МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра інформаційних

систем та мереж

Звіт з лабораторної роботи №1

з дисципліни "Технології програмування в розподілених інформаційних системах"

на тему:

**"Ознайомлення із засобами програмування розподілених систем**"

Варіант №24

Виконав:

Студент групи КН-38

Стасишин О.З.

Прийняв:

доцент каф. ІСМ, к.т.н.

Щербак С. С.

ЛЬВІВ – 2018

**Мета роботи:** набуття навичок програмування розподілених систем.

**Завдання на роботу:**

Вибрати одну з мов програмування С#/Java/Python/JS(NodeJS) та сервер баз даних (Firebase, Mysql, Ms Sql, Postgress). Ознайомитися з основними типами даних вибраної мови програмування та реалізувати веб-орієнтований кліент-серверний додаток – калькулятор простих чисел з підтримкою зворотного польського запису. Додати можливість поелементного +,-,/,\* векторів та таблиць зі збереження результатів обчислень в обраній БД. Розробити методи для забезпечення доступу (читання-запис) до елементів матриці. Обмеження на дані в матрицях згідно варіанту індивідуального завдання.

**Індивідуальне завдання:**



**ControllerMatrixCalculate.java**

package MatrixCalculate;  
  
import javafx.application.Application;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.Initializable;  
import javafx.scene.control.CheckBox;  
import javafx.scene.control.ChoiceBox;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
  
import java.net.URL;  
import java.util.ResourceBundle;  
  
public class ControllerMatrixCalculate implements Initializable {  
  
 ObservableList list = FXCollections.*observableArrayList*();  
  
 @FXML  
 private ChoiceBox<String> action;  
 @FXML  
 private TextField element11 , element12 , element13 , element14;  
 @FXML  
 private TextField element21 , element22 , element23 , element24;  
 @FXML  
 private TextField element31 , element32 , element33 , element34;  
 @FXML  
 private TextField element41 , element42 , element43 , element44;  
  
 @FXML  
 private TextField elementMatrix11 , elementMatrix12 , elementMatrix13 , elementMatrix14;  
 @FXML  
 private TextField elementMatrix21 , elementMatrix22 , elementMatrix23 , elementMatrix24;  
 @FXML  
 private TextField elementMatrix31 , elementMatrix32 , elementMatrix33 , elementMatrix34;  
 @FXML  
 private TextField elementMatrix41 , elementMatrix42 , elementMatrix43 , elementMatrix44;  
  
 @FXML  
 private Label label11,label12,label13,label14,label21,label22,label23,label24,label31,label32,label33,  
 label34,label41,label42,label43,label44;  
  
 int [][] matrixOne = new int[4][4];  
 int [][] matrixTwo = new int[4][4];  
  
 public void getResultMatrix(int [][] resultMatrix){  
 label11.setText(resultMatrix[0][0] + "");  
 label12.setText(resultMatrix[0][1] + "");  
 label13.setText(resultMatrix[0][2] + "");  
 label14.setText(resultMatrix[0][3] + "");  
 label21.setText(resultMatrix[1][0] + "");  
 label22.setText(resultMatrix[1][1] + "");  
 label23.setText(resultMatrix[1][2] + "");  
 label24.setText(resultMatrix[1][3] + "");  
 label31.setText(resultMatrix[2][0] + "");  
 label32.setText(resultMatrix[2][1] + "");  
 label33.setText(resultMatrix[2][2] + "");  
 label34.setText(resultMatrix[2][3] + "");  
 label41.setText(resultMatrix[3][0] + "");  
 label42.setText(resultMatrix[3][1] + "");  
 label43.setText(resultMatrix[3][2] + "");  
 label44.setText(resultMatrix[3][3] + "");  
 DataBaseMatrixCalculate dataBaseMatrixCalculate = new DataBaseMatrixCalculate();  
 dataBaseMatrixCalculate.addDataBase(matrixOne , matrixTwo , resultMatrix);  
 }  
  
 public int[][] getMatrixOne() {  
  
 /\*for (int i = 0; i < 3; i++) {  
 matrixOne[0][i] = Integer.parseInt(arrayMatrixFirstOne[i].getText());  
 matrixOne[1][i] = Integer.parseInt(arrayMatrixFirstTwo[i].getText());  
 matrixOne[2][i] = Integer.parseInt(arrayMatrixFirstThree[i].getText());  
 matrixOne[3][i] = Integer.parseInt(arrayMatrixFirstFour[i].getText());  
 }\*/  
 matrixOne[0][0] = Integer.*parseInt*(element11.getText());  
 matrixOne[0][1] = Integer.*parseInt*(element12.getText());  
 matrixOne[0][2] = Integer.*parseInt*(element13.getText());  
 matrixOne[0][3] = Integer.*parseInt*(element14.getText());  
 matrixOne[1][0] = Integer.*parseInt*(element21.getText());  
 matrixOne[1][1] = Integer.*parseInt*(element22.getText());  
 matrixOne[1][2] = Integer.*parseInt*(element23.getText());  
 matrixOne[1][3] = Integer.*parseInt*(element24.getText());  
 matrixOne[2][0] = Integer.*parseInt*(element31.getText());  
 matrixOne[2][1] = Integer.*parseInt*(element32.getText());  
 matrixOne[2][2] = Integer.*parseInt*(element33.getText());  
 matrixOne[2][3] = Integer.*parseInt*(element34.getText());  
 matrixOne[3][0] = Integer.*parseInt*(element41.getText());  
 matrixOne[3][1] = Integer.*parseInt*(element42.getText());  
 matrixOne[3][2] = Integer.*parseInt*(element43.getText());  
 matrixOne[3][3] = Integer.*parseInt*(element44.getText());  
  
  
  
 return matrixOne;  
 }  
  
 public int[][] getMatrixTwo() {  
  
 /\*for (int i = 0; i < 4; i++) {  
 matrixTwo[0][i] = Integer.parseInt(arrayMatrixSecondOne[i].getText());  
 matrixTwo[1][i] = Integer.parseInt(arrayMatrixSecondTwo[i].getText());  
 matrixTwo[2][i] = Integer.parseInt(arrayMatrixSecondThree[i].getText());  
 matrixTwo[3][i] = Integer.parseInt(arrayMatrixSecondFour[i].getText());  
 }\*/  
 matrixTwo[0][0] = Integer.*parseInt*(elementMatrix11.getText());  
 matrixTwo[0][1] = Integer.*parseInt*(elementMatrix12.getText());  
 matrixTwo[0][2] = Integer.*parseInt*(elementMatrix13.getText());  
 matrixTwo[0][3] = Integer.*parseInt*(elementMatrix14.getText());  
 matrixTwo[1][0] = Integer.*parseInt*(elementMatrix21.getText());  
 matrixTwo[1][1] = Integer.*parseInt*(elementMatrix22.getText());  
 matrixTwo[1][2] = Integer.*parseInt*(elementMatrix23.getText());  
 matrixTwo[1][3] = Integer.*parseInt*(elementMatrix24.getText());  
 matrixTwo[2][0] = Integer.*parseInt*(elementMatrix31.getText());  
 matrixTwo[2][1] = Integer.*parseInt*(elementMatrix32.getText());  
 matrixTwo[2][2] = Integer.*parseInt*(elementMatrix33.getText());  
 matrixTwo[2][3] = Integer.*parseInt*(elementMatrix34.getText());  
 matrixTwo[3][0] = Integer.*parseInt*(elementMatrix41.getText());  
 matrixTwo[3][1] = Integer.*parseInt*(elementMatrix42.getText());  
 matrixTwo[3][2] = Integer.*parseInt*(elementMatrix43.getText());  
 matrixTwo[3][3] = Integer.*parseInt*(elementMatrix44.getText());  
 return matrixTwo;  
 }  
  
 private void loadData(){  
 list.removeAll(list);  
 String a = "+";  
 String b = "-";  
 String c = "\*";  
 list.addAll(a , b , c);  
 action.getItems().addAll(list);  
 }  
  
 public void initialize(URL location, ResourceBundle resources) {  
 loadData();  
 }  
  
 private int[][] addition(int [][] matrixOne , int [][] matrixTwo){  
 int [][] resulMatrix = new int[4][4];  
 for (int i = 0; i < 4; i++) {  
 for (int j = 0; j < 4; j++) {  
 resulMatrix[i][j] = matrixOne[i][j] + matrixTwo[i][j];  
 }  
 }  
 return resulMatrix;  
 }  
 private int[][] sub(int [][] matrixOne , int [][] matrixTwo){  
 int [][] resulMatrix = new int[4][4];  
 for (int i = 0; i < 4; i++) {  
 for (int j = 0; j < 4; j++) {  
 resulMatrix[i][j] = matrixOne[i][j] - matrixTwo[i][j];  
 }  
 }  
  
 return resulMatrix;  
 }  
 private int[][] mult(int [][] matrixOne , int [][] matrixTwo){  
 int [][] resulMatrix = new int[4][4];  
 for (int i = 0; i < 4; i++) {  
 for (int j = 0; j < 4; j++) {  
 for (int k = 0; k < 4; k++) {  
 resulMatrix[i][j] += matrixOne[i][k] \* matrixTwo[k][j];  
 }  
 }  
 }  
 return resulMatrix;  
 }  
  
  
 public void сalculation(ActionEvent actionEvent) {  
 String calculateType = action.getValue();  
 if(calculateType.equals("+"))  
 getResultMatrix(addition(getMatrixOne() ,getMatrixTwo()));  
 if(calculateType.equals("-"))  
 getResultMatrix(sub(getMatrixOne() ,getMatrixTwo()));  
 if(calculateType.equals("\*"))  
 getResultMatrix(mult(getMatrixOne() ,getMatrixTwo()));  
  
 }  
}

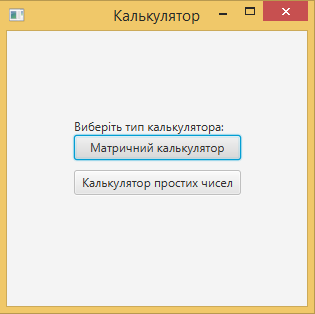
**DataBaseMatrixCalculate.java**

package MatrixCalculate;  
  
import com.google.api.core.ApiFuture;  
import com.google.auth.oauth2.GoogleCredentials;  
import com.google.cloud.firestore.DocumentReference;  
import com.google.cloud.firestore.Firestore;  
import com.google.cloud.firestore.WriteResult;  
import com.google.firebase.FirebaseApp;  
import com.google.firebase.FirebaseOptions;  
import com.google.firebase.cloud.FirestoreClient;  
  
import java.io.FileInputStream;  
import java.io.InputStream;  
import java.text.DateFormat;  
import java.util.Date;  
import java.util.LinkedHashMap;  
import java.util.Map;  
  
public class DataBaseMatrixCalculate {  
 private DocumentReference documentReference;  
 Firestore db;  
 DataBaseMatrixCalculate(){  
 try {  
 InputStream serviceAccount = new FileInputStream("C:\\Users\\User\\Downloads\\TPRIS-fff677a0ea61.json");  
 GoogleCredentials credentials = GoogleCredentials.*fromStream*(serviceAccount);  
 FirebaseOptions options = new FirebaseOptions.Builder()  
 .setCredentials(credentials)  
 .build();  
 FirebaseApp.*initializeApp*(options);  
  
  
 db = FirestoreClient.*getFirestore*();  
 }catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
  
 public void addDataBase(int[][] matrixOne , int[][] matrixTwo , int[][] matrixResult){  
 try {  
  
 Date now = new Date();  
 String stringMatrixOne = "" , stringMatrixTwo = "" , stringMatrixResult = "";  
 for (int i = 0; i < 4; i++) {  
 for (int j = 0; j < 4; j++) {  
 stringMatrixOne += matrixOne[i][j] + " ";  
 stringMatrixTwo += matrixTwo[i][j] + " ";  
 stringMatrixResult += matrixResult[i][j] + " ";  
  
 }  
 stringMatrixOne += "/";  
 stringMatrixTwo += "/";  
 stringMatrixResult += "/";  
 }  
 System.*out*.println(stringMatrixOne);  
 documentReference = db.collection("Matrix calculate").document(DateFormat.*getTimeInstance*().format(now));  
 Map<String , String> data = new LinkedHashMap<String , String>();  
 data.put("Matrix 1" , stringMatrixOne);  
 data.put("Matrix 2" , stringMatrixTwo);  
 data.put("Result" , stringMatrixResult);  
  
 ApiFuture<WriteResult> apiFuture = documentReference.set(data);  
 }catch(Exception e){  
 e.printStackTrace();  
 }  
 }  
}

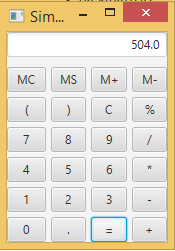
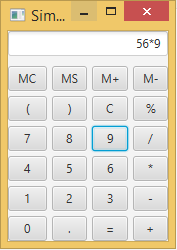
**MatrixCalculate.java**

package MatrixCalculate;  
  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.layout.AnchorPane;  
import javafx.stage.Stage;  
  
public class MatrixCalculate {  
  
 public MatrixCalculate() throws Exception{  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("/MatrixCalculate.fxml"));  
 AnchorPane load = (AnchorPane) loader.load();  
 Stage stage = new Stage();  
 stage.setTitle("MatrixCalculate");  
 Scene scene = new Scene(load);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
}

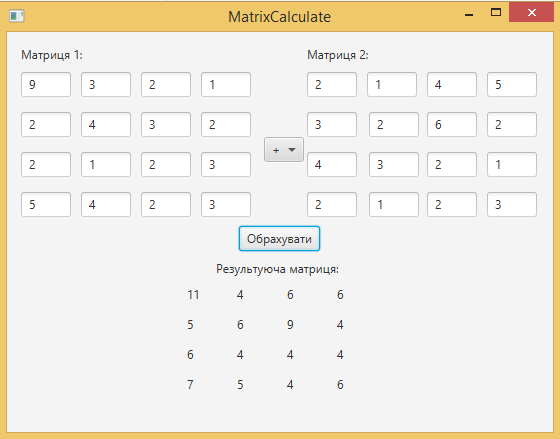
**Результати виконання програми:**



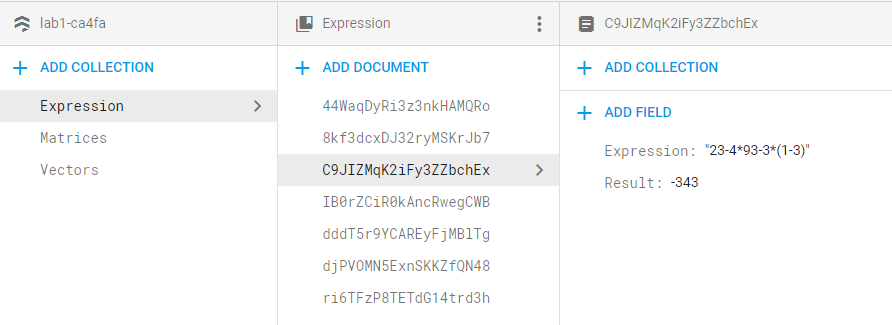
*Рис. 1. Графічний інтерфейс користувача*



*Рис. 2. Обрахування математичного виразу*



*Рис. 3. Обрахування матриць*



*Рис. 5. База даних*

**Висновок:** програма призначена для роботи з матрицями, векторами та математичними виразами. Програма реалізує усі необхідні завдання. На цій лабораторній роботі було набуто навичок програмування розподілених систем. Програма написана на мові Java.