Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №5

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ РАЗРАБОТКИ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПЛАТФОРМЫ JAVAFX 2

по дисциплине «Платформа Java»

Выполнил:

Студент группы ИС/б 17-2-о

Черняев Н.Г.

Проверил:

Кузнецов С.А.

г. Севастополь 2019

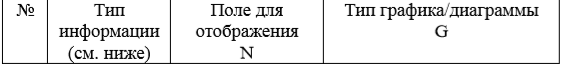
1.Цель работы

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

2.Постановка задачи

Вариант №10

С использованием компонентов JavaFX 2 необходимо создать Java приложение реализующее добавление, редактирование и удаление данных заданного по варианту типа информации T(см. табл. 4.1). 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(см. табл.).







3.Текст программы

Класс Book:

**package** lab5java;

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

**public** **class** Book {

**protected** **final** SimpleStringProperty author;

**protected** **final** SimpleStringProperty yearOfPublishing;

**protected** **final** SimpleStringProperty amountOfPages;

**protected** **final** SimpleStringProperty publishingHouse;

Book(String authorF, String yearOfPublishingF, String amountOfPagesF, String publishingHouseF) {

**this**.author = **new** SimpleStringProperty(authorF);

**this**.yearOfPublishing = **new** SimpleStringProperty(yearOfPublishingF);

**this**.amountOfPages = **new** SimpleStringProperty(amountOfPagesF);

**this**.publishingHouse = **new** SimpleStringProperty(publishingHouseF);

}

**public** String getAuthor() {

**return** author.get();

}

**public** String getYearOfPublishing() {

**return** yearOfPublishing.get();

}

**public** String getAmountOfPages() {

**return** amountOfPages.get();

}

**public** String getPublishingHouse() {

**return** publishingHouse.get();

}

**public** **void** setAuthor(String authorF) {

author.set(authorF);

}

**public** **void** setYearOfPublishing(String yearOfPublishingF) {

yearOfPublishing.set(yearOfPublishingF);

}

**public** **void** setAmountOfPages(String amountOfPagesF) {

amountOfPages.set(amountOfPagesF);

}

**public** **void** setPublishingHouse(String publishingHouseF) {

publishingHouse.set(publishingHouseF);

}

}

Класс Main:

**package** lab5java;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** javax.swing.JFileChooser;

**import** javafx.application.Application;

**import** javafx.collections.FXCollections;

**import** javafx.collections.ListChangeListener;

**import** javafx.collections.ObservableList;

**import** javafx.event.ActionEvent;

**import** javafx.event.EventHandler;

**import** javafx.geometry.Insets;

**import** javafx.scene.Group;

**import** javafx.scene.Scene;

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

**import** javafx.scene.control.Label;

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

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

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

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

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

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

**import** javafx.scene.layout.HBox;

**import** javafx.scene.layout.TilePane;

**import** javafx.scene.layout.VBox;

**import** javafx.scene.text.Font;

**import** javafx.stage.Stage;

**import** javafx.scene.chart.\*;

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

**private** TableView<Book> table = **new** TableView<Book>();

**private** **final** ObservableList<Book> data = FXCollections

.*observableArrayList*(**new** Book("Книга1", "1901", "333", "АТЛ"), **new** Book("Книга2", "1905", "256", "Махаон"));

**final** HBox hb = **new** HBox();

**final** HBox hb1 = **new** HBox();

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

PieChart chart = **new** PieChart();

**public** **void** UpdateChart() {

**int** i;

pcData.clear();

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

**try** {

pcData.add(**new** PieChart.Data(data.get(i).author.getValue().toString(),

Double.*parseDouble*(data.get(i).amountOfPages.getValue().toString())));

} **catch** (NumberFormatException e) {

System.***out***.println("Fields is empty");

}

}

chart.setData(pcData);

}

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

*launch*(args);

}

@SuppressWarnings({ "rawtypes", "unchecked" })

@Override

**public** **void** start(Stage stage) {

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

stage.setTitle("Table Sample");

stage.setWidth(900);

stage.setHeight(550);

**final** Label label = **new** Label("Книги");

label.setFont(**new** Font("Arial", 20));

label.setMaxWidth(300);

table.setEditable(**true**);

data.addListener(**new** ListChangeListener<Book>() {

@Override

**public** **void** onChanged(javafx.collections.ListChangeListener.Change<? **extends** Book> arg0) {

// **TODO** Auto-generated method stub

UpdateChart();

}

});

TableColumn authorCol = **new** TableColumn("Автор");

authorCol.setMinWidth(100);

authorCol.setCellValueFactory(**new** PropertyValueFactory<Book, String>("author"));

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

authorCol.setOnEditCommit(**new** EventHandler<CellEditEvent<Book, String>>() {

@Override

**public** **void** handle(CellEditEvent<Book, String> t) {

((Book) t.getTableView().getItems().get(t.getTablePosition().getRow())).setAuthor(t.getNewValue());

}

});

TableColumn yearOfPublishingCol = **new** TableColumn("Год издания");

yearOfPublishingCol.setMinWidth(100);

yearOfPublishingCol.setCellValueFactory(**new** PropertyValueFactory<Book, String>("yearOfPublishing"));

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

yearOfPublishingCol.setOnEditCommit(**new** EventHandler<CellEditEvent<Book, String>>() {

@Override

**public** **void** handle(CellEditEvent<Book, String> t) {

((Book) t.getTableView().getItems().get(t.getTablePosition().getRow()))

.setYearOfPublishing(t.getNewValue());

}

});

TableColumn amountOfPagesCol = **new** TableColumn("Количество страниц");

amountOfPagesCol.setMinWidth(100);

amountOfPagesCol.setCellValueFactory(**new** PropertyValueFactory<Book, String>("amountOfPages"));

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

amountOfPagesCol.setOnEditCommit(**new** EventHandler<CellEditEvent<Book, String>>() {

@Override

**public** **void** handle(CellEditEvent<Book, String> t) {

((Book) t.getTableView().getItems().get(t.getTablePosition().getRow()))

.setAmountOfPages(t.getNewValue());

}

});

TableColumn publishingHouseCol = **new** TableColumn("Издательство");

publishingHouseCol.setMinWidth(100);

publishingHouseCol.setCellValueFactory(**new** PropertyValueFactory<Book, String>("publishingHouse"));

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

publishingHouseCol.setOnEditCommit(**new** EventHandler<CellEditEvent<Book, String>>() {

@Override

**public** **void** handle(CellEditEvent<Book, String> t) {

((Book) t.getTableView().getItems().get(t.getTablePosition().getRow()))

.setPublishingHouse(t.getNewValue());

}

});

table.setItems(data);

table.getColumns().addAll(authorCol, yearOfPublishingCol, amountOfPagesCol, publishingHouseCol);

table.setMaxWidth(500);

**final** TextField addAuthor = **new** TextField();

addAuthor.setPromptText("Автор");

addAuthor.setMaxWidth(authorCol.getPrefWidth());

**final** TextField addYearOfPublishing = **new** TextField();

addYearOfPublishing.setMaxWidth(yearOfPublishingCol.getPrefWidth());

addYearOfPublishing.setPromptText("Год издания");

**final** TextField addAmountOfPages = **new** TextField();

addAmountOfPages.setMaxWidth(amountOfPagesCol.getPrefWidth());

addAmountOfPages.setPromptText("Количество страниц");

**final** TextField addPublishingHouse = **new** TextField();

addPublishingHouse.setMaxWidth(publishingHouseCol.getPrefWidth());

addPublishingHouse.setPromptText("Издательство");

**final** Button addButton = **new** Button("Добавить");

addButton.setOnAction(**new** EventHandler<ActionEvent>() {

@Override

**public** **void** handle(ActionEvent e) {

**if** ((addAuthor.getText().isEmpty()) || (addYearOfPublishing.getText().isEmpty())

|| (addAmountOfPages.getText().isEmpty()) || (addPublishingHouse.getText().isEmpty())) {

System.***out***.println("Fields is empty");

} **else** {

data.add(**new** Book(addAuthor.getText(), addYearOfPublishing.getText(), addAmountOfPages.getText(),

addPublishingHouse.getText()));

addAuthor.clear();

addYearOfPublishing.clear();

addAmountOfPages.clear();

addPublishingHouse.clear();

}

}

});

**final** Button deleteButton = **new** Button("Удалить");

deleteButton.setOnAction(**new** EventHandler<ActionEvent>() {

@Override

**public** **void** handle(ActionEvent e) {

data.remove(table.getSelectionModel().getSelectedItem());

}

});

**final** Button readButton = **new** Button("Загрузить");

readButton.setOnAction(**new** EventHandler<ActionEvent>() {

@SuppressWarnings("resource")

@Override

**public** **void** handle(ActionEvent e) {

JFileChooser fileChooser = **new** JFileChooser();

fileChooser.setCurrentDirectory(**new** File(System.*getProperty*("user.dir")));

**int** res = fileChooser.showOpenDialog(fileChooser);

**if** (res == JFileChooser.***APPROVE\_OPTION***) {

**try** {

BufferedReader br = **new** BufferedReader(**new** FileReader(fileChooser.getSelectedFile()));

{

String line = br.readLine();

String[] temp = line.split(",");

data.add(**new** Book(temp[0], temp[1], temp[2], temp[3]));

**while** ((line = br.readLine()) != **null**) {

temp = line.split(",");

data.add(**new** Book(temp[0], temp[1], temp[2], temp[3]));

}

}

} **catch** (IOException e1) {

e1.printStackTrace();

}

}

}

});

**final** Button writeButton = **new** Button("Сохранить");

writeButton.setOnAction(**new** EventHandler<ActionEvent>() {

@SuppressWarnings("resource")

@Override

**public** **void** handle(ActionEvent e) {

JFileChooser fileChooser = **new** JFileChooser();

fileChooser.setCurrentDirectory(**new** File(System.*getProperty*("user.dir")));

**int** res = fileChooser.showOpenDialog(fileChooser);

**if** (res == JFileChooser.***APPROVE\_OPTION***) {

FileWriter fw = **null**;

**try** {

fw = **new** FileWriter(fileChooser.getSelectedFile());

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

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

StringBuilder sb = **new** StringBuilder();

sb.append(data.get(i).author.getValue().toString());

sb.append(',');

sb.append(data.get(i).yearOfPublishing.getValue().toString());

sb.append(',');

sb.append(data.get(i).amountOfPages.getValue().toString());

sb.append(',');

sb.append(data.get(i).publishingHouse.getValue().toString());

sb.append("\r\n");

**try** {

fw.write(sb.toString());

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

sb = **null**;

}

**try** {

fw.close();

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

}

});

hb.getChildren().addAll(addAuthor, addYearOfPublishing, addAmountOfPages, addPublishingHouse);

hb.setSpacing(20);

hb.setMaxWidth(400);

hb1.getChildren().addAll(addButton, deleteButton, readButton, writeButton);

hb1.setSpacing(35);

hb1.setMaxWidth(400);

**final** VBox vbox = **new** VBox();

vbox.setSpacing(5);

vbox.setPadding(**new** Insets(10, 0, 0, 10));

vbox.getChildren().addAll(label, table, hb, hb1);

vbox.setMaxWidth(600);

chart.setTitle("Количество страниц в книгах");

chart.setPadding(**new** Insets(10, 0, 0, 10));

chart.setMaxWidth(400);

UpdateChart();

TilePane tilePane = **new** TilePane();

tilePane.getChildren().addAll(vbox, chart);

((Group) scene.getRoot()).getChildren().add(tilePane);

stage.setScene(scene);

stage.show();

}

}

4.Результаты

Графический интерфейс программы состоит из одного окна. В нём присутствуют кнопки загрузки из файла и сохранения в файл, удаления выделенной записи и добавления новой. Редактирование записей происходит прямо в ячейках таблицы. Окно представлено на рисунке 1.

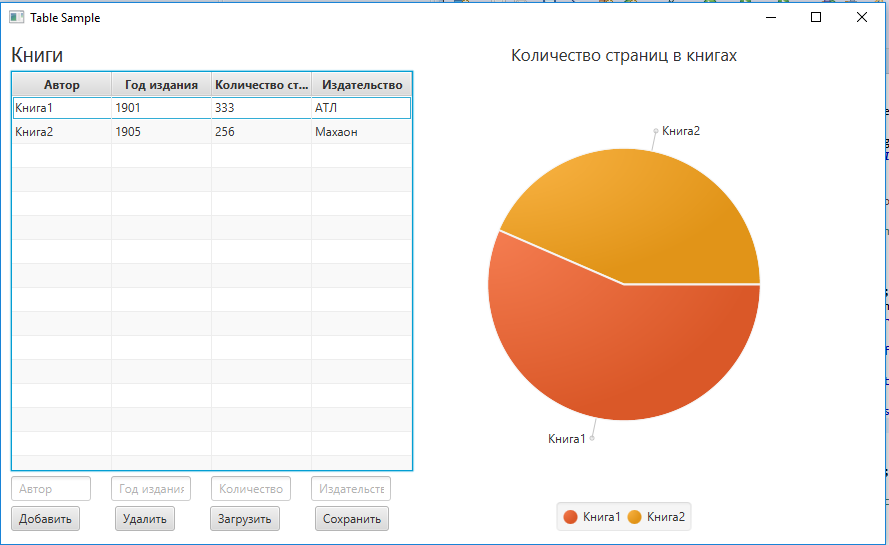


Рисунок 1 – Графический интерфейс программы

5.Вывод

В ходе выполнения данной лабораторной работы изучены особенности платформы JavaFX 2, приобретены практические навыки создания насыщенных пользовательских интерфейсов Java-программ. Изучено средство верстки представлений JavaFX 2 Scene Builder. В ходе работы было отмечено, что использование JavaFX 2 является отличным решением с точки зрения красоты и удобства интерфейса, однако потребляет достаточно много оперативной памяти, а именно в районе 100 МБ при старте.