Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Севастопольский государственный университет»

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ РАЗРАБОТКИ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ПЛАТФОРМЫ JAVAFX 2

Методические указания
к выполнению лабораторной работы
для студентов, обучающихся по направлению
09.03.02 “Информационные системы и технологии”
очной и заочной форм обучения

Севастополь
2015

УДК 004.42 (075.8)

Исследование возможностей разработки приложений с использованием платформы JavaFX2: методические указания к лабораторной работе №5 по дисциплине “Платформа Java” для студентов направления 09.03.02 “Информационные системы и технологии”/ Сост. **С.А. Кузнецов, А.Л. Овчинников** — Севастополь: Изд-во СевГУ, 2015. — 14 с.

Цель указаний: оказание помощи студентам направления 09.03.02 “Информационные системы и технологии” при выполнении лабораторной работы №5 по дисциплине “Платформа Java”.

Методические указания составлены в соответствии с требованиями программы дисциплины «Платформа Java» для студентов дневной и заочной формы обучения направления 09.03.02 “Информационные системы и технологии” и утверждены на заседании кафедры «Информационные системы» протоколом № 1 от 31 августа 2015 года.

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

Рецензент: Кожаев Е.А., канд. техн. наук, доцент кафедры кибернетики и вычислительной техники

СОДЕРЖАНИЕ

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

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. 4.1).

3. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

3.1 Общие сведения о платформе JavaFX 2

Платформа JavaFX 2 обеспечивает создание мощного графического интерфейса пользователя (Graphical User Interface (GUI)) для крупномасштабных приложений, ориентированных на обработку данных, насыщенных медиа-приложений, поставляющих разнообразный медиа-контент пользователю, компонентов высококачественной графики и анимации для Web-сайтов, различного рода пользовательских программ, насыщенных графикой, анимацией и интерактивными элементами.

Один и тот же Java-код, созданный на базе платформы JavaFX 2, может запускаться как десктопное приложение, которое разворачивается на клиентском компьютере автономно, но может разворачиваться и как Web-приложение на веб сервере(), или отображаться в Web-браузере как JavaFX-апплет, встроенный в HTML-страницу.

Платформа JavaFX 2 предоставляет современные GUI-компоненты, богатый набор графических и медиа библиотек, а также высокопроизводительную среду выполнения приложений.

JavaFX 2.2 и более поздних версий имеет следующие особенности:

- *Java API*. JavaFX представляет собой библиотеку Java, которая состоит из классов и интерфейсов, которые написаны на языке Java.
- *WebView*. Веб-компонент, использующий движок WebKit для отображения веб-страниц и позволяющий вставлять веб-страницы в приложения JavaFX.
- *Совместимость со Swing*. Существующие приложения Swing могут быть дополнены новыми возможностями JavaFX - богатая графика, воспроизведение медиа-файлов и встроенное веб-содержимое.

- *Встроенные элементы управления пользовательского интерфейса*. JavaFX предоставляет все основные элементы управления пользовательского интерфейса, необходимые для разработки полнофункциональных приложений.
- *Canvas API*. Canvas API позволяет рисовать непосредственно в области сцены JavaFX.
- *Поддержка мультитач*. JavaFX обеспечивает поддержку мультитач, основываясь на возможностях аппаратной платформы.
- *Аппаратное ускорение графического конвейера*. Графика JavaFX основана на использовании графического конвейера рендеринга Prism, обеспечивающего высокую производительность при использовании поддерживаемых моделей графического процессора.

Платформа JavaFX 2 интегрирована в JDK 7, и не требует отдельной инсталляции. Для создания JavaFX-приложения в Eclipse достаточно подключить библиотеку **sdk-path/rt/lib/jfxrt.jar (или jdk-path/jre/lib/jfxrt.jar)** в свойствах проекта.

3.2 Создание JavaFX 2 приложений

Точкой входа в JavaFX-приложение служит Java-класс, расширяющий абстрактный класс javafx.application.Application и содержащий метод main():

```
public class JavaFXApp extends Application {
   public static void main(String[] args) {
      launch(args);
   }
   public void init(){

. . .
}
   @Override
   public void start(Stage primaryStage) {

. . .
      primaryStage.setScene(scene);
      primaryStage.setVisible(true);
}

public void stop(){
. . .
}
}
```

В методе main() главного класса JavaFX-приложения вызывается метод launch() класса Application, отвечающий за загрузку JavaFX-приложения. Кроме того, главный класс JavaFX-приложения должен переопределить абстрактный метод start()

класса Application, обеспечивающий создание и отображение *сцены* JavaFX-приложения.

Методы init() и stop() класса Application могут использоваться для инициализации данных и освобождения ресурсов JavaFX-приложения.

Обработка входных аргументов или параметров в главном классе JavaFX-приложения может быть осуществлена с помощью вызова метода getParameters() класса Application.

Метод start() класса Application содержит в качестве параметра объект javafx.stage.Stage, представляющий *графический контейнер главного окна* JavaFX-приложения. Объект Stage создается средой выполнения при запуске JavaFX-приложения и передается в метод start() главного класса JavaFX-приложения, что позволяет использовать методы объекта Stage для установки и отображения *сцены* JavaFX-приложения. Вместо объекта Stage, аргумента метода start(), разработчик может создать свой экземпляр класса Stage для отображения сцены JavaFX-приложения.

Перед отображением сцены в контейнере Stage главного окна JavaFX-приложения необходимо создать *граф сцены* - иерархическое дерево узлов, состоящее из корневого узла и его дочерних элементов, и на его основе создать объект javafx.scene.Scene.

Как правило, в качестве корневого узла используется объект javafx.scene.Group, который позволяет группировать элементы, однако в качестве корневого узла может использоваться и один из множества менеджеров компоновки(javafx.scene.layout).

Дочерние узлы графа сцены, представляющие графику, элементы контроля GUI-интерфейса, медиа контент, добавляются в корневой узел с помощью метода getChildren().add() или метода getChildren().addAll(). При этом дочерние узлы могут иметь визуальные эффекты, режимы наложения, CSS-стили, прозрачность, трансформации, обработчики событий, участвовать в анимации по ключевым кадрам, программируемой анимации и др.

В примере представленном ниже в граф сцены добавляется экземпляр круговой диаграммы(PieChart):

```
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.chart.*;
import javafx.scene.Group;

public class PieChartSample extends Application {

    @Override public void start(Stage stage) {
        Scene scene = new Scene(new Group());
        stage.setTitle("Использование языков программирования");
        stage.setWidth(500);
        stage.setHeight(500);

       ObservableList<PieChart.Data> pieChartData =
```

```
                FXCollections.observableArrayList(
                new PieChart.Data("Java", 18),
                new PieChart.Data("C", 17),
                new PieChart.Data("C++", 8),
                new PieChart.Data("Objective-C", 7),
                new PieChart.Data("PHP", 6));
        final PieChart chart = new PieChart(pieChartData);
        chart.setTitle("Использование языков программрования");

        ((Group) scene.getRoot()).getChildren().add(chart);
        stage.setScene(scene);
        stage.show();
    }

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

Результат выполнения программы представлен на рис. 1:

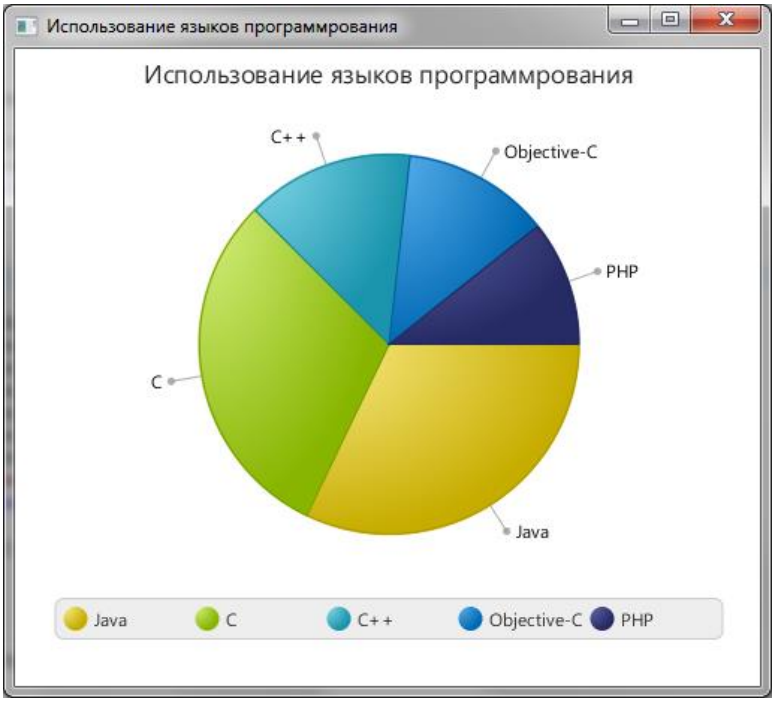

Рисунок 1 – Результат работы программы

В качестве узлов графа сцены могут быть использованы разнообразные элементы управления. Вид элементов управления JavaFX 2 представлен на рисунке 2:

Рисунок 2 – Элементы управления JavaFX 2

В приложении А представлен комплексный пример использования графических компонентов JavaFX для редактирования и отображения данных.

4. ВАРИАНТЫ ЗАДАНИЙ

Таблица 4.1 Варианты заданий

№	Тип информации (см. ниже)	Поле для отображения N	Тип графика/диаграммы G
1	A	3	PieChart
2	B	3	BarChart
3	C	3	StackedBarChart
4	D	2	PieChart
5	E	3	BarChart
6	A	4	StackedBarChart
7	B	4	PieChart
8	C	4	BarChart
9	D	4	StackedBarChart
10	E	3	PieChart
11	A	3	BarChart
12	B	3	StackedBarChart
13	C	3	PieChart
14	D	2	BarChart
15	E	3	StackedBarChart

Тип информации:

A: Компакт диск(Название альбома, Исполнитель, Количество треков, Длительность звучания);

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

C: Автомобиль(Марка, Год выпуска, Объем двигателя, Максимальная скорость);

D: Смартфон(Модель, Размер экрана, Тип экрана, Объем встроенной флэш-памяти).

E: Книга(Автор, Год издания, Количество страниц, Издательство);

5. СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать:

Титульный лист, цель работы, постановку задачи, вариант задания, текст программы с комментариями, скриншоты выполнения и описание тестовых примеров, выводы по работе.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Расскажите о платформе JavaFX?
2. Каковы особенности JavaFX 2?
3. Возможно ли совместное использование JavaFX 2 и библиотеки Swing?
4. Что такое сцена JavaFX 2?
5. Приведите пример каркаса JavaFX 2 приложения?
6. Что такое граф сцены?
7. Какие элементы ввода данных реализованы в JavaFX 2?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ноутон, П. Java™ 2 [Текст] : пер. с англ. / П. Ноутон, Г. Шилдт. - СПб. : БХВ – Петербург, 2007. - 1050 с.

2. Шилдт, Г. Искусство программирования на Java [Текст] : пер. с англ. / Г. Шилдт, Д. Холмс. - М. ; СПб. ; К. : Вильямс, 2005. - 334 с.

3. Хабибуллин, И. Ш. Java 2 [Текст] : самоучитель / И. Ш. Хабибуллин. - СПб. : БХВ - Петербург, 2005. - 720 с.

4. Портянкин, И. А. Swing: эффективные пользовательские интерфейсы [Текст] / И. А. Портянкин. - М. и др. : Питер, 2005. - 528 с.

5. Шилдт, Г. Swing: Руководство для начинающих [Текст] : пер. с англ. / Г. Шилдт - М. ; СПб. ; К. : Вильямс, 2007. – 967 с.

ПРИЛОЖЕНИЕ А

Пример JavaFX 2 приложения

А.1 Интерфейс JavaFX приложения

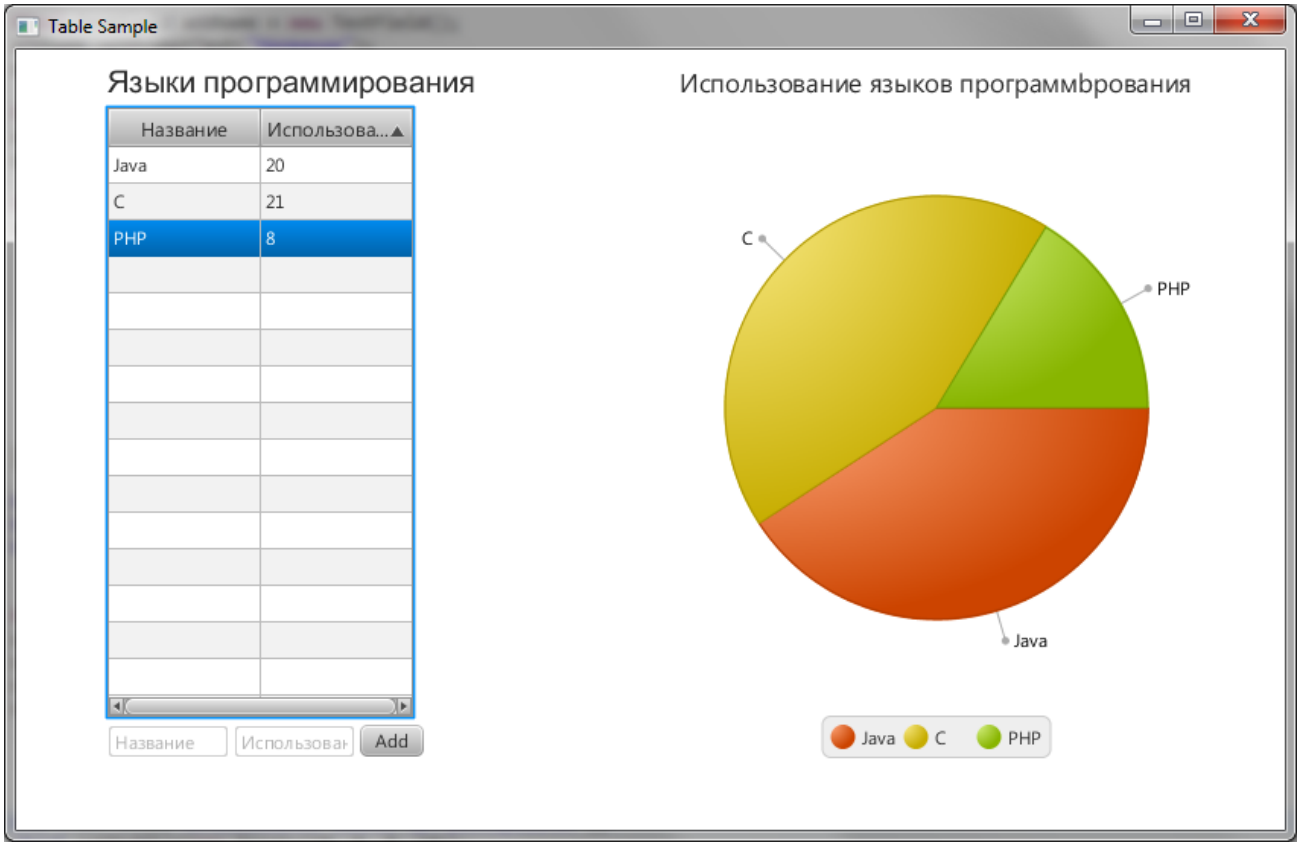

Рисунок А.1 Окно JavaFX приложения

А.2 Текст программы

```
import javafx.application.Application;
import javafx.beans.property.SimpleStringProperty;
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 TableViewExample extends Application {

    private TableView<Language> table = new TableView<Language>();
    private final ObservableList<Language> data =
            FXCollections.observableArrayList(
            new Language("C", "60"),
            new Language("Java", "40"));

    final HBox hb = 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++){
            pcData.add(new PieChart.Data(data.get(i).langName.getValue().toString(),

      Double.parseDouble(data.get(i).langPercent.getValue().toString())));
      }
      chart.setData(pcData);
    }


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

    @Override
    public void start(Stage stage) {
        Scene scene = new Scene(new Group());
        stage.setTitle("Table Sample");
        stage.setWidth(850);
        stage.setHeight(550);

        final Label label = new Label("Языки программирования");
        label.setFont(new Font("Arial", 20));
        label.setMaxWidth(300);

        table.setEditable(true);

             data.addListener(new ListChangeListener<Language>() {
                   @Override
                   public void onChanged(
                               javafx.collections.ListChangeListener.Change<? extends
Language> arg0) {
                         // TODO Auto-generated method stub
                         UpdateChart();
                   }
            });


        TableColumn langNameCol = new TableColumn("Название");
        langNameCol.setMinWidth(100);
        langNameCol.setCellValueFactory(
```

```
    new PropertyValueFactory<Language, String>("langName"));
langNameCol.setCellFactory(TextFieldTableCell.forTableColumn());
langNameCol.setOnEditCommit(
    new EventHandler<CellEditEvent<Language, String>>() {
        @Override
        public void handle(CellEditEvent<Language, String> t) {
            ((Language) t.getTableView().getItems().get(
                    t.getTablePosition().getRow())
                    ).setLangName(t.getNewValue());
        }
    }
);



TableColumn langPercentCol = new TableColumn("Использование");
langPercentCol.setMinWidth(100);
langPercentCol.setCellValueFactory(
    new PropertyValueFactory<Language, String>("langPercent"));
     langPercentCol.setCellFactory(TextFieldTableCell.forTableColumn());
     langPercentCol.setOnEditCommit(
                new EventHandler<CellEditEvent<Language, String>>() {
                @Override
                        public void handle(CellEditEvent<Language, String> t) {
                        ((Language) t.getTableView().getItems().get(
                                    t.getTablePosition().getRow())
                ).setLangPercent(t.getNewValue());
        }
    }
);

table.setItems(data);
table.getColumns().addAll(langNameCol, langPercentCol);
table.setMaxWidth(201);

final TextField addName = new TextField();
addName.setPromptText("Название");
addName.setMaxWidth(langNameCol.getPrefWidth());

final TextField addLang = new TextField();
addLang.setMaxWidth(langPercentCol.getPrefWidth());
addLang.setPromptText("Использование");

final Button addButton = new Button("Add");
addButton.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent e) {
        data.add(new Language(
                addName.getText(),
                addLang.getText()));
        addName.clear();
        addLang.clear();

    }
});

hb.getChildren().addAll(addName, addLang, addButton);
hb.setSpacing(3);
hb.setMaxWidth(300);


final VBox vbox = new VBox();
vbox.setSpacing(5);
```

```
            vbox.setPadding(new Insets(10, 0, 0, 10));
        vbox.getChildren().addAll(label, table, hb);
        vbox.setMaxWidth(300);

        chart.setTitle("Использование языков программbрования");
        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();
    }


    public static class Language {

        private final SimpleStringProperty langName;
        private final SimpleStringProperty langPercent;

        private Language(String fName, String lName) {
            this.langName = new SimpleStringProperty(fName);
            this.langPercent = new SimpleStringProperty(lName);
        }

        public String getLangName() {
            return langName.get();
        }

        public void setLangName(String fName) {
            langName.set(fName);
        }

        public String getLangPercent() {
            return langPercent.get();
        }

        public void setLangPercent(String fName) {
            langPercent.set(fName);
        }

    }
      }
```

Заказ № _____ от «_____» _____2015г. Тираж _____экз.
Изд-во СевГУ