# ЛАБОРАТОРНАЯ РАБОТА №5

# ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ РАЗРАБОТКИ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПЛАТФОРМЫ JAVAFX 2

Работу выполнил: Мовенко Константин, ИС/б-21-2-о

Цель работы

Ознакомиться с особенностями платформы JavaFX 2 и приобрести практические навыки создания насыщенных пользовательских интерфейсов Java программ.

Задачи

С использованием компонентов JavaFX 2 необходимо создать Java приложение, реализующее добавление, редактирование и удаление данных заданного по варианту типа информации T (Таблица 1). Данные отображать в виде таблицы. Реализовать поля ввода для добавления новых записей. Редактирование записей реализовать в таблице (использовать CellValueFactory). Предусмотреть возможность загрузки информации из текстового файла и сохранения в текстовый файл. Данные столбца N отображать в виде автоматически обновляющегося графика/диаграммы G (Таблица 1).

Вариант задания

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип информации T | Поле для отображения N | Тип графика/диаграммы G |
| 13 | C | 3 | PieChart |

Таблица 1 – Вариант задания

C: Автомобиль (марка, год выпуска, объём двигателя, максимальная скорость).

Ход работы

Программа была запущена. В результате запуска было открыто окно программы с незаполненной таблицей, пустой диаграммой и кнопками (Рисунок 1).

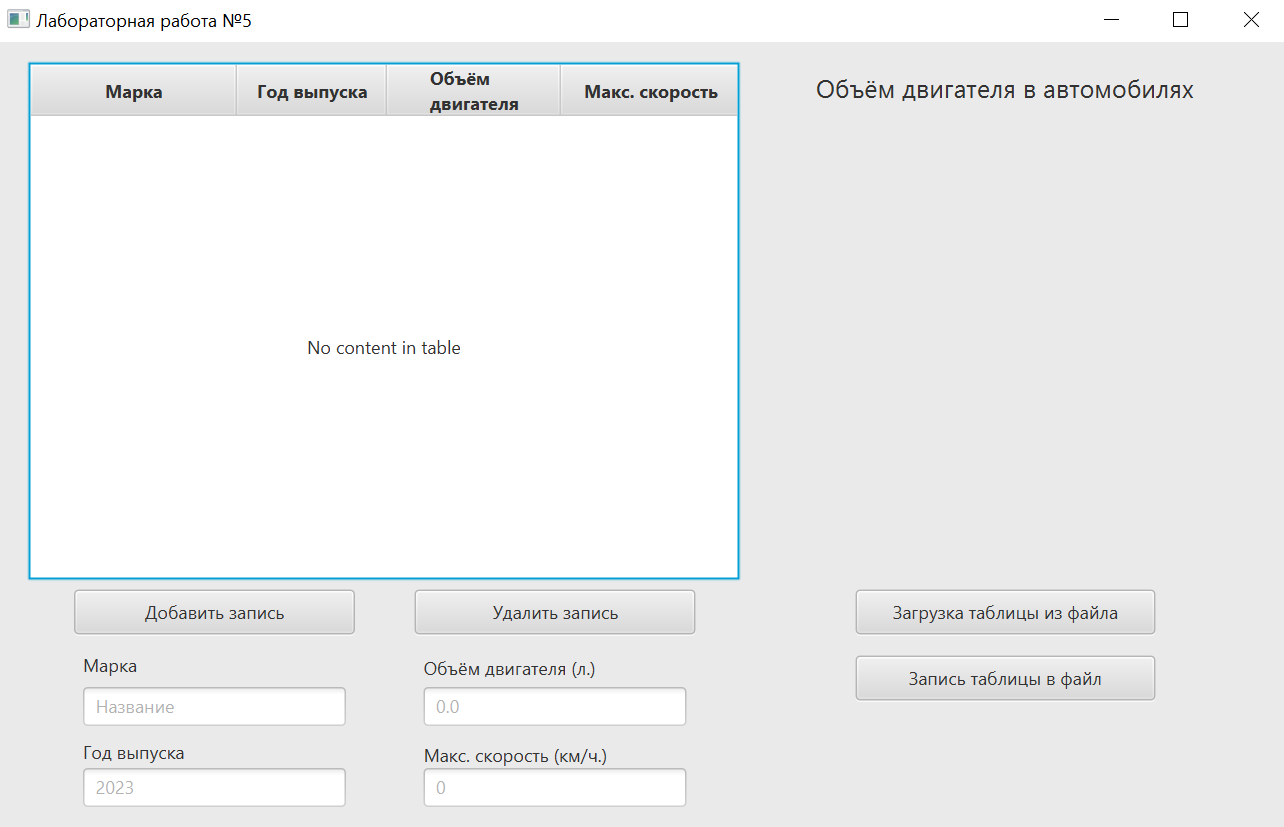


Рисунок 1 – Начало работы с программой

Был создан файл формата csv, содержащий информацию о нескольких автомобилях (Рисунок 2).

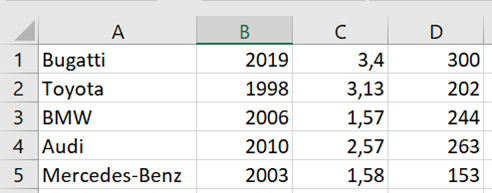


Рисунок 2 – Файл входных данных

При нажатии кнопки «Загрузка таблицы из файла» и выборе созданного файла таблица программы была заполнена значениями из файла, на основе третьего столбца была сформирована круговая диаграмма (Рисунок 3).

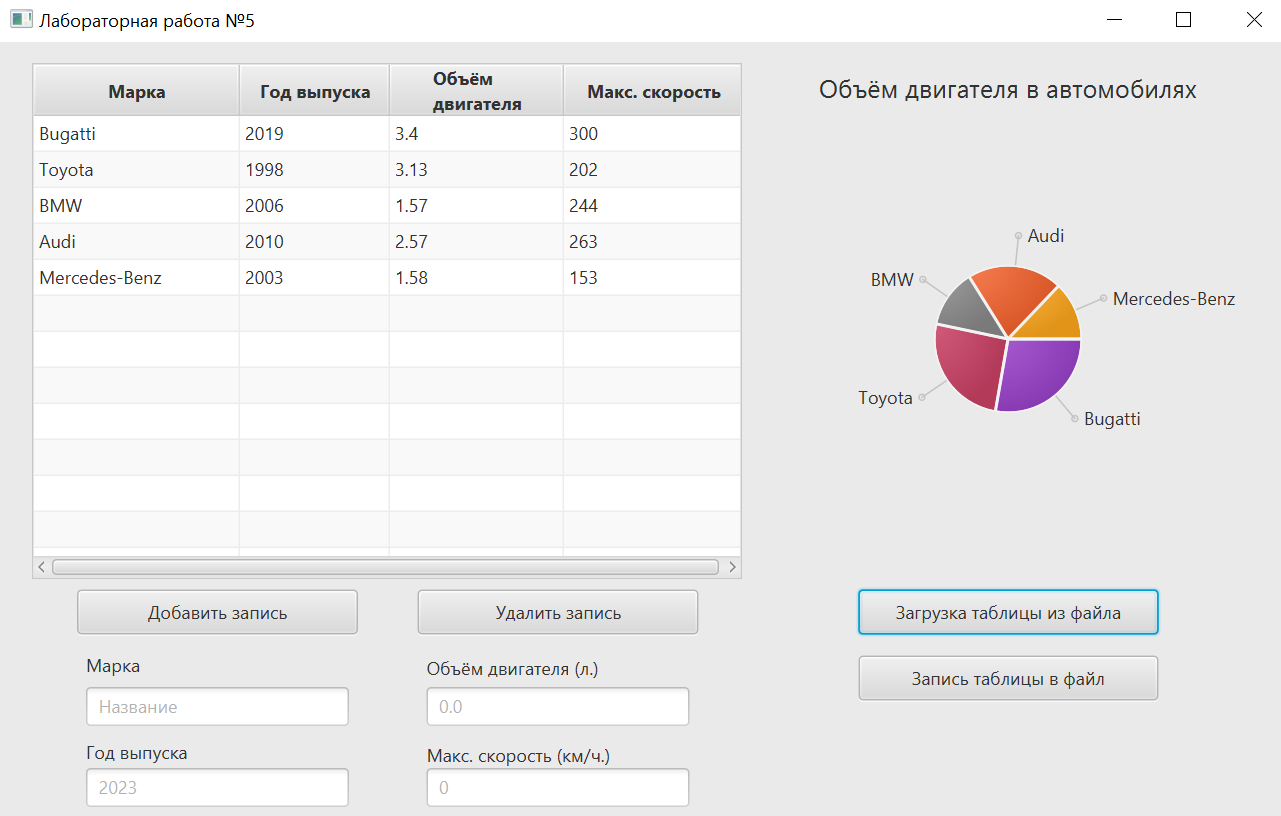


Рисунок 3 – Результат загрузки данных из файла

Над полученной таблицей был проведён ряд операций: строка с «Audi» была удалена, в строке с Toyota год был заменён на 2000, была добавлена новая строка (Рисунок 4).

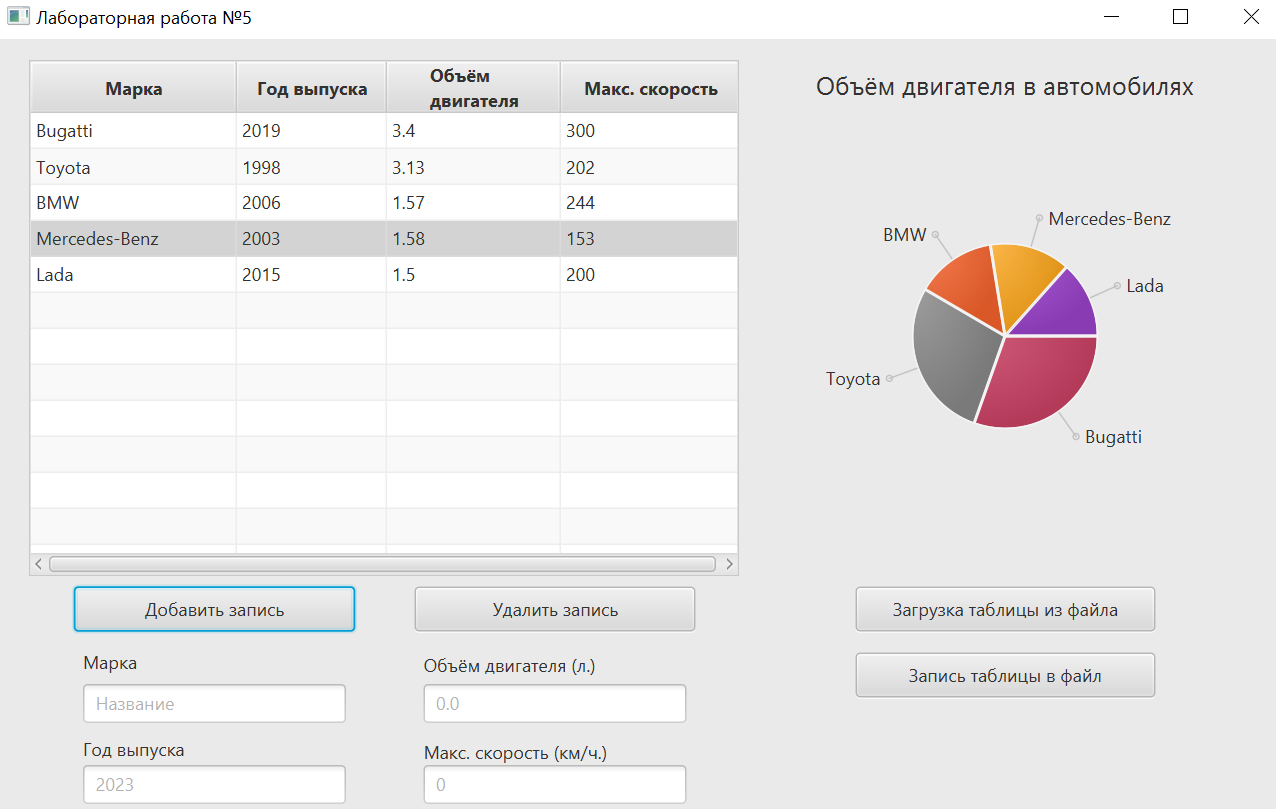


Рисунок 4 – Обновление таблицы

При нажатии кнопки «Запись таблицы в файл» данные полученной таблицы были выгружены в другой csv файл (Рисунок 5).

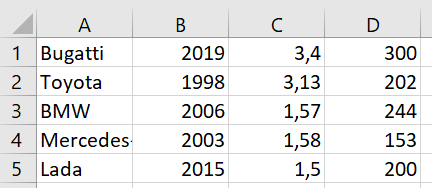


Рисунок 5 – Выгрузка данных в файл

Текст программы

Листинг 1 – Main.java

**package** application;

**import** javafx.application.Application;

**import** javafx.fxml.FXMLLoader;

**import** javafx.scene.Scene;

**import** javafx.stage.Stage;

// Точка входа в JavaFX-приложение

**public** **class** Main **extends** Application

{

// Загрузка JavaFX-приложения

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

// Создание и отображение сцены приложения

@Override **public** **void** start(Stage stage) **throws** Exception

{

stage.setTitle("Лабораторная работа №5");

//stage.setResizable(false);

stage.setScene(**new** Scene(FXMLLoader.*load*(getClass().getResource("Layout.fxml"))));

stage.show();

}

}

Листинг 2 – Controller.java

**package** application;

**import** java.io.File;

**import** java.io.BufferedWriter;

**import** java.io.FileWriter;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** javafx.collections.FXCollections;

**import** javafx.collections.ListChangeListener;

**import** javafx.collections.ObservableList;

**import** javafx.fxml.FXML;

**import** javafx.scene.chart.PieChart;

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

**import** javafx.scene.control.Button;

**import** javafx.scene.control.TableColumn;

**import** javafx.scene.control.TableView;

**import** javafx.scene.control.TextField;

**import** javafx.scene.control.cell.PropertyValueFactory;

**import** javafx.scene.control.cell.TextFieldTableCell;

**import** javafx.stage.FileChooser;

**import** javafx.stage.Stage;

**import** javafx.util.converter.DoubleStringConverter;

**import** javafx.util.converter.IntegerStringConverter;

// Класс управления компонентами приложения

**public** **class** Controller

{

// Таблица и её колонки

@FXML **private** TableView<Auto> table;

**private** ObservableList<Auto> data = FXCollections.*observableArrayList*();

@FXML **private** TableColumn<Auto, String> colBrand;

@FXML **private** TableColumn<Auto, Integer> colYear;

@FXML **private** TableColumn<Auto, Double> colEngineVolume;

@FXML **private** TableColumn<Auto, Integer> colMaxSpeed;

// Кнопки

@FXML **private** Button btnAdd, btnLoad, btnSave, btnDelete;

// Текстовые поля ввода

@FXML **private** TextField txtBrand, txtYear, txtEngineVolume, txtMaxSpeed;

// Круговая диаграмма

@FXML **private** PieChart chart;

**private** ObservableList<PieChart.Data> pcData = FXCollections.*observableArrayList*();

// Компонент выбора файла

**private** FileChooser fileChooser = **new** FileChooser();

// Начальные настройки компонентов приложения

@FXML **private** **void** initialize()

{

fileChooser.setInitialDirectory(

**new** File("C:\\Users\\kosta\\Documents\\Учёба\\СДЕЛАНО\\5 семестр\\Java"));

table.setEditable(**true**);

table.setItems(data);

data.addListener(**new** ListChangeListener<Auto>()

{

@Override **public** **void** onChanged(

javafx.collections.ListChangeListener.Change<? **extends** Auto> arg0) {

updateChart();

}

});

setColBrand();

setColYear();

setColEngineVolume();

setColMaxSpeed();

}

// Настройка колонки "Марка"

**private** **void** setColBrand()

{

colBrand.setCellValueFactory(**new** PropertyValueFactory<Auto, String>("brand"));

colBrand.setCellFactory(TextFieldTableCell.*forTableColumn*());

colBrand.setOnEditCommit(event ->

{

String brand = event.getNewValue();

**if** (brand.isEmpty())

{

showAlert("Пустое поле", "Заполните поле \"Марка\"");

}

**else** event.getRowValue().setBrand(brand);

updateChart();

});

}

// Настройка колонки "Год выпуска"

**private** **void** setColYear()

{

colYear.setCellValueFactory(**new** PropertyValueFactory<Auto, Integer>("year"));

colYear.setCellFactory(TextFieldTableCell.*forTableColumn*(**new** IntegerStringConverter()

{

@Override **public** Integer fromString(String value)

{

**try**

{

**return** Integer.*parseInt*(value);

}

**catch** (NumberFormatException e)

{

**return** **null**;

}

}

}));

colYear.setOnEditCommit(event ->

{

**try**

{

Integer year = event.getNewValue();

**if** (year == **null**) { **throw** **new** Exception(); }

event.getRowValue().setYear(year);

}

**catch** (Exception e)

{

showAlert("Ошибка формата", "Введите корректное значение года выпуска.");

}

updateChart();

});

}

// Настройка колонки "Объём двигателя"

**private** **void** setColEngineVolume()

{

colEngineVolume.setCellValueFactory(**new** PropertyValueFactory<Auto, Double>("engineVolume"));

colEngineVolume.setCellFactory(TextFieldTableCell.*forTableColumn*(**new** DoubleStringConverter()

{

@Override **public** Double fromString(String value)

{

**try**

{

**return** Double.*parseDouble*(value);

}

**catch** (NumberFormatException e)

{

**return** **null**;

}

}

}));

colEngineVolume.setOnEditCommit(event ->

{

**try**

{

Double volume = event.getNewValue();

**if** (volume == **null**) { **throw** **new** Exception(); }

event.getRowValue().setEngineVolume(volume);

}

**catch** (Exception e)

{

showAlert("Ошибка формата","Введите корректное значение объёма двигателя.");

}

updateChart();

});

}

// Настройка колонки "Макс. скорость"

**private** **void** setColMaxSpeed()

{

colMaxSpeed.setCellValueFactory(**new** PropertyValueFactory<Auto, Integer>("maxSpeed"));

colMaxSpeed.setCellFactory(TextFieldTableCell.*forTableColumn*(**new** IntegerStringConverter()

{

@Override **public** Integer fromString(String value)

{

**try**

{

**return** Integer.*parseInt*(value);

}

**catch** (NumberFormatException e)

{

**return** **null**;

}

}

}));

colMaxSpeed.setOnEditCommit(event ->

{

**try**

{

Integer maxSpeed = event.getNewValue();

**if** (maxSpeed == **null**) { **throw** **new** Exception(); }

event.getRowValue().setMaxSpeed(maxSpeed);

}

**catch**(Exception e)

{

showAlert("Ошибка формата", "Введите корректное значение скорости.");

}

updateChart();

});

}

// Вывод окна с сообщением

**private** **void** showAlert(String header, String text)

{

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

alert.setTitle("Ошибка");

alert.setHeaderText(header);

alert.setContentText(text);

alert.show();

}

// Обновление круговой диаграммы

**public** **void** updateChart()

{

**int** i;

pcData.clear();

**for** (i = 0; i < data.size(); i++)

{

pcData.add(**new** PieChart.Data(data.get(i).getBrand().toString(), data.get(i).getEngineVolume()));

}

chart.setData(pcData);

}

// Действия для кнопки "Добавить запись"

@FXML **public** **void** addEntry()

{

String brand = txtBrand.getText(),

year = txtYear.getText(),

engineVolume = txtEngineVolume.getText(),

maxSpeed = txtMaxSpeed.getText();

**int** yearValue, speedValue;

**double** volumeValue;

**if** (brand.isEmpty() || year.isEmpty() || engineVolume.isEmpty() || maxSpeed.isEmpty())

{

showAlert("Запись не создана", "Заполните все поля.");

}

**else**

{

**try** { yearValue = Integer.*valueOf*(year); }

**catch** (NumberFormatException e)

{

showAlert("Ошибка формата", "Введите корректное значение года выпуска.");

**return**;

}

**try** { volumeValue = Double.*valueOf*(engineVolume); }

**catch** (NumberFormatException e)

{

showAlert("Ошибка формата", "Введите корректное значение объёма двигателя.");

**return**;

}

**try** { speedValue = Integer.*valueOf*(maxSpeed); }

**catch** (NumberFormatException e)

{

showAlert("Ошибка формата", "Введите корректное значение скорости.");

**return**;

}

data.add(**new** Auto(brand, yearValue, volumeValue, speedValue));

txtBrand.clear();

txtYear.clear();

txtEngineVolume.clear();

txtMaxSpeed.clear();

updateChart();

}

}

// Действия для кнопки "Удалить запись"

@FXML **public** **void** deleteEntry()

{

Auto car = table.getSelectionModel().getSelectedItem();

**if** (car != **null**)

{

data.remove(car);

updateChart();

}

}

// Действия для кнопки "Загрузка таблицы"

@FXML **public** **void** loadData()

{

fileChooser.setTitle("Загрузка");

File file = fileChooser.showOpenDialog(**new** Stage());

**if** (file != **null**)

{

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(file)))

{

ObservableList<Auto> list = FXCollections.*observableArrayList*();

String line;

**while** ((line = reader.readLine()) != **null**)

{

String[] parts = line.split(";");

String brand = parts[0];

**int** year = Integer.*parseInt*(parts[1]);

**double** pageCount = Double.*parseDouble*(parts[2].replace(",", "."));

**int** speed = Integer.*parseInt*(parts[3]);

list.add(**new** Auto(brand, year, pageCount, speed));

}

data.setAll(list);

updateChart();

}

**catch** (Exception e)

{

showAlert("Ошибка чтения", "Не удалось прочесть данные.");

}

}

}

// Действия для кнопки "Запись таблицы"

@FXML **public** **void** saveData()

{

fileChooser.setTitle("Сохранение");

File file = fileChooser.showSaveDialog(**new** Stage());

**if** (file != **null**)

{

**try**

{

BufferedWriter writer = **new** BufferedWriter(**new** FileWriter(file));

ObservableList<Auto> list = data;

**for** (Auto car: list)

{ writer.write(car.toString() + '\n'); }

writer.close();

updateChart();

}

**catch** (Exception e)

{

showAlert("Ошибка сохранения", "Не удалось сохранить записи в файле.");

}

}

}

}

Листинг 3 – Auto.java

**package** application;

**import** javafx.beans.property.SimpleDoubleProperty;

**import** javafx.beans.property.SimpleIntegerProperty;

**import** javafx.beans.property.SimpleStringProperty;

//Класс для типа информации T по варианту

**public** **class** Auto

{

// Параметры

**private** SimpleStringProperty brand; // марка

**private** SimpleIntegerProperty year; // год выпуска

**private** SimpleDoubleProperty engineVolume; // объём двигателя, л.

**private** SimpleIntegerProperty maxSpeed; // макс. скорость, км/ч.

// Передача параметров через конструктор

**public** Auto(String brand, **int** year, **double** engineVolume, **int** maxSpeed)

{

**this**.brand = **new** SimpleStringProperty(brand);

**this**.year = **new** SimpleIntegerProperty(year);

**this**.engineVolume = **new** SimpleDoubleProperty(engineVolume);

**this**.maxSpeed = **new** SimpleIntegerProperty(maxSpeed);

}

// Методы получения параметров (getters)

**public** String getBrand()

{ **return** brand.get(); }

**public** **int** getYear()

{ **return** year.get(); }

**public** **double** getEngineVolume()

{ **return** engineVolume.get(); }

**public** **int** getMaxSpeed()

{ **return** maxSpeed.get(); }

// Методы изменения параметров (setters)

**public** **void** setBrand(String brand)

{ **this**.brand.set(brand); }

**public** **void** setYear(**int** year)

{ **this**.year.set(year); }

**public** **void** setEngineVolume(**double** volume)

{ **this**.engineVolume.set(volume); }

**public** **void** setMaxSpeed(**int** speed)

{ **this**.maxSpeed.set(speed); }

// Представление объекта в виде строки (для csv файла)

@Override **public** String toString()

{

**return** String.*format*("%s;%d;%f;%d",

brand.get(), year.get(), engineVolume.get(), maxSpeed.get());

}

}

Листинг 4 – Layout.fxml

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

<?import javafx.scene.chart.PieChart?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TableColumn?>

<?import javafx.scene.control.TableView?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane id=*"pane"* minHeight=*"-Infinity"* minWidth=*"-Infinity"* prefHeight=*"527.0"* prefWidth=*"860.0"* style=*"-fx-background-color: #EAEAEA;"* xmlns=*"http://javafx.com/javafx/8.0.171"* xmlns:fx=*"http://javafx.com/fxml/1"* fx:controller=*"application.Controller"*>

<children>

<TableView id=*"table"* fx:id=*"table"* layoutX=*"22.0"* layoutY=*"14.0"* prefHeight=*"344.0"* prefWidth=*"473.0"*>

<columns>

<TableColumn id=*"colBrand"* fx:id=*"colBrand"* prefWidth=*"137.33331298828125"* text=*"Марка"* />

<TableColumn id=*"colYear"* fx:id=*"colYear"* prefWidth=*"99.66665649414062"* text=*"Год выпуска"* />

<TableColumn id=*"colEngineVolume"* fx:id=*"colEngineVolume"* prefWidth=*"116.0"* text=*"Объём &#10;двигателя"* />

<TableColumn id=*"colMaxSpeed"* fx:id=*"colMaxSpeed"* prefWidth=*"120.0"* text=*"Макс. скорость"* />

</columns>

</TableView>

<Button id=*"btnAdd"* fx:id=*"btnAdd"* layoutX=*"52.0"* layoutY=*"365.0"* mnemonicParsing=*"false"* onAction=*"#addEntry"* prefHeight=*"30.0"* prefWidth=*"187.0"* text=*"Добавить запись"* />

<Button id=*"btnImport"* fx:id=*"btnLoad"* layoutX=*"573.0"* layoutY=*"365.0"* mnemonicParsing=*"false"* onAction=*"#loadData"* prefHeight=*"30.0"* prefWidth=*"200.0"* text=*"Загрузка таблицы из файла"* />

<Button id=*"btnExport"* fx:id=*"btnSave"* layoutX=*"573.0"* layoutY=*"409.0"* mnemonicParsing=*"false"* onAction=*"#saveData"* prefHeight=*"30.0"* prefWidth=*"200.0"* text=*"Запись таблицы в файл"* />

<TextField id=*"txtBrand"* fx:id=*"txtBrand"* layoutX=*"58.0"* layoutY=*"430.0"* prefHeight=*"26.0"* prefWidth=*"175.0"* promptText=*"Название"* />

<TextField id=*"txtYear"* fx:id=*"txtYear"* layoutX=*"58.0"* layoutY=*"484.0"* prefHeight=*"26.0"* prefWidth=*"175.0"* promptText=*"2023"* />

<TextField id=*"txtEngineVolume"* fx:id=*"txtEngineVolume"* layoutX=*"285.0"* layoutY=*"430.0"* prefHeight=*"26.0"* prefWidth=*"175.0"* promptText=*"0.0"* />

<TextField id=*"txtMaxSpeed"* fx:id=*"txtMaxSpeed"* layoutX=*"285.0"* layoutY=*"484.0"* prefHeight=*"26.0"* prefWidth=*"175.0"* promptText=*"0"* />

<PieChart fx:id=*"chart"* layoutX=*"502.0"* layoutY=*"14.0"* prefHeight=*"344.0"* prefWidth=*"341.0"* title=*"Объём двигателя в автомобилях"* />

<Button fx:id=*"btnDelete"* layoutX=*"279.0"* layoutY=*"365.0"* mnemonicParsing=*"false"* onAction=*"#deleteEntry"* prefHeight=*"30.0"* prefWidth=*"187.0"* text=*"Удалить запись"* />

<Label layoutX=*"58.0"* layoutY=*"407.0"* text=*"Марка"* />

<Label layoutX=*"58.0"* layoutY=*"465.0"* text=*"Год выпуска"* />

<Label layoutX=*"285.0"* layoutY=*"409.0"* text=*"Объём двигателя (л.)"* />

<Label layoutX=*"285.0"* layoutY=*"467.0"* text=*"Макс. скорость (км/ч.)"* />

</children>

</AnchorPane>

Вывод

В ходе работы было проведено ознакомление с фреймворком JavaFX и его возможностями по разработке графического пользовательского интерфейса (GUI). Были получены навыки проектирования интерфейсов программы через приложение Scene Builder, а также связывания его с программным кодом.