5 Лабораторная работа №5
Исследование возможностей разработки приложений с использованием платформы JavaFX 2

5.1. Цель работы

В ходе выполнения данной лабораторной работы необходимо ознакомиться с особенностями платформы JavaFX 2 и приобрести практические навыки создания насыщенных пользовательских интерфейсов Java-программ.

5.2. Постановка задачи

Был выдан вариант 13.

С использованием компонентов JavaFX 2 необходимо создать Java приложение реализующее добавление, редактирование и удаление данных заданного по варианту типа информации Auto. 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 3 отображать в виде автоматически обновляющегося графика/диаграммы PieChart.

5.3. Ход выполнения работы

5.3.1. Текст программы

Программа составлена на языке Java.

Листинг 1 — Содержимое основного класса App

```
package org.cory7666.lab5;

import javafx.application.Application;
```

```
public class App
{
  public static void main (String[] args)
  {
    Application.launch(MainWindow.class, args);
  }
}
```

Листинг 2 — Содержимое класса MainWindow

```
package org.cory7666.lab5;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.TabPane;
import javafx.stage.Stage;

public class MainWindow extends Application
{
  private final String layout = "layout/MainWindow.fxml";

  @Override
  public void start (Stage primaryStage) throws Exception
  {
    primaryStage.setScene(new Scene((TabPane)
FXMLLoader.load(getClass().getClassLoader().getResource(layout))));
    primaryStage.setMinHeight(400.0);
    primaryStage.setTitle("Автомобили");
    primaryStage.centerOnScreen();
    primaryStage.show();

    Logger.debug(getClass(), "Окно создано.");
  }
}
```

Листинг 3 — Содержимое класса MainWindowController

```
package org.cory7666.lab5;

import javafx.beans.property.SimpleFloatProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.scene.chart.PieChart;
import javafx.scene.control.Button;
import javafx.scene.control.SelectionMode;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;

public class MainWindowController
{
  @FXML private Button loadFromFileButton, saveToFileButton;

  @FXML private TableView<Auto> table;
  @FXML private TableColumn<Auto, String> tableColumn_Brand;
  @FXML private TableColumn<Auto, Number> tableColumn_Year;
  @FXML private TableColumn<Auto, Number> tableColumn_EngineVolume;
  @FXML private TableColumn<Auto, Number> tableColumn_MaxSpeed;

  @FXML private TextField brandTextField, yearTextField, volumeTextField, speedTextField;
  @FXML private Button addRecordButton, editRecordButton, deleteRecordButton;

  @FXML private PieChart chart;

  private FileManipulationController fileManipController;
  private TableManipulationController tableManipController;
  private ChartController chartController;

  public MainWindowController ()
  {}
```

```
  @FXML
  public void initialize ()
  {
    Logger.debug(getClass(), "Контроллер создан.");

    table.getSelectionModel().setSelectionMode(SelectionMode.SINGLE);
    chartController = new ChartController(table, chart);

    tableColumn_Brand.setCellValueFactory(x -> new SimpleStringProperty(x.getValue().brand));
    tableColumn_Year.setCellValueFactory(x -> new SimpleIntegerProperty(x.getValue().year));
    tableColumn_EngineVolume.setCellValueFactory(x -> new
SimpleFloatProperty(x.getValue().engineVolume));
    tableColumn_MaxSpeed.setCellValueFactory(x -> new SimpleIntegerProperty(x.getValue().maxSpeed));

    this.fileManipController = new FileManipulationController(table, chartController);
    loadFromFileButton.setOnMouseClicked(fileManipController::onLoadFromFileRequest);
    saveToFileButton.setOnMouseClicked(fileManipController::onSaveToFileRequest);

    this.tableManipController = new TableManipulationController(table, chartController, brandTextField,
      yearTextField, volumeTextField, speedTextField);
    addRecordButton.setOnMouseClicked(tableManipController::onAddDataRequest);
    editRecordButton.setOnMouseClicked(tableManipController::onEditDataRequest);
    deleteRecordButton.setOnMouseClicked(tableManipController::onDeleteDataRequest);

    table.setItems(FXCollections.observableArrayList());
    table.setOnMouseClicked(tableManipController::onTableRowSelected);
  }
}
```

Листинг 4 — Содержимое класса TableManipulationController

```
package org.cory7666.lab5;

import javafx.event.Event;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.input.MouseEvent;

public class TableManipulationController
{
  private final TableView<Auto> table;
  private final TextField brandTextField, yearTextField, volumeTextField, speedTextField;
  private final ChartController chartController;

  private Auto previoslySelectedItem = null;

  public TableManipulationController (TableView<Auto> table, ChartController chartController,
    TextField brandTextField, TextField yearTextField, TextField volumeTextField, TextField
speedTextField)
  {
    this.table = table;
    this.brandTextField = brandTextField;
    this.yearTextField = yearTextField;
    this.volumeTextField = volumeTextField;
    this.speedTextField = speedTextField;
    this.chartController = chartController;
  }

  public void onAddDataRequest (MouseEvent event)
  {
    Logger.debug(getClass(), "Добавление данных в таблицу.");

    try
    {
      var prevData = table.getItems();
      prevData.add(createObjectFromFields());

      chartController.updateChart();
    }
    catch (NumberFormatException ex)
    {
      Logger.error(getClass(), "Невозможно добавить: Одно из полей не заполнено.");
    }
    catch (Exception e)
    {
      Logger.error(getClass(), "Невозможно добавить: " + e.getMessage() + ".");
```

```
    }
  }

  public void onEditDataRequest (MouseEvent event)
  {
    Logger.debug(getClass(), "Изменение данных в таблице.");

    Auto newItemValue = createObjectFromFields();

    var data = table.getItems();
    data.add(data.indexOf(previoslySelectedItem), newItemValue);
    table.setItems(data);

    previoslySelectedItem = newItemValue;

    chartController.updateChart();
  }

  public void onDeleteDataRequest (MouseEvent event)
  {
    Logger.debug(getClass(), "Удаление данных из таблицы.");

    var data = table.getItems();
    data.remove(createObjectFromFields());
    table.setItems(data);

    chartController.updateChart();
  }

  public void onTableRowSelected (Event event)
  {
    previoslySelectedItem = table.getSelectionModel().getSelectedItem();

    if (previoslySelectedItem == null)
    {
      return;
    }

    writeObjectToFields(previoslySelectedItem);

    table.getSelectionModel().clearSelection();
  }

  private Auto createObjectFromFields ()
  {
    return new Auto(brandTextField.getText(), Integer.parseInt(yearTextField.getText()),
      Float.parseFloat(volumeTextField.getText()), Integer.parseInt(speedTextField.getText()));
  }

  private void writeObjectToFields (Auto o)
  {
    brandTextField.setText(o.brand);
    yearTextField.setText(o.year.toString());
    volumeTextField.setText(o.engineVolume.toString());
    speedTextField.setText(o.maxSpeed.toString());
  }
}
```

Листинг 5 — Содержимое класса Auto

```
package org.cory7666.lab5;

import com.opencsv.bean.CsvBindByName;

public class Auto
{
  @CsvBindByName (column = "Brand") public final String brand;
  @CsvBindByName (column = "Year") public final Integer year;
  @CsvBindByName (column = "Max Speed") public final Integer maxSpeed;
  @CsvBindByName (column = "Engine Volume") public final Float engineVolume;

  public Auto (String brand, Integer year, Float engineVolume, Integer maxSpeed)
  {
    if (brand == null)
    {
      throw new IllegalArgumentException("brand не может быть пустым.");
```

```
    }

    if (year < 1800)
    {
      throw new IllegalArgumentException("Год должен быть больше 1800.");
    }

    if (engineVolume < 0.0)
    {
      throw new IllegalArgumentException("Объём двигателя не может быть меньше нуля.");
    }

    if (maxSpeed <= 0)
    {
      throw new IllegalArgumentException("Максимальная скорость должна быть больше нуля.");
    }

    this.brand = brand;
    this.year = year;
    this.engineVolume = engineVolume;
    this.maxSpeed = maxSpeed;
  }

  public Auto ()
  {
    brand = "undefined";
    year = 0;
    maxSpeed = 0;
    engineVolume = 0.0f;
  }

  @Override
  public boolean equals (Object o)
  {
    if (o instanceof Auto other)
    {
      return this.brand.equals(other.brand) && this.engineVolume.equals(other.engineVolume)
        && this.year.equals(other.year) && this.maxSpeed.equals(other.maxSpeed);
    }
    else
    {
      return false;
    }
  }
}
```

Листинг 6 — Содержимое класса Logger

```
package org.cory7666.lab5;

import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonType;

public class Logger
{
  public static int logLevel = 0;

  public static void debug (Class<?> who, String message)
  {
    if (logLevel <= 0)
    {
      print("DEBUG", Thread.currentThread().getName(), who, message);
    }
  }

  public static void error (Class<?> who, String message)
  {
    if (logLevel <= 1)
    {
      print("ERROR", Thread.currentThread().getName(), who, message);
      new Alert(AlertType.ERROR, message, ButtonType.CLOSE).showAndWait();
    }
  }

  private static void print (String rock, String threadName, Class<?> who, String message)
```

```
  {
    System.out.printf("[%s] [%s/%s] %s\n", rock, threadName, who.getName(), message);
  }
}
```

Листинг 7 — Содержимое класса ChartController

```
package org.cory7666.lab5;

import java.util.HashMap;

import javafx.collections.FXCollections;
import javafx.scene.chart.PieChart;
import javafx.scene.control.TableView;

public class ChartController
{
  private final TableView<Auto> table;
  private final PieChart chart;

  public ChartController (TableView<Auto> table, PieChart chart)
  {
    this.table = table;
    this.chart = chart;
  }

  public void updateChart ()
  {
    var tableData = table.getItems();
    var counter = new HashMap<Float, Integer>();

    tableData.forEach(x -> {
      var c = counter.get(x.engineVolume);
      if (c == null)
      {
        counter.put(x.engineVolume, 1);
      }
      else
      {
        counter.replace(x.engineVolume, c + 1);
      }
    });

    chart
      .setData(FXCollections
        .observableList(counter
          .entrySet().stream().map(x -> new PieChart.Data(x.getKey().toString() + " попугаев",
x.getValue().doubleValue()))
          .toList()));;
  }
}
```

Листинг 8 — Содержимое класса DataFile

```
package org.cory7666.lab5;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.Iterator;
import java.util.List;

import com.opencsv.bean.CsvToBeanBuilder;
import com.opencsv.bean.StatefulBeanToCsvBuilder;
import com.opencsv.exceptions.CsvDataTypeMismatchException;
import com.opencsv.exceptions.CsvRequiredFieldEmptyException;

public class DataFile
{
  private final File file;

  public DataFile (File file)
  {
    this.file = file;
```

```
  }

  public List<Auto> readAll () throws IOException
  {
    try (var buffReader = Files.newBufferedReader(file.toPath()))
    {
      return new
CsvToBeanBuilder<Auto>(buffReader).withSeparator(';').withType(Auto.class).build().parse();
    }
  }

  public void writeAll (Iterator<Auto> collection)
    throws IOException, CsvDataTypeMismatchException, CsvRequiredFieldEmptyException
  {
    try (var writer = Files.newBufferedWriter(file.toPath()))
    {
      new StatefulBeanToCsvBuilder<Auto>(writer).withSeparator(';').build().write(collection);
    }
  }
}
```

Листинг 9 — Содержимое класса FileManipulationController

```
package org.cory7666.lab5;

import java.io.File;

import javafx.collections.FXCollections;
import javafx.scene.Node;
import javafx.scene.control.TableView;
import javafx.scene.input.MouseEvent;
import javafx.stage.FileChooser;
import javafx.stage.FileChooser.ExtensionFilter;
import javafx.stage.Window;

public class FileManipulationController
{
  private final TableView<Auto> table;
  private final ChartController chartController;

  public FileManipulationController (TableView<Auto> table, ChartController chartController)
  {
    this.table = table;
    this.chartController = chartController;
  }

  public void onLoadFromFileRequest (MouseEvent event)
  {
    try
    {
      Logger.debug(getClass(), "Запрос на загрузку данных из файла. Запрос пути к файлу.");

      Window scene = getSceneFromEvent(event);
      FileChooser chooser = new FileChooser();
      chooser.setTitle("Загрузить из...");
      chooser
        .getExtensionFilters()
        .addAll(new ExtensionFilter("CSV файл", "*.csv"), new ExtensionFilter("Все файлы", "*"));
      File inFile = chooser.showOpenDialog(scene);

      if (inFile == null)
      {
        Logger.debug(getClass(), "Пользователь отменил выбор файла.");
      }
      else
      {
        Logger.debug(getClass(), "Пользователь выбрал файл " + inFile.toString() + ".");
        table.setItems(FXCollections.observableArrayList(new DataFile(inFile).readAll()));
      }

      chartController.updateChart();
    }
    catch (Exception ex)
    {
      ex.printStackTrace();
    }
```

```
  }

  public void onSaveToFileRequest (MouseEvent event)
  {
    try
    {
      Logger.debug(getClass(), "Запрос на выгрузку данных из приложения.");

      Window scene = getSceneFromEvent(event);
      FileChooser chooser = new FileChooser();
      chooser.setTitle("Сохранить в...");
      chooser
        .getExtensionFilters()
        .addAll(new ExtensionFilter("CSV файл", "*.csv"), new ExtensionFilter("Все файлы", "*"));

      File outFile = chooser.showSaveDialog(scene);

      if (outFile != null)
      {
        Logger.debug(getClass(), "Выгрузка в " + outFile.toString() + ".");
        new DataFile(outFile).writeAll(table.getItems().iterator());
      }
      else
      {
        Logger.debug(getClass(), "Пользователь отменил операцию выгрузки.");
      }

      chartController.updateChart();
    }
    catch (Exception ex)
    {
      ex.printStackTrace();
    }
  }

  private Window getSceneFromEvent (MouseEvent event)
  {
    return ((Node) event.getSource()).getScene().getWindow();
  }
}
```

5.3.2. Результаты тестирования

Программа была скомпилирована и запущена.

Рисунок 1 демонстрирует вкладку с кнопками, позволяющими загрузить или выгрузить данные. Рисунок 2 демонстрирует вкладку с таблицей, содержащей загруженные из файла данные.

Рисунок 3 демонтрирует добавление элемента в таблицу.

Рисунок 4 демонстрирует результат попытки добавления элемента в список с одним незаполненным полем. Как и предполагалось, была выведена ошибка добавления.

Рисунок 5 демонстрирует вкладку с диаграммой, построенной на основе данных из таблицы.

Рисунок 1 — Вкладка с загрузкой/выгрузкой данных

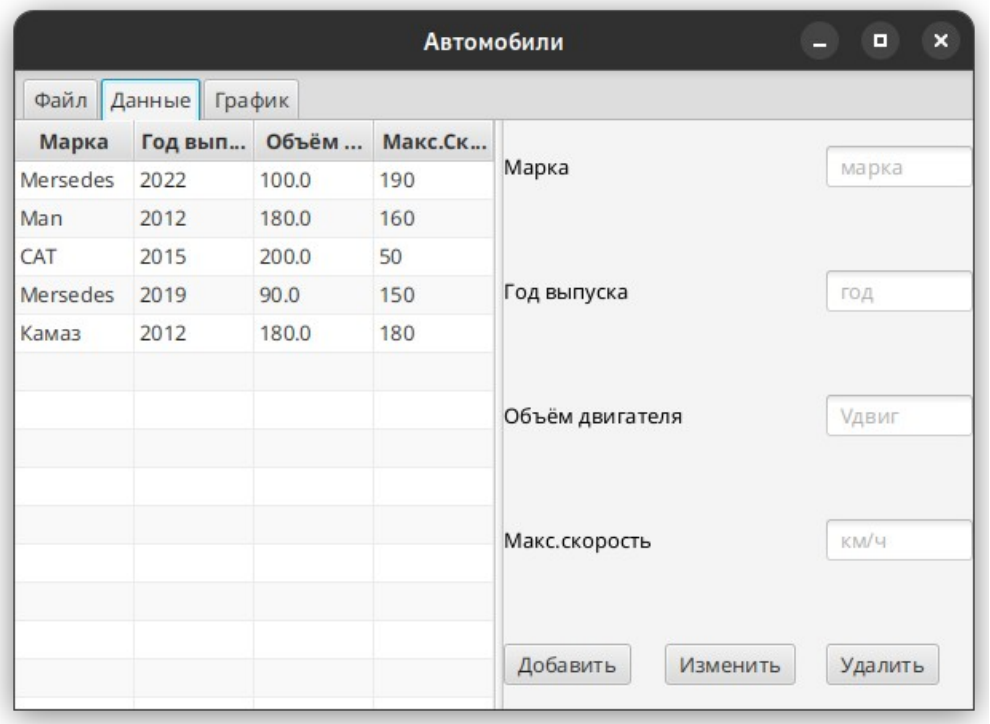

Рисунок 2 — Вкладка с таблицей

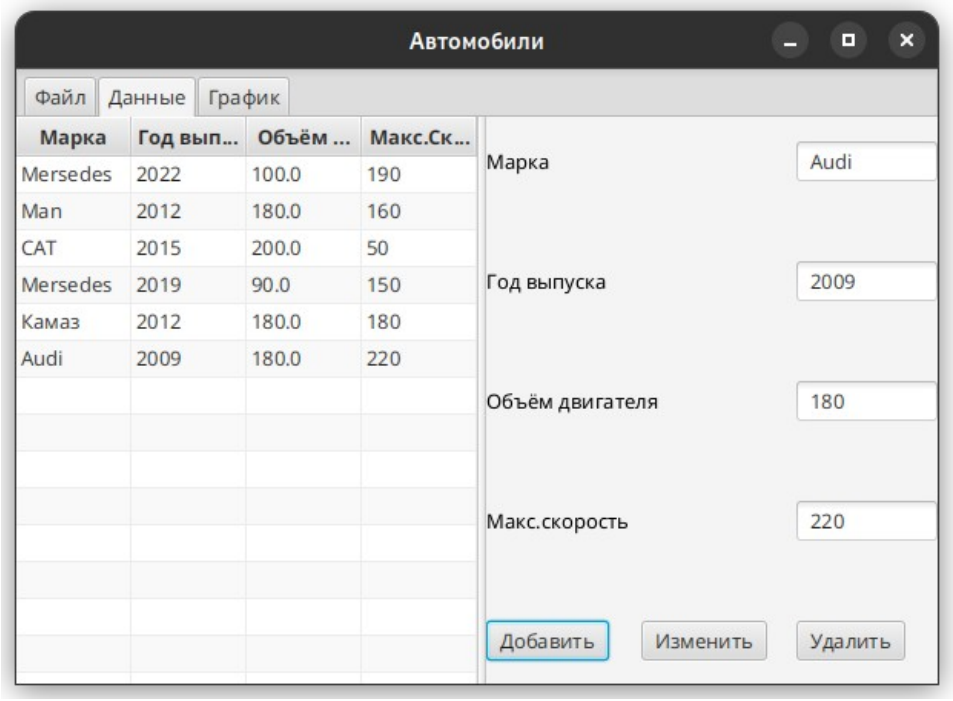

Рисунок 3 — Вставка элемента в таблицу

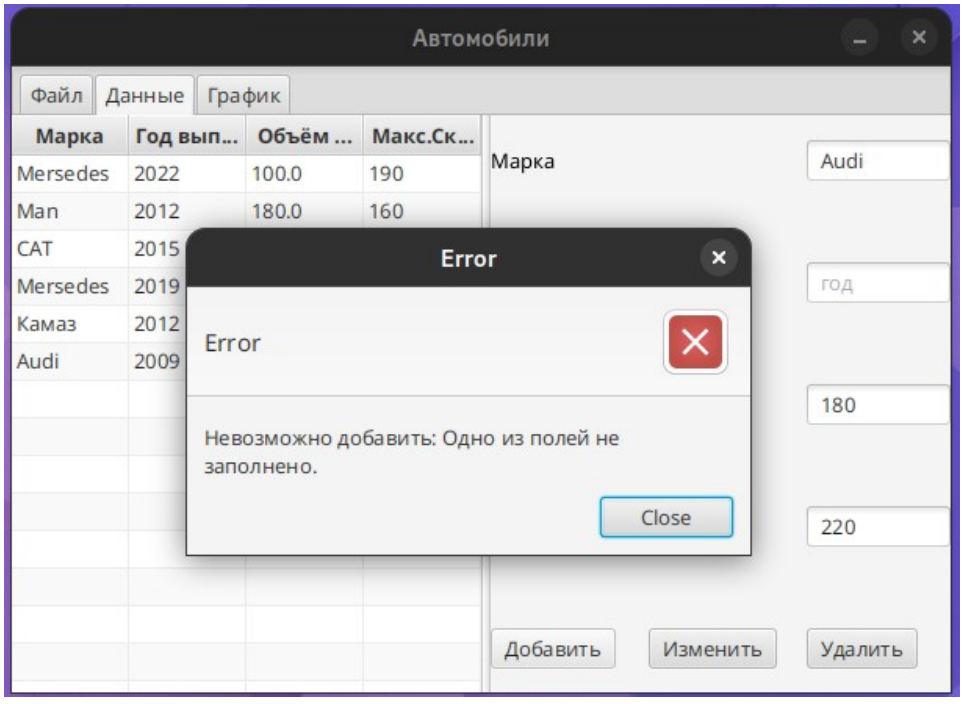

Рисунок 4 — Попытка вставки незаполненного элемента в таблицу

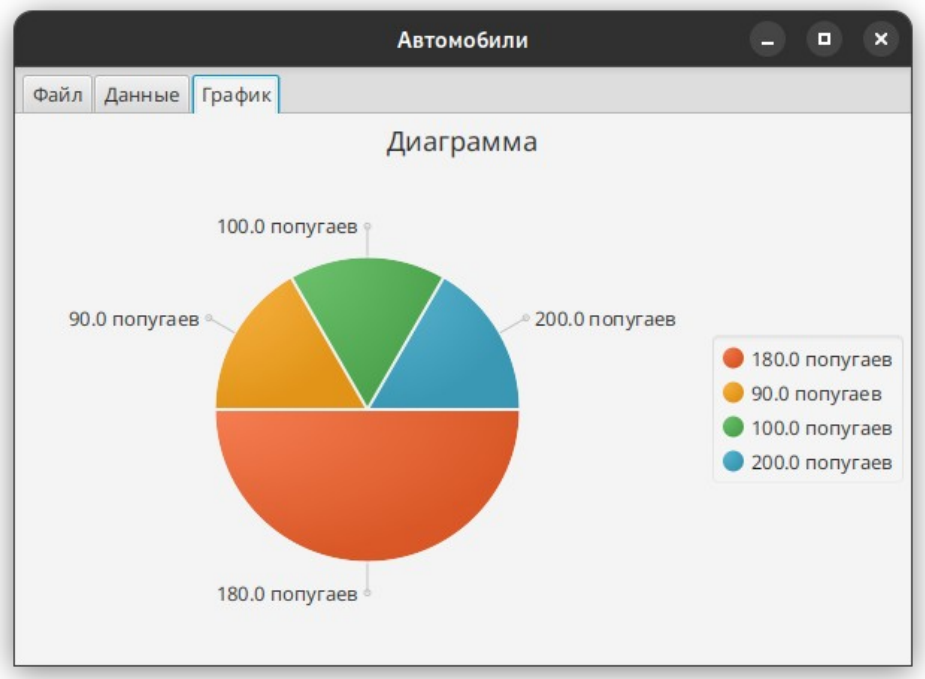

Рисунок 5 — Вкладка с диаграммой PieChart

Вывод

При выполнении данной лабораторной работы были получены навыки создания графического приложения с использованием платформы JavaFX 2. Были получены навыки создания диаграмм. Также были повторно закреплены навыки создания таблиц, импорта и экспорта данных.