Министерство образования и науки РФ

Институт информационных технологий и управления в технических системах

Севастопольский государственный университет

Лабораторная работа №5

по дисциплине: «Платформа Java»

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ РАЗРАБОТКИ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПЛАТФОРМЫ JAVAFX 2.

Выполнил:

ст.гр. ИСб/22о

Воронин И.Ю.

Проверил:

Кузнецов С.А.

г. Севастополь

2016 г.

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.

1. Вариант задания

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. 2.1).

Таблица 2.1 Варианты заданий

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип информации (см. ниже) | Поле для отображения  N | Тип графика/диаграммы  G |
| 2 | B | 3 | BarChart |

Тип информации:

B: Ноутбук (Идентификатор модели, Производитель процессора, Тактовая частота процессора, Объем ОЗУ);

1. Код программы

Класс MainApp:

public class MainApp extends Application {

private Stage primaryStage;

private BorderPane rootLayout;

//коллекция

private static ObservableList<Notebook> notebookData = FXCollections.observableArrayList();

public static ObservableList<Notebook> getNotebookList(){

return notebookData;

}

@Override

public void start(Stage primaryStage) {

this.primaryStage = primaryStage;

this.primaryStage.setTitle("AddressApp");

initRootLayout();

showPersonOverview();

}

public void initRootLayout() {

try {

// Load root layout from fxml file.

FXMLLoader loader = new FXMLLoader();

loader.setLocation(MainApp.class.getResource("view/RootLayout.fxml"));

rootLayout = (BorderPane) loader.load();

// Show the scene containing the root layout.

Scene scene = new Scene(rootLayout);

primaryStage.setScene(scene);

primaryStage.show();

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* Shows the person overview inside the root layout.

\*/

public void showPersonOverview() {

try {

// Load person overview.

FXMLLoader loader = new FXMLLoader();

loader.setLocation(MainApp.class.getResource("view/NotebookOverview.fxml"));

AnchorPane personOverview = (AnchorPane)loader.load();

// Set person overview into the center of root layout.

rootLayout.setCenter(personOverview);

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* Returns the main stage.

\* @return

\*/

public Stage getPrimaryStage() {

return primaryStage;

}

public static void main(String[] args) {

launch(args);

}

public void showStatistics() {

try {

// Load the fxml file and create a new stage for the popup.

FXMLLoader loader = new FXMLLoader();

loader.setLocation(MainApp.class.getResource("view/BarChartStatistic.fxml"));

AnchorPane page = (AnchorPane)loader.load();

Stage dialogStage = new Stage();

dialogStage.setTitle("Freq Statistics");

dialogStage.initModality(Modality.WINDOW\_MODAL);

dialogStage.initOwner(primaryStage);

Scene scene = new Scene(page);

dialogStage.setScene(scene);

// Set the persons into the controller.

BarChartStatisticController controller = loader.getController();

controller.setNotebookData(notebookData);

dialogStage.show();

} catch (IOException e) {

}

}

}

Класс BarChartStatisticController:

public class BarChartStatisticController {

@FXML

private BarChart<String, Double> barChart;

@FXML

private CategoryAxis xAxis;

private ObservableList<String> Names = FXCollections.observableArrayList();

@FXML

private void initialize() {

for(Notebook n: MainApp.getNotebookList()){

if(!Names.contains(n.CPU\_Maker.get())){

Names.add(n.CPU\_Maker.get());

}

}

xAxis.setCategories(Names);

}

public void setNotebookData(ObservableList<Notebook> notebook) {

// Count the number of people having their birthday in a specific month.

XYChart.Series<String, Double> series = new XYChart.Series<>();

int Counter[] = new int[100];

double Freq[] = new double[100];

for(int i = 0; i < 100; i++) {

Counter[i] = 0;

Freq[i] = 0;

}

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

for(Notebook n: notebook){

if(n.CPU\_Maker.get().equals(Names.get(i))){

Counter[i]++;

Freq[i]+= Double.valueOf(n.CPU\_Freq.get());

}

}

}

for(int i=0; i<100; i++){

if(Counter[i]>0){

Freq[i] /= Counter[i];

}

}

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

series.getData().add(new XYChart.Data<>(Names.get(i), Freq[i]));

}

barChart.getData().add(series);

}

}

Класс NotebookOverviewController:

public class NotebookOverviewController {

@FXML

private TableView<Notebook> personTable;

@FXML

private TableColumn<Notebook, String> IDNameColumn;

@FXML

private TableColumn<Notebook, String> MakerNameColumn;

@FXML

private TableColumn<Notebook, String> FreqNameColumn;

@FXML

private TableColumn<Notebook, String> RAMNameColumn;

@FXML

private Label labelID;

@FXML

private Label labelMaker;

@FXML

private Label labelFreq;

@FXML

private Label labelRAM;

@FXML

public TextField tfID;

@FXML

public TextField tfMaker;

@FXML

public TextField tfFreq;

@FXML

public TextField tfRAM;

@FXML

private Button butAdd;

@FXML

private Button butDel;

public NotebookOverviewController() {

}

public void Add(){

MainApp.getNotebookList().add(new Notebook(

new SimpleStringProperty(this.tfID.getText()),

new SimpleStringProperty(this.tfMaker.getText()),

new SimpleStringProperty(this.tfFreq.getText()),

new SimpleStringProperty(this.tfRAM.getText())));

}

public void Delete(){

if(MainApp.getNotebookList().size()>0){

int selectedIndex = personTable.getSelectionModel().getSelectedIndex();

personTable.getItems().remove(selectedIndex);

}

}

@FXML

private void initialize() {

personTable.setEditable(true);

/\*

col.setCellValueFactory(new Callback<TableColumn.CellDataFeatures<ObservableList,String>,ObservableValue<String>>(){

public ObservableValue<String> call(TableColumn.CellDataFeatures<ObservableList, String> param) {

return new SimpleStringProperty((String) param.getValue().get(j));

}

});

matrix.getColumns().add(col);

\*/

IDNameColumn.setCellValueFactory(cellData -> cellData.getValue().ID);

MakerNameColumn.setCellValueFactory(cellData -> cellData.getValue().CPU\_Maker);

FreqNameColumn.setCellValueFactory(cellData -> cellData.getValue().CPU\_Freq);

RAMNameColumn.setCellValueFactory(cellData -> cellData.getValue().RAM);

MakerNameColumn.setCellFactory(TextFieldTableCell.forTableColumn());

FreqNameColumn.setCellFactory(TextFieldTableCell.forTableColumn());

RAMNameColumn.setCellFactory(TextFieldTableCell.forTableColumn());

personTable.setItems(MainApp.getNotebookList());

}

public void setMainApp(MainApp mainApp) {

// Add observable list data to the table

personTable.setItems((ObservableList<Notebook>) mainApp.getNotebookList());

}

}

Класс RootOverviewController:

public class RootOverviewController {

MainApp mainapp = new MainApp();

public void BarChartShow(){

mainapp.showStatistics();

}

public void FileOpen(){

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Открыть файл");

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

File file = fileopen.getSelectedFile();

try {

FileRead(file.getAbsolutePath());

} catch (Exception e1) {

}

}

}

public void FileSave(){

JFileChooser fileopen = new JFileChooser();

int ret = fileopen.showDialog(null, "Сохранить в файл");

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

File file = fileopen.getSelectedFile();

try {

FileWrite(file.getAbsolutePath());

} catch (IOException e1) {

}

}

}

public void FileClose(){

System.exit(0);

}

public void FileWrite(String Filename)throws IOException{

FileWriter f = new FileWriter(Filename);

for(Notebook n : MainApp.getNotebookList()){

f.write(n.ID.getValue() + "\r\n" +

n.CPU\_Maker.getValue() + "\r\n" +

n.CPU\_Freq.getValue() + "\r\n" +

n.RAM.getValue() + "\r\n\r\n");

}

f.close();

}

public void FileRead(String Filename) throws FileNotFoundException, IOException{

MainApp.getNotebookList().clear();

Scanner fin = new Scanner(new File(Filename));

String author, ID, RAM, CPU;

while(fin.hasNext())

{

if(MainApp.getNotebookList().size()>0){

fin.nextLine();

}

ID = fin.nextLine();

author = fin.nextLine();

CPU = fin.nextLine();

RAM = fin.nextLine();

MainApp.getNotebookList().add(new Notebook(

new SimpleStringProperty(ID),

new SimpleStringProperty(author),

new SimpleStringProperty(CPU),

new SimpleStringProperty(RAM)

));

}

fin.close();

}

}

1. Тестовые примеры

При запуске программы на экран выводится следующее окно. В данном окне для добавления элементов необходимо заполнить текстовые поля в левой части экрана и нажать кнопку «Добавить». Аналогичным способом является выбор пункта «File->Open» и в окне выбрать текстовый файл, с которого будут считаны данные. Элементы будут отображаться в таблице в правой части экрана (рис.4.1)

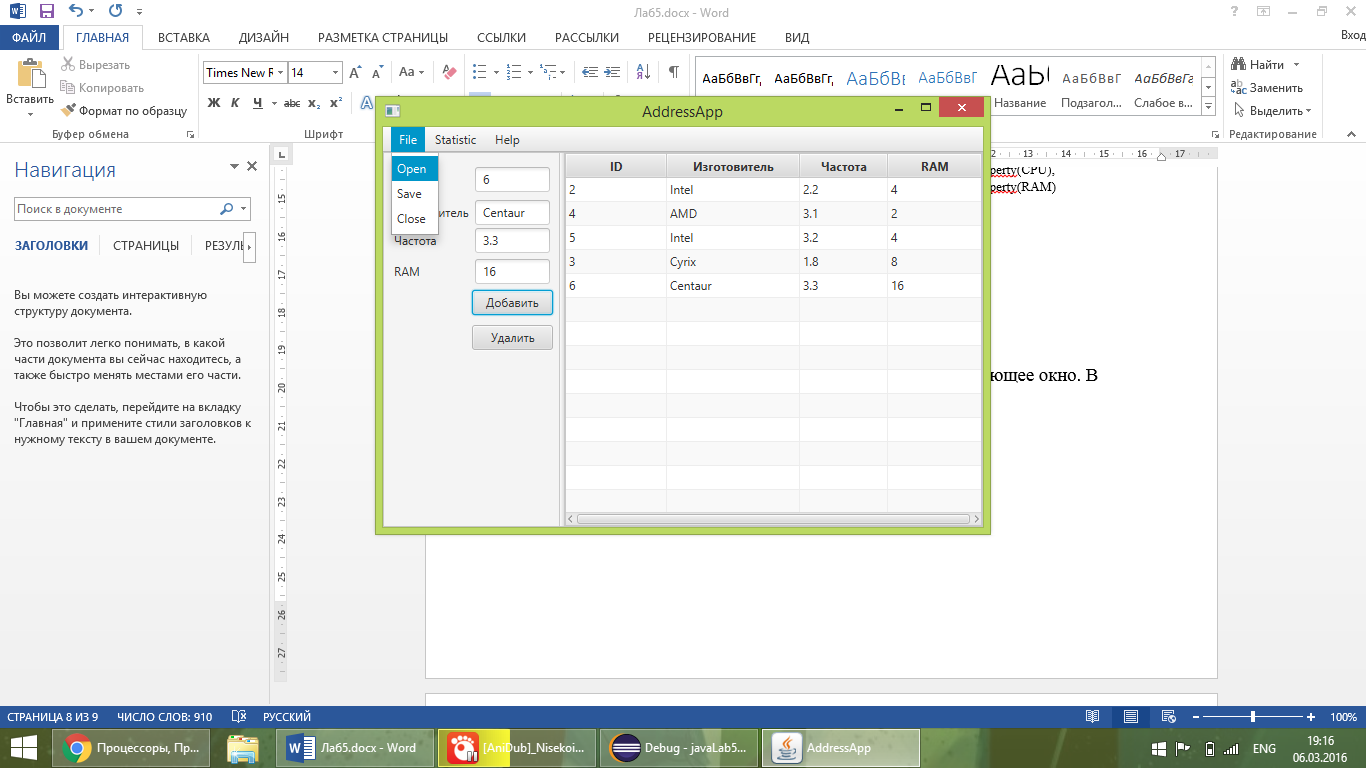


Рисунок 4.1 – Добавление элемента.

Нажав на поле столбца таблицы (кроме первого столбца) оно становится открытым для редактирования. Выделив столбец и нажав кнопку «Удалить», элемент будет удалён из коллекции.

Выбрав пункт «File->Save» можно выбрать в файл, куда будут сохранены данные таблицы в текстовой форме.

Выбрав пункт «Statistic->BarChart» можно получить данные о таблице в виду столбчатой диаграммы, где производители процессоров сравниваются по Средней тактовой частоте процессоров. (рис.4.2)

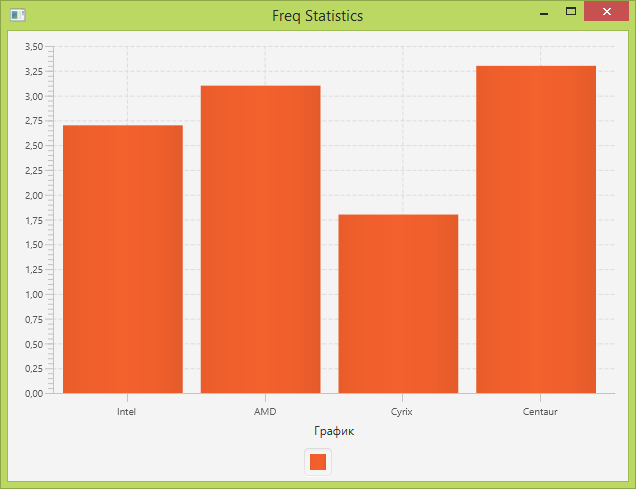


Рисунок 4.2 – BarChart.

ВЫВОДЫ

В данной лабораторной работе была написана программа на платформе JavaFX 2.0 в среде Eclipse Mars с использованием конструктора SceneBuilder 2.0. Для создания программы мы создавали FXML-Document, который редактировался при помощи указанного конструктора, а после необходимо было связать его с классом-контролером, который отвечал за весь функционал.

Была реализована столбчатая диаграмма, которая отображала соотношение данных в таблице, ввод и вывод в текстовый файл.