editDriver;

@FXML

private Button saveEditedBus;

@FXML

private Button editBus;

@FXML

private Button saveEditedRoute;

@FXML

private Button editRoute;

@FXML

private Button saveEditedViolation;

@FXML

private Button editViolation;

private Stage stage;

private File file;

@FXML

void initialize() {

saveFile.setDisable(true);

closeFile.setDisable(true);

initDriversTable();

initBusesTable();

initRoutesTable();

initViolationsTable();

driverFilter.getItems().addAll(driverFilterValues);

busFilter.getItems().addAll(busFilterValues);

routeFilter.getItems().addAll(routeFilterValues);

violationFilter.getItems().addAll(violationFilterValues);

}

private void initDriversTable() {

driversTable.setEditable(false);

driverName.setCellValueFactory(new PropertyValueFactory<Driver, String>("driverName"));

driverName.setCellFactory(TextFieldTableCell.forTableColumn());

driverSurname.setCellValueFactory(new PropertyValueFactory<Driver, String>("driverSurname"));

driverSurname.setCellFactory(TextFieldTableCell.forTableColumn());

driverExperience.setCellValueFactory(new PropertyValueFactory<Driver, String>("driverExperience"));

driverExperience.setCellFactory((TableColumn<Driver, String> p) -> {

ComboBoxTableCell cell = new ComboBoxTableCell(IntStream.rangeClosed(0, 40).boxed().toArray());

cell.setComboBoxEditable(true);

return cell;

});

driverClass.setCellValueFactory(new PropertyValueFactory<Driver, String>("driverClass"));

driverClass.setCellFactory((TableColumn<Driver, String> p) -> {

ComboBoxTableCell cell = new ComboBoxTableCell(DriverClasses.values());

cell.setComboBoxEditable(true);

return cell;

});

addDriverClass.setItems(FXCollections.observableArrayList("D", "BD", "CD", "BCD"));

addDriverExperience.setItems(FXCollections.observableArrayList("0", "1", "2", "3", "4", "5", "6", "7",

"8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24",

"25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40"));

createDriver.setDisable(true);

deleteDriver.setDisable(true);

addDriverClass.setDisable(true);

addDriverExperience.setDisable(true);

addFirstName.setDisable(true);

addSurname.setDisable(true);

driverFilter.setDisable(true);

driverFilterField.setDisable(true);

editDriver.setDisable(true);

saveEditedDriver.setDisable(true);

}

private void initBusesTable() {

busesTable.setEditable(false);

governmentNumber.setCellValueFactory(new PropertyValueFactory<Bus, String>("governmentNumber"));

governmentNumber.setEditable(true);

busRouteNumber.setCellValueFactory(new PropertyValueFactory<Bus, String>("busRouteNumber"));

busRouteNumber.setCellFactory((TableColumn<Bus, String> p) -> {

ComboBoxTableCell cell = new ComboBoxTableCell(Route.possibleBusNumbers);

cell.setComboBoxEditable(true);

return cell;

});

addBusRouteNumber.getItems().setAll(Route.possibleBusNumbers);

createBus.setDisable(true);

deleteBus.setDisable(true);

addBusRouteNumber.setDisable(true);

addGovernmentNumber.setDisable(true);

busFilter.setDisable(true);

busFilterField.setDisable(true);

editBus.setDisable(true);

saveEditedBus.setDisable(true);

}

private void initRoutesTable() {

routesTable.setEditable(false);

routeNumber.setCellValueFactory(new PropertyValueFactory<Route, String>("routeNumber"));

routeNumber.setCellFactory((TableColumn<Route, String> p) -> {

ComboBoxTableCell cell = new ComboBoxTableCell(Route.possibleBusNumbers);

cell.setComboBoxEditable(true);

return cell;

});

routeStartTime.setCellValueFactory(new PropertyValueFactory<Route, String>("routeStartTime"));

routeStartTime.setEditable(true);

routeEndTime.setCellValueFactory(new PropertyValueFactory<Route, String>("routeEndTime"));

routeEndTime.setEditable(true);

addRouteNumber.getItems().setAll(Route.possibleBusNumbers);

createRoute.setDisable(true);

deleteRoute.setDisable(true);

addRouteNumber.setDisable(true);

addRouteStartTime.setDisable(true);

addRouteEndTime.setDisable(true);

routeFilter.setDisable(true);

routeFilterField.setDisable(true);

editRoute.setDisable(true);

saveEditedRoute.setDisable(true);

}

private void initViolationsTable() {

violationsTable.setEditable(false);

violatedRouteNumber.setCellValueFactory(new PropertyValueFactory<Violation, String>("violatedRouteNumber"));

violatedRouteNumber.setCellFactory((TableColumn<Violation, String> p) -> {

ComboBoxTableCell cell = new ComboBoxTableCell(Route.possibleBusNumbers);

cell.setComboBoxEditable(true);

return cell;

});

violationDate.setCellValueFactory(new PropertyValueFactory<Violation, String>("violationDate"));

violationDate.setCellFactory(TextFieldTableCell.forTableColumn());

violatedDriver.setCellValueFactory(new PropertyValueFactory<Violation, String>("violatedDriver"));

violatedDriver.setCellFactory((TableColumn<Violation, String> p) -> {

ComboBoxTableCell cell = new ComboBoxTableCell(availableDrivers());

cell.setComboBoxEditable(true);

return cell;

});

violationDescription.setCellValueFactory(new PropertyValueFactory<Violation, String>("violationDescription"));

violationDescription.setCellFactory(TextFieldTableCell.forTableColumn());

addViolatedRouteNumber.getItems().setAll(Route.possibleBusNumbers);

createViolation.setDisable(true);

deleteViolation.setDisable(true);

addViolatedRouteNumber.setDisable(true);

addViolationDate.setDisable(true);

addViolatedDriver.setDisable(true);

addViolationDescription.setDisable(true);

violationFilter.setDisable(true);

violationFilterField.setDisable(true);

editViolation.setDisable(true);

saveEditedViolation.setDisable(true);

}

private String[] availableDrivers() {

String[] availableDrivers = new String[driversList.size()];

for (int i = 0; i < availableDrivers.length; i++) {

availableDrivers[i] = driversList.get(i).getDriverName() + " " + driversList.get(i).getDriverSurname();

}

return availableDrivers;

}

@FXML

void openFile(ActionEvent event) throws FileCannotBeReadException {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Open Resource File");

FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("XML files (\*.xml)", "\*.xml");

fileChooser.getExtensionFilters().add(extFilter);

file = fileChooser.showOpenDialog(stage);

driversList = FXCollections.observableArrayList(getDriverListFromXml(file));

driversTable.setItems(driversList);

busesList = FXCollections.observableArrayList(getBusListFromXml(file));

busesTable.setItems(busesList);

routesList = FXCollections.observableArrayList(getRouteListFromXml(file));

routesTable.setItems(routesList);

violationsList = FXCollections.observableArrayList(getViolationListFromXml(file));

violationsTable.setItems(violationsList);

driversTable.refresh();

busesTable.refresh();

routesTable.refresh();

violationsTable.refresh();

saveFile.setDisable(false);

closeFile.setDisable(false);

createDriver.setDisable(false);

createBus.setDisable(false);

createRoute.setDisable(false);

createViolation.setDisable(false);

deleteDriver.setDisable(false);

deleteBus.setDisable(false);

deleteRoute.setDisable(false);

deleteViolation.setDisable(false);

addDriverClass.setDisable(false);

addDriverExperience.setDisable(false);

addFirstName.setDisable(false);

addSurname.setDisable(false);

addBusRouteNumber.setDisable(false);

addGovernmentNumber.setDisable(false);

addRouteNumber.setDisable(false);

addRouteStartTime.setDisable(false);

addRouteEndTime.setDisable(false);

addViolatedRouteNumber.setDisable(false);

addViolationDate.setDisable(false);

addViolatedDriver.setDisable(false);

addViolationDescription.setDisable(false);

driverFilter.setDisable(false);

driverFilterField.setDisable(false);

busFilter.setDisable(false);

busFilterField.setDisable(false);

routeFilter.setDisable(false);

routeFilterField.setDisable(false);

violationFilter.setDisable(false);

violationFilterField.setDisable(false);

editDriver.setDisable(false);

editBus.setDisable(false);

editRoute.setDisable(false);

editViolation.setDisable(false);

addViolatedDriver.getItems().setAll(availableDrivers());

}

private List<Driver> getDriverListFromXml(File file) throws FileCannotBeReadException {

Document doc = null;

List<Driver> driverList = new LinkedList<>();

if (file != null) {

try {

DocumentBuilder dBuilder =

DocumentBuilderFactory.newInstance().newDocumentBuilder();

doc = dBuilder.parse(file);

doc.getDocumentElement().normalize();

NodeList items = doc.getElementsByTagName("items").item(0).getChildNodes();

NodeList driverNodeList = items.item(0).getChildNodes();

for (int i = 0; i < driverNodeList.getLength(); i++) {

Node elem = driverNodeList.item(i);

NamedNodeMap attrs = elem.getAttributes();

String name = attrs.getNamedItem("name").getNodeValue();

String surname = attrs.getNamedItem("surname").getNodeValue();

String driverClass = attrs.getNamedItem("class").getNodeValue();

String experience = attrs.getNamedItem("experience").getNodeValue();

Driver driver = new Driver(name, surname, experience, driverClass);

driverList.add(driver);

}

} catch (ParserConfigurationException | IOException | SAXException e) {

throw new FileCannotBeReadException("Содержимое файла не соответствует корректной схеме");

}

}

return driverList;

}

private List<Bus> getBusListFromXml(File file) throws FileCannotBeReadException {

Document doc = null;

List<Bus> busList = new LinkedList<>();

if (file != null) {

try {

DocumentBuilder dBuilder =

DocumentBuilderFactory.newInstance().newDocumentBuilder();

doc = dBuilder.parse(file);

doc.getDocumentElement().normalize();

NodeList items = doc.getElementsByTagName("items").item(0).getChildNodes();

NodeList busNodeList = items.item(1).getChildNodes();

for (int i = 0; i < busNodeList.getLength(); i++) {

Node elem = busNodeList.item(i);

NamedNodeMap attrs = elem.getAttributes();

String governmentNumber = attrs.getNamedItem("governmentNumber").getNodeValue();

String busRouteNumber = attrs.getNamedItem("busRouteNumber").getNodeValue();

Bus bus = new Bus(governmentNumber, busRouteNumber);

busList.add(bus);

}

} catch (ParserConfigurationException | IOException | SAXException e) {

throw new FileCannotBeReadException("Содержимое файла не соответствует корректной схеме");

}

}

return busList;

}

private List<Route> getRouteListFromXml(File file) throws FileCannotBeReadException {

Document doc = null;

List<Route> routeList = new LinkedList<>();

if (file != null) {

try {

DocumentBuilder dBuilder =

DocumentBuilderFactory.newInstance().newDocumentBuilder();

doc = dBuilder.parse(file);

doc.getDocumentElement().normalize();

NodeList items = doc.getElementsByTagName("items").item(0).getChildNodes();

NodeList routeNodeList = items.item(2).getChildNodes();

for (int i = 0; i < routeNodeList.getLength(); i++) {

Node elem = routeNodeList.item(i);

NamedNodeMap attrs = elem.getAttributes();

String routeNumber = attrs.getNamedItem("busRouteNumber").getNodeValue();

String routeStartTime = attrs.getNamedItem("startTime").getNodeValue();

String routeEndTime = attrs.getNamedItem("endTime").getNodeValue();

Route route = new Route(routeNumber, routeStartTime, routeEndTime);

routeList.add(route);

}

} catch (ParserConfigurationException | IOException | SAXException e) {

throw new FileCannotBeReadException("Содержимое файла не соответствует корректной схеме");

}

}

return routeList;

}

private List<Violation> getViolationListFromXml(File file) throws FileCannotBeReadException {

Document doc = null;

List<Violation> violationList = new LinkedList<>();

if (file != null) {

try {

DocumentBuilder dBuilder =

DocumentBuilderFactory.newInstance().newDocumentBuilder();

doc = dBuilder.parse(file);

doc.getDocumentElement().normalize();

NodeList items = doc.getElementsByTagName("items").item(0).getChildNodes();

NodeList violationNodeList = items.item(3).getChildNodes();

for (int i = 0; i < violationNodeList.getLength(); i++) {

Node elem = violationNodeList.item(i);

NamedNodeMap attrs = elem.getAttributes();

String violatedRouteNumber = attrs.getNamedItem("violatedRouteNumber").getNodeValue();

String violationDate = attrs.getNamedItem("violationDate").getNodeValue();

String violatedDriver = attrs.getNamedItem("violatedDriver").getNodeValue();

String violationDescription = attrs.getNamedItem("violationDescription").getNodeValue();

Violation violation = new Violation(violatedRouteNumber, violationDate, violatedDriver, violationDescription);

violationList.add(violation);

}

} catch (ParserConfigurationException | IOException | SAXException e) {

throw new FileCannotBeReadException("Содержимое файла не соответствует корректной схеме");

}

}

return violationList;

}

@FXML

void saveFile(ActionEvent event) throws ValueDoesntMatchPattern {

FileChooser fileChooser = new FileChooser();

fileChooser.setTitle("Save Resource File");

FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("XML files (\*.xml)", "\*.xml");

fileChooser.getExtensionFilters().add(extFilter);

fileChooser.setInitialDirectory(file.getParentFile());

file = fileChooser.showSaveDialog(stage);

if (file == null) {

throw new FileNameIsntDefinedException("Не указано имя файла");

} else {

try {

DocumentBuilder builder =

DocumentBuilderFactory.newInstance().newDocumentBuilder();

Document doc = builder.newDocument();

Node itemsList = doc.createElement("items");

doc.appendChild(itemsList);

Node driversList = doc.createElement("drivers");

itemsList.appendChild(driversList);

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

Element driver = doc.createElement("driver");

driversList.appendChild(driver);

driver.setAttribute("name", driversTable.getItems().get(i).getDriverName());

driver.setAttribute("surname", driversTable.getItems().get(i).getDriverSurname());

driver.setAttribute("class", driversTable.getItems().get(i).getDriverClass());

driver.setAttribute("experience", driversTable.getItems().get(i).getDriverExperience());

}

Node busesList = doc.createElement("buses");

itemsList.appendChild(busesList);

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

Element bus = doc.createElement("bus");

busesList.appendChild(bus);

String patternStr = "\\w\\d{3}\\w{2}";

Pattern pattern = Pattern.compile(patternStr);

if (!pattern.matcher(busesTable.getItems().get(i).getGovernmentNumber()).find()) {

throw new ValueDoesntMatchPattern("Номер автобуса не соответствует схеме 'бЦЦЦбб'");

} else {

bus.setAttribute("governmentNumber", busesTable.getItems().get(i).getGovernmentNumber());

bus.setAttribute("busRouteNumber", busesTable.getItems().get(i).getBusRouteNumber());

}

}

Node routesList = doc.createElement("routes");

itemsList.appendChild(routesList);

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

Element route = doc.createElement("route");

routesList.appendChild(route);

route.setAttribute("busRouteNumber", routesTable.getItems().get(i).getRouteNumber());

String patternStr = "\\d{2}:\\d{2}";

Pattern pattern = Pattern.compile(patternStr);

if (!pattern.matcher(routesTable.getItems().get(i).getRouteStartTime()).find()) {

throw new ValueDoesntMatchPattern("Время старта маршрута не соответствует схеме 'чч:мм'");

} else {

route.setAttribute("startTime", routesTable.getItems().get(i).getRouteStartTime());

}

if (!pattern.matcher(routesTable.getItems().get(i).getRouteEndTime()).find()) {

throw new ValueDoesntMatchPattern("Время старта маршрута не соответствует схеме 'чч:мм'");

} else {

route.setAttribute("endTime", routesTable.getItems().get(i).getRouteEndTime());

}

}

Node violationsList = doc.createElement("violations");

itemsList.appendChild(violationsList);

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

Element violation = doc.createElement("violation");

violationsList.appendChild(violation);

violation.setAttribute("violatedRouteNumber", violationsTable.getItems().get(i).getViolatedRouteNumber());

violation.setAttribute("violationDate", violationsTable.getItems().get(i).getViolationDate());

violation.setAttribute("violatedDriver", violationsTable.getItems().get(i).getViolatedDriver());

violation.setAttribute("violationDescription", violationsTable.getItems().get(i).getViolationDescription());

}

try {

Transformer trans = TransformerFactory.newInstance().newTransformer();

java.io.FileWriter fw = new FileWriter(new File(file.getParent(), file.getName()));

trans.transform(new DOMSource(doc), new StreamResult(fw));

fw.close();

} catch (TransformerException | IOException eq) {

eq.printStackTrace();

}

} catch (ParserConfigurationException e1) {

e1.printStackTrace();

}

}

}

@FXML

void filterDrivers(KeyEvent event) {

ObservableList<Driver> masterData = driversTable.getItems();

FilteredList<Driver> filteredData = new FilteredList<>(masterData, p -> true);

driverFilterField.textProperty().addListener((observable, oldValue, newValue) -> filteredData.setPredicate(driver -> {

if (newValue == null || newValue.isEmpty()) {

return true;

}

String lowerCaseFilter = newValue.toLowerCase();

if (Objects.equals(driverFilter.getSelectionModel().getSelectedItem(), "Имя") &

driver.getDriverName().toLowerCase().contains(lowerCaseFilter)) {

return true;

} else if (Objects.equals(driverFilter.getSelectionModel().getSelectedItem(), "Фамилия") &

driver.getDriverSurname().toLowerCase().contains(lowerCaseFilter)) {

return true;

} else if (Objects.equals(driverFilter.getSelectionModel().getSelectedItem(), "Стаж") &

driver.getDriverExperience().toLowerCase().contains(lowerCaseFilter)) {

return true;

} else if (Objects.equals(driverFilter.getSelectionModel().getSelectedItem(), "Класс") &

driver.getDriverClass().toLowerCase().contains(lowerCaseFilter)) {

return true;

}

return false;

}

));

SortedList<Driver> sortedData = new SortedList<>(filteredData);

sortedData.comparatorProperty().bind(driversTable.comparatorProperty());

driversTable.setItems(sortedData);

}

@FXML

void createDriver(ActionEvent event) {

driversTable.getItems().add(new Driver(addFirstName.getText(), addSurname.getText(),

addDriverExperience.getSelectionModel().getSelectedItem(),

addDriverClass.getSelectionModel().getSelectedItem()));

addDriverClass.getItems().clear();

addDriverExperience.getItems().clear();

addFirstName.clear();

addSurname.clear();

}

@FXML

void deleteDriver(ActionEvent event) {

Driver driver = driversTable.getSelectionModel().getSelectedItem();

driverFilterField.clear();

driversTable.setItems(driversList);

driversTable.getItems().remove(driver);

}

@FXML

void filterBuses(KeyEvent event) {

ObservableList<Bus> masterData = busesTable.getItems();

FilteredList<Bus> filteredData = new FilteredList<>(masterData, p -> true);

busFilterField.textProperty().addListener((observable, oldValue, newValue) -> filteredData.setPredicate(bus -> {

if (newValue == null || newValue.isEmpty()) {

return true;

}

String lowerCaseFilter = newValue.toLowerCase();

if (Objects.equals(busFilter.getSelectionModel().getSelectedItem(), "Госномер") &

bus.getGovernmentNumber().toLowerCase().contains(lowerCaseFilter)) {

return true;

} else if (Objects.equals(driverFilter.getSelectionModel().getSelectedItem(), "Номер Маршрута") &

bus.getBusRouteNumber().toLowerCase().contains(lowerCaseFilter)) {

return true;

}

return false;

}

));

SortedList<Bus> sortedData = new SortedList<>(filteredData);

sortedData.comparatorProperty().bind(busesTable.comparatorProperty());

busesTable.setItems(sortedData);

}

@FXML

void createBus(ActionEvent event) {

busesTable.getItems().add(new Bus(addGovernmentNumber.getText(), addBusRouteNumber.getValue()));

addGovernmentNumber.clear();

addBusRouteNumber.getItems().clear();

}

@FXML

void deleteBus(ActionEvent event) {

Bus bus = busesTable.getSelectionModel().getSelectedItem();

busFilterField.clear();

busesTable.setItems(busesList);

busesTable.getItems().remove(bus);

}

@FXML

void filterRoutes(KeyEvent event) {

ObservableList<Route> masterData = routesTable.getItems();

FilteredList<Route> filteredData = new FilteredList<>(masterData, p -> true);

routeFilterField.textProperty().addListener((observable, oldValue, newValue) -> filteredData.setPredicate(route -> {

if (newValue == null || newValue.isEmpty()) {

return true;

}

String lowerCaseFilter = newValue.toLowerCase();

if (Objects.equals(routeFilter.getSelectionModel().getSelectedItem(), "Номер маршрута") &

route.getRouteNumber().toLowerCase().contains(lowerCaseFilter)) {

return true;

}

if (Objects.equals(routeFilter.getSelectionModel().getSelectedItem(), "Время начала движения") &

route.getRouteStartTime().toLowerCase().contains(lowerCaseFilter)) {

return true;

}

return Objects.equals(routeFilter.getSelectionModel().getSelectedItem(), "Время окончания движения") &

route.getRouteEndTime().toLowerCase().contains(lowerCaseFilter);

}

));

SortedList<Route> sortedData = new SortedList<>(filteredData);

sortedData.comparatorProperty().bind(routesTable.comparatorProperty());

routesTable.setItems(sortedData);

}

@FXML

void createRoute(ActionEvent event) throws ValueDoesntMatchPattern {

String patternStr = "\\d{2}:\\d{2}";

Pattern pattern = Pattern.compile(patternStr);

if (!pattern.matcher(addRouteStartTime.getText()).find()) {

throw new ValueDoesntMatchPattern("Время старта маршрута не соответствует схеме 'чч:мм'");

}

if (!pattern.matcher(addRouteEndTime.getText()).find()) {

throw new ValueDoesntMatchPattern("Время конца маршрута не соответствует схеме 'чч:мм'");

} else {

routesTable.getItems().add(new Route(addRouteNumber.getValue(), addRouteStartTime.getText(), addRouteEndTime.getText()));

addRouteNumber.getItems().clear();

addRouteStartTime.clear();

addRouteEndTime.clear();

}

}

@FXML

void deleteRoute(ActionEvent event) {

Route route = routesTable.getSelectionModel().getSelectedItem();

routeFilterField.clear();

routesTable.setItems(routesList);

routesTable.getItems().remove(route);

}

@FXML

void filterViolations(KeyEvent event) {

ObservableList<Violation> masterData = violationsTable.getItems();

FilteredList<Violation> filteredData = new FilteredList<>(masterData, p -> true);

violationFilterField.textProperty().addListener((observable, oldValue, newValue) -> filteredData.setPredicate(violation -> {

if (newValue == null || newValue.isEmpty()) {

return true;

}

String lowerCaseFilter = newValue.toLowerCase();

if (Objects.equals(violationFilter.getSelectionModel().getSelectedItem(), "Номер маршрута") &

violation.getViolatedRouteNumber().toLowerCase().contains(lowerCaseFilter)) {

return true;

}

if (Objects.equals(violationFilter.getSelectionModel().getSelectedItem(), "Дата") &

violation.getViolationDate().toLowerCase().contains(lowerCaseFilter)) {

return true;

}

if (Objects.equals(violationFilter.getSelectionModel().getSelectedItem(), "Водитель") &

violation.getViolatedDriver().toLowerCase().contains(lowerCaseFilter)) {

return true;

}

return Objects.equals(violationFilter.getSelectionModel().getSelectedItem(), "Описание нарушения") &

violation.getViolationDescription().toLowerCase().contains(lowerCaseFilter);

}

));

SortedList<Violation> sortedData = new SortedList<>(filteredData);

sortedData.comparatorProperty().bind(violationsTable.comparatorProperty());

violationsTable.setItems(sortedData);

}

@FXML

void createViolation(ActionEvent event) {

violationsTable.getItems().add(new Violation(addViolatedRouteNumber.getValue(), addViolationDate.getValue().toString(),

addViolatedDriver.getValue(), addViolationDescription.getText()));

addViolatedRouteNumber.getItems().clear();

addViolationDate.getEditor().clear();

addViolatedDriver.getItems().clear();

addViolationDescription.clear();

}

@FXML

void deleteViolation(ActionEvent event) {

Violation violation = violationsTable.getSelectionModel().getSelectedItem();

violationFilterField.clear();

violationsTable.setItems(violationsList);

violationsTable.getItems().remove(violation);

}

@FXML

void editDriver(ActionEvent event) {

addFirstName.appendText(driversTable.getSelectionModel().getSelectedItem().getDriverName());

addSurname.appendText(driversTable.getSelectionModel().getSelectedItem().getDriverSurname());

addDriverExperience.setValue(driversTable.getSelectionModel().getSelectedItem().getDriverExperience());

addDriverClass.setValue(driversTable.getSelectionModel().getSelectedItem().getDriverClass());

editDriver.setDisable(true);

saveEditedDriver.setDisable(false);

}

@FXML

void saveEditedDriver(ActionEvent event) {

driversTable.getSelectionModel().getSelectedItem().setDriverName(addFirstName.getText());

driversTable.getSelectionModel().getSelectedItem().setDriverSurname(addSurname.getText());

driversTable.getSelectionModel().getSelectedItem().setDriverExperience(addDriverExperience.getValue());

driversTable.getSelectionModel().getSelectedItem().setDriverClass(addDriverClass.getValue());

driversTable.refresh();

addFirstName.clear();

addSurname.clear();

addDriverExperience.getItems().clear();

addDriverClass.getItems().clear();

saveEditedDriver.setDisable(true);

editDriver.setDisable(false);

}

@FXML

void editBus(ActionEvent event) {

addGovernmentNumber.appendText(busesTable.getSelectionModel().getSelectedItem().getGovernmentNumber());

addBusRouteNumber.setValue(busesTable.getSelectionModel().getSelectedItem().getBusRouteNumber());

editBus.setDisable(true);

saveEditedBus.setDisable(false);

}

@FXML

void saveEditedBus(ActionEvent event) {

busesTable.getSelectionModel().getSelectedItem().setGovernmentNumber(addGovernmentNumber.getText());

busesTable.getSelectionModel().getSelectedItem().setBusRouteNumber(addBusRouteNumber.getValue());

busesTable.refresh();

addGovernmentNumber.clear();

addBusRouteNumber.getItems().clear();

saveEditedBus.setDisable(true);

editBus.setDisable(false);

}

@FXML

void editRoute(ActionEvent event) {

addRouteNumber.setValue(routesTable.getSelectionModel().getSelectedItem().getRouteNumber());

addRouteStartTime.appendText(routesTable.getSelectionModel().getSelectedItem().getRouteStartTime());

addRouteEndTime.appendText(routesTable.getSelectionModel().getSelectedItem().getRouteEndTime());

editRoute.setDisable(true);

saveEditedRoute.setDisable(false);

}

@FXML

void saveEditedRoute(ActionEvent event) {

routesTable.getSelectionModel().getSelectedItem().setRouteNumber(addRouteNumber.getValue());

routesTable.getSelectionModel().getSelectedItem().setRouteStartTime(addRouteStartTime.getText());

routesTable.getSelectionModel().getSelectedItem().setRouteEndTime(addRouteEndTime.getText());

addRouteNumber.getItems().clear();

addRouteStartTime.clear();

addRouteEndTime.clear();

routesTable.refresh();

editRoute.setDisable(false);

saveEditedRoute.setDisable(true);

}

@FXML

void editViolation(ActionEvent event) {

addViolatedRouteNumber.setValue(violationsTable.getSelectionModel().getSelectedItem().getViolatedRouteNumber());

addViolationDate.setValue(LocalDate.parse(violationsTable.getSelectionModel().getSelectedItem().getViolationDate()));

addViolatedDriver.setValue(violationsTable.getSelectionModel().getSelectedItem().getViolatedDriver());

addViolationDescription.appendText(violationsTable.getSelectionModel().getSelectedItem().getViolationDescription());

editViolation.setDisable(true);

saveEditedViolation.setDisable(false);

}

@FXML

void saveEditedViolation(ActionEvent event) {

violationsTable.getSelectionModel().getSelectedItem().setViolatedRouteNumber(addViolatedRouteNumber.getValue());

violationsTable.getSelectionModel().getSelectedItem().setViolationDate(addViolationDate.getValue().toString());

violationsTable.getSelectionModel().getSelectedItem().setViolatedDriver(addViolatedDriver.getValue());

violationsTable.getSelectionModel().getSelectedItem().setViolationDescription(addViolationDescription.getText());

addViolatedRouteNumber.getItems().clear();

addViolationDate.getEditor().clear();

addViolatedDriver.getItems().clear();

addViolationDescription.clear();

violationsTable.refresh();

editViolation.setDisable(false);

saveEditedViolation.setDisable(true);

}

@FXML

void closeFile(ActionEvent event) {

driverFilterField.clear();

driversTable.setItems(driversList);

busFilterField.clear();

busesTable.setItems(busesList);

routeFilterField.clear();

routesTable.setItems(routesList);

violationFilterField.clear();

violationsTable.setItems(violationsList);

IntStream.range(0, driversTable.getItems().size()).forEach(i -> driversTable.getItems().clear());

IntStream.range(0, busesTable.getItems().size()).forEach(i -> busesTable.getItems().clear());

IntStream.range(0, routesTable.getItems().size()).forEach(i -> routesTable.getItems().clear());

IntStream.range(0, violationsTable.getItems().size()).forEach(i -> violationsTable.getItems().clear());

createDriver.setDisable(true);

createBus.setDisable(true);

createRoute.setDisable(true);

createViolation.setDisable(true);

deleteDriver.setDisable(true);

deleteBus.setDisable(true);

deleteRoute.setDisable(true);

deleteViolation.setDisable(true);

addDriverClass.setDisable(true);

addDriverExperience.setDisable(true);

addFirstName.setDisable(true);

addSurname.setDisable(true);

addBusRouteNumber.setDisable(true);

addGovernmentNumber.setDisable(true);

addRouteNumber.setDisable(true);

addRouteStartTime.setDisable(true);

addRouteEndTime.setDisable(true);

addViolatedRouteNumber.setDisable(true);

addViolationDate.setDisable(true);

addViolatedDriver.setDisable(true);

addViolationDescription.setDisable(true);

driverFilter.setDisable(true);

driverFilterField.setDisable(true);

busFilter.setDisable(true);

busFilterField.setDisable(true);

routeFilter.setDisable(true);

routeFilterField.setDisable(true);

violationFilter.setDisable(true);

violationFilterField.setDisable(true);

saveFile.setDisable(true);

closeFile.setDisable(true);

editDriver.setDisable(true);

editBus.setDisable(true);

editRoute.setDisable(true);

editViolation.setDisable(true);

}

}

FileCannotBeReadException.java

**package Exceptions;**

**import javafx.scene.control.Alert;**

**public class FileCannotBeReadException extends Exception {**

**public FileCannotBeReadException(String message) {**

**super(message);**

**Alert alert = new Alert(Alert.AlertType.ERROR);**

**alert.setTitle("Ошибка при открытии файла");**

**alert.setContentText(message);**

**alert.showAndWait();**

**}**

**}**

FileNameIsntDefinedException.java

**package Exceptions;**

**import javafx.scene.control.Alert;**

**public class FileNameIsntDefinedException extends NullPointerException {**

**public FileNameIsntDefinedException(String message) {**

**super(message);**

**Alert alert = new Alert(Alert.AlertType.ERROR);**

**alert.setTitle("Ошибка при сохранении файла");**

**alert.setContentText(message);**

**alert.showAndWait();**

**}**

**}**

ValueDoesntMatchPattern.java

**package Exceptions;**

**import javafx.scene.control.Alert;**

**public class ValueDoesntMatchPattern extends Exception {**

**public ValueDoesntMatchPattern(String message) {**

**super(message);**

**Alert alert = new Alert(Alert.AlertType.ERROR);**

**alert.setTitle("Ошибка при сохранении файла");**

**alert.setContentText(message);**

**alert.showAndWait();**

**}**

**}**

BusPark.java

**import javafx.application.Application;**

**import javafx.fxml.FXMLLoader;**

**import javafx.scene.Parent;**

**import javafx.scene.Scene;**

**import javafx.stage.Stage;**

**public class BusPark extends Application {**

**public static void main(String[] args) {**

**launch(args);**

**}**

**public void start(Stage primaryStage) throws Exception {**

**Parent root = FXMLLoader.load(getClass().getResource("mainForm.fxml"));**

**Scene scene = new Scene(root);**

**primaryStage.setTitle("База водителей");**

**primaryStage.setScene(scene);**

**primaryStage.show();**

**}**

**}**

DriverClasses.java

**public enum DriverClasses {**

**D, BD, CD, BCD**

**}**