Донецкий Национальный Технический Университет

Лабораторная работа № 2

«Конструирование сканеров»

Выполнил:

ст. группы ИПЗ -13

Лысенко А. С.

Проверила:

доцент каф. ПМИ

Дмитриева О. А.

Покровск 2017

Задание

**1.2.1 Таблиці для транслятора з мови C**

4) Таблиця операцій і її ініціалізація.

|  |  |  |  |
| --- | --- | --- | --- |
| id | обозначение операции | приоритет | ассоциативность |
| 1 | + | 4 | Слева-направо |
| 2 | - | 4 | Слева-направо |
| 3 | \* | 3 | Слева-направо |
| 4 | / | 3 | Слева-направо |
| 5 | % | 3 | Слева-направо |
| 6 | == | 7 | Слева-направо |

Программа написанная на javafx с использованием графического редактора scene builder 2.0. Программа представляет собой сканнер. Данные задаются статически в виде массива объектов, при условии их обнаружения в тексте программы.

Main.java – запускает главное окно программы

**package** sample;  
  
**import** javafx.application.Application;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.layout.Pane;  
**import** javafx.stage.Stage;  
  
**public class** Main **extends** Application {  
  
 @Override  
 **public void** start(Stage primaryStage) **throws** Exception  
 {  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"sample.fxml"**));  
 primaryStage.setTitle(**"Scanner"**);  
 *// минимальные размеры окна* primaryStage.setMinWidth(550);  
 primaryStage.setMinHeight(600);  
 primaryStage.setScene(**new** Scene(root, 550, 600));  
 primaryStage.show();  
}  
**public static void** main(String[] args) {  
 *launch*(args);  
 }  
}

Controller.java – обработчик кнопок главного окна

**package** sample;  
  
**import** javafx.event.ActionEvent;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.PropertyValueFactory;  
**import** javafx.stage.Modality;  
**import** javafx.stage.Stage;  
  
**import** javax.swing.\*;  
**import** javax.xml.soap.Node;  
  
  
**import** java.io.DataInputStream;  
**import** java.io.File;  
**import** java.io.FileInputStream;  
**import** java.io.IOException;  
  
**public class** Controller  
{  
  
 @FXML  
 **private** Label **labelCount**;  
 @FXML  
 **private** CollectionLeksem **collectionLeksemImpl** = **new** CollectionLeksem();  
  
 **private** CollectionLeksem1 **collectionLeksemImpl1** = **new** CollectionLeksem1();  
  
 @FXML  
 **private** TableColumn<Leksem, Integer> **columnId**;  
  
 @FXML  
 **private** TableView **tableScanner**;  
  
 @FXML  
 **private** TableColumn<Leksem, String> **columnSymbol**;  
  
 @FXML  
 **private** TableColumn<Leksem, String> **columnLeksname**;  
  
 String **sss** = **"#include <assert.h>\n"** +  
 **"#include <stdio.h>\n"** +  
 **"int main ()\n"** +  
 **"{\n"** +  
 **"int a, b;\n"** +  
 **"int r = scanf(\"%d%d\";, &a, &b);\n"** +  
 **"assert (r == 2);\n"** +  
 **"printf (\"%d\\n\", a + b);\n"** +  
 **"return 0;\n"** +  
 **"}\""**;  
  
 **public** TextArea **txtArea**;  
 @FXML  
 **private** Button **btnAdd**;  
 @FXML  
 **private** Button **btnDelete**;  
 @FXML  
 **private** Button **btnScan**;  
  
 **public void** initialize()  
 {  
 **txtArea**.setText(**sss**);  
  
  
 }  
  
 **public void** scanText(ActionEvent actionEvent)  
 {  
**txtArea**.setText(**txtArea**.getText());  
 **txtArea**.setWrapText(**true**);  
 *// int length = txtArea.getText().length();* String searchChar = **"+"**;  
 **char** searchChar1 = **'-'**;  
 **char** searchChar2 = **'\*'**;  
 **char** searchChar3 = **'/'**;  
 **char** searchChar4 = **'%'**;  
 String searchChar5 = **"=="**;  
 String searchChar6 = **"="**;  
  
 **boolean** isFound = **false**;  
 */\*for(int i = 0; i < length; i++)  
 {  
 if(txtArea.getText().charAt(i) == searchChar))  
 {  
 isFound = true;  
 break;  
 }  
 }  
 System.out.println(message(isFound) + searchChar);\*/* message(**txtArea**.getText().indexOf(searchChar) != -1);  
 message1(**txtArea**.getText().indexOf(searchChar1) != -1);  
 message2(**txtArea**.getText().indexOf(searchChar2) != -1);  
 message3(**txtArea**.getText().indexOf(searchChar3) != -1) ;  
 message4(**txtArea**.getText().indexOf(searchChar4) != -1);  
 message5(**txtArea**.getText().indexOf(searchChar5) != -1);  
  
 }  
  
 String **searchChar** = **"+"**;  
 String **searchChar1** = **"-"**;  
 String **searchChar2** = **"\*"**;  
 String **searchChar3** = **"/"**;  
 String **searchChar4** = **"%"**;  
 String **searchChar5** = **"=="**;  
  
 **private** String message(**boolean** b)  
 {  
**if** (b && **searchChar** == **"+"**) {  
 **txtArea**.setText(**txtArea**.getText().replace(**"+"**, **"$LeksPlus"**));  
  
 **collectionLeksemImpl**.testData();  
  
 **columnId**.setCellValueFactory(**new** PropertyValueFactory<Leksem, Integer>(**"id"**));  
 **columnSymbol**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"symbol"**));  
 **columnLeksname**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"leksem"**));  
  
 **tableScanner**.setItems(**collectionLeksemImpl**.getLeksemList());  
  
 updateCountLabel();  
 }**return ""**;  
  
 }  
  
 **private** String message1(**boolean** b) {  
 **if** (b && **searchChar1** == **"-"**) {  
 **txtArea**.setText(**txtArea**.getText().replace(**"-"**, **"$LeksMinus"**));  
  
 **collectionLeksemImpl**.testData1();  
 **columnId**.setCellValueFactory(**new** PropertyValueFactory<Leksem, Integer>(**"id"**));  
 **columnSymbol**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"symbol"**));  
 **columnLeksname**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"leksem"**));  
  
 **tableScanner**.setItems(**collectionLeksemImpl**.getLeksemList());  
  
 updateCountLabel();  
  
 }  
 **return ""**;  
 }  
  
 **private** String message2(**boolean** b) {  
 **if** (b && **searchChar2** == **"\*"**) {  
 **txtArea**.setText(**txtArea**.getText().replace(**"\*"**, **"$LeksMultiply"**));  
 **collectionLeksemImpl**.testData2();  
 **columnId**.setCellValueFactory(**new** PropertyValueFactory<Leksem, Integer>(**"id"**));  
 **columnSymbol**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"symbol"**));  
 **columnLeksname**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"leksem"**));  
  
 **tableScanner**.setItems(**collectionLeksemImpl**.getLeksemList());  
  
 updateCountLabel();  
  
  
 }  
 **return ""**;  
 }  
 **private** String message3(**boolean** b) {  
 **if** (b && **searchChar3** == **"/"**) {  
 **txtArea**.setText(**txtArea**.getText().replace(**"/"**, **"$LeksDivide"**));  
 **collectionLeksemImpl**.testData3();  
 **columnId**.setCellValueFactory(**new** PropertyValueFactory<Leksem, Integer>(**"id"**));  
 **columnSymbol**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"symbol"**));  
 **columnLeksname**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"leksem"**));  
  
 **tableScanner**.setItems(**collectionLeksemImpl**.getLeksemList());  
  
 updateCountLabel();  
 }  
  
 **return ""**;  
 }  
 **private** String message4(**boolean** b) {  
 **if** (b && **searchChar4** == **"%"**) {  
 **txtArea**.setText(**txtArea**.getText().replace(**"%"**, **"$LeksRemainder"**));  
 **collectionLeksemImpl**.testData4();  
 **columnId**.setCellValueFactory(**new** PropertyValueFactory<Leksem, Integer>(**"id"**));  
 **columnSymbol**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"symbol"**));  
 **columnLeksname**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"leksem"**));  
  
 **tableScanner**.setItems(**collectionLeksemImpl**.getLeksemList());  
  
 updateCountLabel();  
 }  
 **return ""**;  
 }  
  
 **private** String message5(**boolean** b) {  
 **if** (b && **searchChar5** == **"=="**) {  
 **txtArea**.setText(**txtArea**.getText().replace(**"=="**, **"$LeksCompare"**));  
 **collectionLeksemImpl**.testData5();  
 **columnId**.setCellValueFactory(**new** PropertyValueFactory<Leksem, Integer>(**"id"**));  
 **columnSymbol**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"symbol"**));  
 **columnLeksname**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"leksem"**));  
  
 **tableScanner**.setItems(**collectionLeksemImpl**.getLeksemList());  
  
 updateCountLabel();  
 }  
 **return ""**;  
 }  
  
  
  
 **private void** updateCountLabel()  
 {  
 **labelCount**.setText(**"Количество записей: "** + **collectionLeksemImpl**.getLeksemList().size());  
 }  
**public void** showDialog(ActionEvent actionEvent) {  
 **try** {  
 Stage stage = **new** Stage();  
 Parent root = FXMLLoader.*load*(getClass().getResource(**"edit.fxml"**));  
 stage.setTitle(**"Scanner"**);  
 stage.setMinWidth(300);  
 stage.setMinHeight(150);  
 stage.setResizable(**false**);  
 stage.setScene(**new** Scene(root));  
 stage.initModality(Modality.***APPLICATION\_MODAL***);  
 stage.showAndWait();  
 } **catch** (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
  
 **public void** deleteText(ActionEvent actionEvent) {  
  
 initialize();  
 **txtArea**.setText(**""**);  
 **collectionLeksemImpl**.testData7();  
  
  
 **columnId**.setCellValueFactory(**new** PropertyValueFactory<Leksem, Integer>(**"id"**));  
 **columnSymbol**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"symbol"**));  
 **columnLeksname**.setCellValueFactory(**new** PropertyValueFactory<Leksem, String>(**"leksem"**));  
  
 **tableScanner**.setItems(**collectionLeksemImpl**.getLeksemList());  
  
 updateCountLabel();  
 }  
}

LeksemInterface.java – интерфейс коллекций

**package** sample;  
**public interface** LeksemInterface  
{  
 **void** add(Leksem leksem);  
  
 **void** update(Leksem leksem);  
  
 **void** delete(Leksem leksem);  
}

sample.fxml – форма главного окна

*<?***xml version="1.0" encoding="UTF-8"***?>  
  
<?***import java.lang.\****?>  
<?***import javafx.geometry.\****?>  
<?***import javafx.scene.control.\****?>  
<?***import javafx.scene.layout.\****?>  
<?***import javafx.geometry.Insets***?>  
<?***import javafx.scene.control.Button***?>  
<?***import javafx.scene.control.Label***?>  
<?***import javafx.scene.control.TableColumn***?>  
<?***import javafx.scene.control.TableView***?>  
<?***import javafx.scene.control.TextArea***?>  
<?***import javafx.scene.layout.AnchorPane***?>  
<?***import javafx.scene.layout.HBox***?>  
<?***import javafx.scene.layout.VBox***?>*<**VBox minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="550.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller"**>  
 <**children**>  
 <**AnchorPane prefHeight="200.0" prefWidth="200.0"**>  
 <**children**>  
 <**TextArea fx:id="txtArea" prefHeight="200.0" prefWidth="510.0" text="" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"** />  
 </**children**>  
 <**VBox.margin**>  
 <**Insets bottom="20.0" left="20.0" right="20.0" top="20.0"** />  
 </**VBox.margin**>  
 </**AnchorPane**>  
 <**AnchorPane prefHeight="30.0" prefWidth="200.0"**>  
 <**children**>  
 <**HBox prefHeight="15.0" prefWidth="500.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0"**>  
 <**children**>  
 <**Button fx:id="btnScan" mnemonicParsing="false" onAction="#scanText" prefHeight="25.0" prefWidth="115.0" text="Сканировать"**>  
 <**HBox.margin**>  
 <**Insets bottom="20.0" left="50.0" right="20.0" top="20.0"** />  
 </**HBox.margin**>  
 </**Button**>  
 <**Button fx:id="btnDelete" mnemonicParsing="false" onAction="#deleteText" prefHeight="25.0" prefWidth="115.0" text="Очистить"**>  
 <**HBox.margin**>  
 <**Insets bottom="20.0" left="20.0" right="20.0" top="20.0"** />  
 </**HBox.margin**>  
 </**Button**>  
 </**children**>  
 </**HBox**>  
 </**children**>  
 <**VBox.margin**>  
 <**Insets** />  
 </**VBox.margin**>  
 </**AnchorPane**>  
 <**AnchorPane prefHeight="200.0" prefWidth="200.0" VBox.vgrow="ALWAYS"**>  
 <**children**>  
 <**TableView fx:id="tableScanner" maxHeight="1000.0" prefHeight="200.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0"**>  
 <**columns**>  
 <**TableColumn fx:id="columnId" prefWidth="132.0" text="Идентификатор"** />  
 <**TableColumn fx:id="columnSymbol" minWidth="0.0" prefWidth="150.0" text="Символ"** />  
 <**TableColumn fx:id="columnLeksname" prefWidth="217.0" text="Лексическое имя"** />  
 </**columns**>  
 <**columnResizePolicy**>  
 <**TableView fx:constant="CONSTRAINED\_RESIZE\_POLICY"** />  
 </**columnResizePolicy**>  
 </**TableView**>  
 </**children**>  
 <**VBox.margin**>  
 <**Insets bottom="20.0" left="20.0" right="20.0" top="20.0"** />  
 </**VBox.margin**>  
 </**AnchorPane**>  
 <**AnchorPane prefHeight="50.0" prefWidth="200.0"**>  
 <**children**>  
 <**Label fx:id="labelCount" layoutY="-8.0" text="Количество записей: " AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0"** />  
 </**children**>  
 <**VBox.margin**>  
 <**Insets bottom="20.0" left="20.0" right="20.0" top="20.0"** />  
 </**VBox.margin**>  
 </**AnchorPane**>  
 </**children**>  
</**VBox**>

СollectionLeksem.java – реализация интерфейса коллекций

**package** sample;  
  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
  
**import** java.util.ArrayList;  
*// класс реализовывает интерфейс с помошью коллекции***public class** CollectionLeksem **implements** LeksemInterface  
{  
 **private** ObservableList<Leksem> **leksemList** = FXCollections.*observableArrayList*();  
  
  
 **public** ObservableList<Leksem> getLeksemList()  
 {  
 **return leksemList**;  
 }  
  
 @Override  
 **public void** add(Leksem leksem) {  
 **leksemList**.add(leksem);  
 }  
  
 @Override  
 **public void** update(Leksem leksem) {  
  
 }  
  
 @Override  
 **public void** delete(Leksem leksem) {  
 **leksemList**.remove(leksem);  
 }  
  
 **int n** = 0;  
 **int n1** = 0;  
 **int n2** = 0;  
 **int n3** = 0;  
 **int n4** = 0;  
 **int n5** = 0;  
  
 Leksem **leksem** = **new** Leksem();  
  
 **public void** testData()  
 {  
 Leksem leksem = **new** Leksem();  
 */\*Leksem leksem = new Leksem();  
 leksem.setId(leksem.hashCode());  
 leksem.setLeksem("$LeksPlus");  
 leksem.setSymbol("+");  
  
 leksemList.add(leksem);  
\*/  
 //if(n == 0)* **leksemList**.add(**new** Leksem(leksem.hashCode(),**"+"**,**"$LeksPlus"**));  
  
 **n** = 1;  
  
 */\* System.out.println(leksem.getLeksem());  
 System.out.println(leksem.getId());  
 System.out.println(leksem.getSymbol());\*/* }  
  
 **public void** testData1()  
 {  
 Leksem leksem = **new** Leksem();  
 */\*Leksem leksem = new Leksem();  
 leksem.setId(leksem.hashCode());  
 leksem.setLeksem("$LeksMinus");  
 leksem.setSymbol("-");  
  
 leksemList.add(leksem);\*/  
  
 //if(n1 == 0)* **leksemList**.add(**new** Leksem(leksem.hashCode(),**"-"**,**"$LeksMinus"**));  
 **n1** = 1;  
 }  
  
 **public void** testData2()  
 {  
  
 Leksem leksem = **new** Leksem();  
 *// if(n2 == 0)* **leksemList**.add(**new** Leksem(leksem.hashCode(),**"\*"**,**"$LeksMultiply"**));  
 **n2** = 1;  
 }  
 **public void** testData3()  
 {  
 Leksem leksem = **new** Leksem();  
 *//if(n3 == 0)* **leksemList**.add(**new** Leksem(leksem.hashCode(),**"/"**,**"$LeksDivide"**));  
 **n3** = 1;  
 }  
 **public void** testData4()  
 {  
 Leksem leksem = **new** Leksem();  
 *// if(n4 == 0)* **leksemList**.add(**new** Leksem(leksem.hashCode(),**"%"**,**"$LeksRemainder"**));  
 **n4** = 1;  
 }  
 **public void** testData5()  
 {  
  
 Leksem leksem = **new** Leksem();  
 *// if(n5 == 0)* **leksemList**.add(**new** Leksem(leksem.hashCode(),**"=="**,**"$LeksCompare"**));  
 **n5** = 1;  
 }  
  
 **public void** testData7()  
 {  
 **leksemList**.remove(0,**leksemList**.size());  
 }  
  
}

Leksem.java – заполнения данных в главной форме

**package** sample;  
  
**public class** Leksem  
{  
 **private int id**;  
 **private** String **symbol**;  
 **private** String **leksem**;  
  
 **public** Leksem(**int** id, String symbol, String leksem) {  
 **this**.**id** = id;  
 **this**.**symbol** = symbol;  
 **this**.**leksem** = leksem;  
 }  
  
 **public** Leksem() {  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getSymbol() {  
 **return symbol**;  
 }  
  
 **public void** setSymbol(String symbol) {  
 **this**.**symbol** = symbol;  
 }  
  
 **public** String getLeksem() {  
 **return leksem**;  
 }  
  
 **public void** setLeksem(String leksem) {  
 **this**.**leksem** = leksem;  
 }  
}

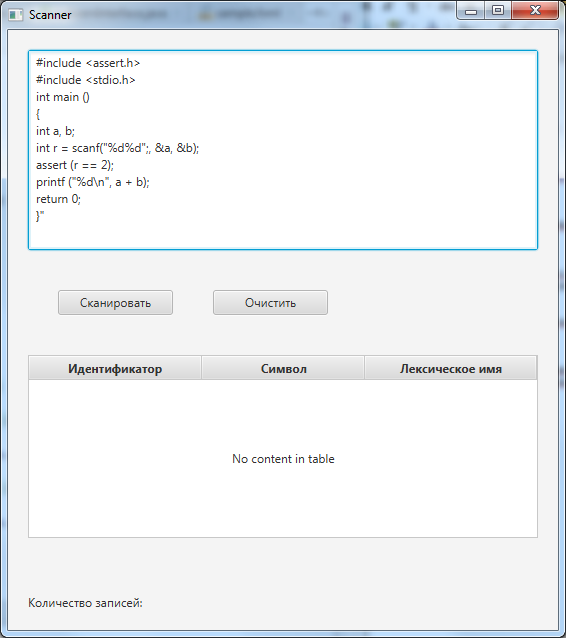


Рис. 1 – главная форма программы

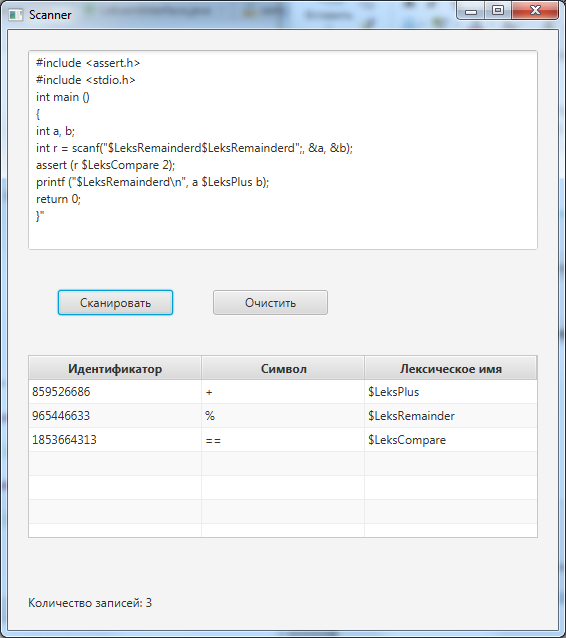


Рис 2. Операция сканирования

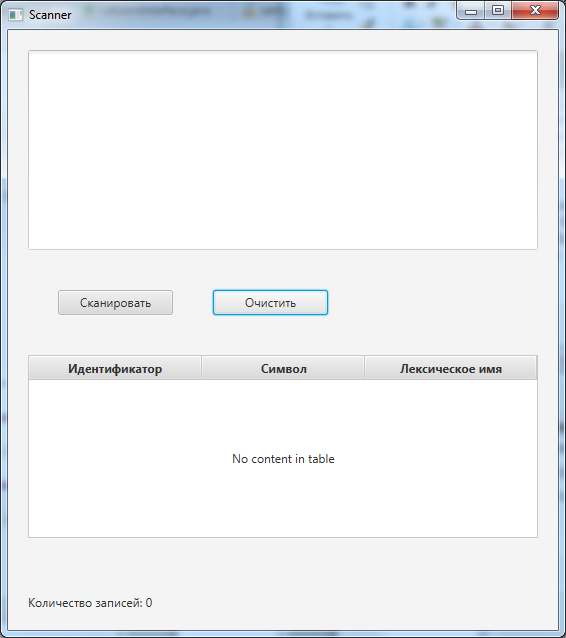


Рис 2. Операция очистки

Извините, в программе присутствует баг, при повторном добавлении символов в код программы, без очистки, они заносятся в таблицу.