МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Звіт з лабораторної роботи № 3

з предмету «Об'єктно-орієнтоване програмування»

Виконав:

Студент групи КН-36а

Кулик В.В.

Перевірили:

Козуля М.М.

Кізілов О.С.

Харків 2017

### Завдання на лабораторну роботу

### 1.1 Індивідуальне завдання

Необхідно реалізувати мовою Java за допомогою засобів JavaFX застосунок графічного інтерфейсу користувача, в якому здійснюється обробка даних індивідуальних завдань попередніх лабораторних робіт. Головне вікно повинно містити меню, в якому необхідно реалізувати такі функції:

* створення нового набору даних
* завантаження даних з XML-документу для редагування
* зберігання змінених даних в XML-документі
* пошук за ознаками, визначеними в [лабораторній роботі № 5 курсу "Алгоритмізація та програмування" (друга частина)](http://iwanoff.96.lt/algorithmization_2/LabTraining05.html)
* здійснення сортування за ознаками, визначеними в [лабораторній роботі № 6 курсу "Алгоритмізація та програмування" (друга частина)](http://iwanoff.96.lt/algorithmization_2/LabTraining06.html)
* отримання вікна "Про програму" з даними про програму і автора.

У лівій частині вікна слід розташувати рядки для введення скалярних даних, область відображення для результатів пошуку, а також кнопки, які забезпечують виконання основних функцій програми. В середній частині вікна слід розташувати таблицю для відображення та редагування даних.

Розробці програми графічного інтерфейсу користувача повинне передувати створення консольного застосунку, в якому здійснюється читання даних з підготовленого XML-документу, пошук певних даних, сортування і зберігання у новому XML-документі. Для роботи з XML скористатися технологією зв'язування даних.

### 1.2 Перелік для опису місяців року

Створити перелік "Місяць". Необхідно визначати у конструкторі і зберігати кількість днів (для невисокосного року). Додати методи отримання попереднього та наступного місяця, а також функцію, яка повертає сезон для кожного місяця. Передбачити виведення місяців українською (російською) мовою. Створити статичну функцію виведення даних про усі місяці. Протестувати переліку в функції main() тестового класу.

### 1.3 Дані про користувачів

Представити дані про користувачів у вигляді асоціативного масиву (ім'я / пароль) з припущенням, що всі імена користувачів різні. Вивести дані про користувачів з довжиною пароля більше 6 символів.

### 1.4 Міні-калькулятор

Створити застосунок графічного інтерфейсу користувача, в якому після введення чисел у двох рядках типу TextField виконується одна з чотирьох арифметичних дій (залежно від вибраної кнопки RadioButton). Результат виводиться в інше текстове поле.

### 1.5 Словник (додаткове завдання)

Розробити програму графічного інтерфейсу користувача перегляду слів невеличкого англо-українського (англо-російського) словника. Реалізувати функції пошуку слова, додавання нових слів.

**Хід роботи**

### 1.1 Індивідуальне завдання

*Код програми 1.1:*

**Doc.java**

package seclab.f.ex;

public abstract class Doc {

private String name;

private String spec;

public Doc() {

}

public Doc(String name, String spec) {

this.name = name;

this.spec = spec;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSpec() {

return spec;

}

public void setSpec(String spec) {

this.spec = spec;

}

abstract public Recept minOfVisitors();

abstract public int sumOfVisitors();

abstract public void addRecept(Recept recept);

abstract public void sortByVisitors();

abstract public void clearRecept();

abstract public void sortByDay();

abstract public Recept getRecept(int i);

abstract public int receptCount();

}

**DocArr.java**

package seclab.f.ex;

import java.util.Arrays;

public class DocArr extends Doc {

private Recept[] recepts;

public DocArr() {

// TODO Auto-generated constructor stub

}

public DocArr(String name,String spec, Recept[] recepts) {

super(name, spec);

this.recepts = recepts;

}

public void setRecepts(Recept[] recepts) {

this.recepts = recepts;

}

public Recept[] getRecepts() {

return recepts;

}

*@Override*

public Recept minOfVisitors() {

Recept min = recepts[0];

for (Recept h : recepts) {

if (h.getNumOfVisit() < min.getNumOfVisit())

min = h;

}

return min;

}

*@Override*

public int sumOfVisitors() {

int sum = 0;

for (Recept rec : recepts) {

sum += rec.getNumOfVisit();

}

return sum;

}

*@Override*

public void sortByVisitors() {

Arrays.*sort*(recepts, (a, b) -> Integer.*compare*(a.getNumOfVisit(), b.getNumOfVisit()));

}

*@Override*

public void sortByDay() {

Arrays.*sort*(recepts, (a, b) -> a.getDay().compareTo(b.getDay()));

}

*@Override*

public String toString() {

return "Doctor " + getName() + " " + getSpec() + "\n" + Arrays.*asList*(recepts);

}

*@Override*

public int receptCount() {

// TODO Auto-generated method stub

return 0;

}

*@Override*

public Recept getRecept(int i) {

return recepts[i];

}

*@Override*

public void addRecept(Recept recept) {

// TODO Auto-generated method stub

}

*@Override*

public void clearRecept() {

// TODO Auto-generated method stub

}

}

**DocArrTXT.java**

package seclab.f.ex;

import java.io.FileReader;

import java.io.FileWriter;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class DocArrTXT extends DocArr {

public DocArrTXT() {

}

public DocArrTXT(String name, String spec, Recept[] recepts) {

super(name, spec, recepts);

}

public void toTxt(String name) {

try (FileWriter fReader = new FileWriter(name)) {

fReader.write(this.getName() + "\n");

fReader.write(this.getSpec() + "\n");

for (Recept rec : getRecepts()) {

fReader.write(rec.toString() + "\n");

}

} catch (Exception e) {

// TODO: handle exception

}

}

public DocArrTXT fromTxt() {

DocArrTXT sArr = new DocArrTXT();

try (Scanner scanner = new Scanner(new FileReader("Doctor.txt"))) {

sArr.setName(scanner.nextLine());

sArr.setSpec(scanner.nextLine());

List<Recept> sList = new ArrayList<>();

while (scanner.hasNext()) {

String[] string = scanner.nextLine().split(",");

Integer integer = Integer.parseInt(string[2].replaceAll(" ", ""));

sList.add(new Recept(integer, string[0], string[1]));

}

sArr.setRecepts(sList.toArray(new Recept[sList.size()]));

} catch (Exception e) {

e.printStackTrace();

}

return sArr;

}

}

**DocList.java**

package seclab.f.ex;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class DocList extends Doc{

private List<Recept> list;

public DocList() {

}

public DocList(String name, String spec, List<Recept> list) {

super(name, spec);

this.list = list;

}

public List<Recept> getList() {

return list;

}

public void setList(List<Recept> list) {

this.list = list;

}

@Override

public Recept minOfVisitors() {

Recept h = list.get(0);

for (Recept spec : list) {

if (h.getNumOfVisit() > spec.getNumOfVisit())

h = spec;

}

return h;

}

@Override

public int sumOfVisitors() {

int sum = 0;

for (Recept spec : list) {

sum += spec.getNumOfVisit();

}

return sum;

}

@Override

public void sortByVisitors() {

Collections.sort(list, (a, b) -> Integer.compare(a.getNumOfVisit(), b.getNumOfVisit()));

}

@Override

public void sortByDay() {

Collections.sort(list, (a, b) -> b.getDay().compareTo(a.getDay()));

}

@Override

public String toString() {

return "Doctor " + getName() + " " + getSpec() + "\n" + Arrays.asList(list.toArray());

}

@Override

public int receptCount() {

// TODO Auto-generated method stub

return 0;

}

@Override

public Recept getRecept(int i) {

// TODO Auto-generated method stub

return null;

}

@Override

public void addRecept(Recept recept) {

}

@Override

public void clearRecept() {

}

}

**DocListXML.java**

package seclab.f.ex;

import java.io.File;

import java.io.FileWriter;

import java.util.ArrayList;

import java.util.List;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.Marshaller;

import javax.xml.bind.Unmarshaller;

public class DocListXML extends DocList{

private generated.Doc doc;

public DocListXML() {

}

public DocListXML(String name, String spec, List<Recept> list) {

super(name, spec, list);

}

public DocListXML fromXml(String filename) {

try {

JAXBContext jaxbContext=JAXBContext.newInstance(generated.Doc.class);

Unmarshaller unmarshaller=jaxbContext.createUnmarshaller();

doc =(generated.Doc) unmarshaller.unmarshal(new File("Doctor.xml"));

List<Recept> lis=new ArrayList<>();

for (generated.Doc.Recept recept : doc.getRecept()) {

lis.add(new Recept(recept.getNumOfVisit(),recept.getDay(),recept.getShift()));

}

this.setList(lis);

this.setName(doc.getName());

this.setSpec(doc.getSpec());

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

public void toXml(String name) {

try {

JAXBContext jaxbContext = JAXBContext.newInstance(generated.Doc.class);

Marshaller marshaller = jaxbContext.createMarshaller();

marshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, Boolean.TRUE);

doc.setName(getName());

doc.setSpec(getSpec());

List<generated.Doc.Recept> list=new ArrayList<>();

for (Recept recept : getList()) {

list.add(new generated.Doc.Recept(recept.getNumOfVisit(),recept.getDay(),recept.getShift()));

}

marshaller.marshal(doc, new FileWriter(name));

} catch (Exception e) {

e.printStackTrace();

}

}

}

**Recept.java**

**package** seclab.f.ex;

**public** **class** **Recept** {

**private** **int** numOfVisit;

**private** **String** day;

**private** **String** shift;

**public** **Recept**() {}

**public** **Recept**(**int** numOfVisit, **String** day,**String** shift) {

**this**.numOfVisit = numOfVisit;

**this**.day = day;

**this**.shift = shift;

}

**public** **void** **setDay**(**String** day) {

**this**.day = day;

}

**public** **void** **setNumOfVisit**(**int** numOfVisit) {

**this**.numOfVisit = numOfVisit;

}

**public** **void** **setShift**(**String** shift) {

**this**.shift = shift;

}

**public** **String** **getDay**() {

**return** day;

}

**public** **int** **getNumOfVisit**() {

**return** numOfVisit;

}

**public** **String** **getShift**() {

**return** shift;

}

***@Override***

**public** **String** **toString**() {

**return** day + " , " + shift+","+numOfVisit;

}

}

**Test.java**

**package** seclab.f.ex;

**public** **class** **Test** {

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

**DocArrTXT** **sArr** = **new** DocArrTXT();

sArr = sArr.fromTxt();

sArr.sortByDay();

sArr.toTxt("sortbyDay.txt");

sArr.sortByVisitors();

sArr.toTxt("sortbyVisitors.txt");

**DocListXML** **list** = **new** DocListXML();

list.fromXml("Doctor.xml");

**System**.***out***.println(list);

list.sortByDay();

list.toXml("sortbyDay.xml");

list.sortByVisitors();

list.toXml("sortbyVisitors.xml");

}

}

**ReceptionsFX.java**

package sample;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.scene.layout.BorderPane;

import javafx.stage.Stage;

public class ReceptionsFX extends Application{

private BorderPane recep;

public void start(Stage primaryStage) {

try {

FXMLLoader loader = new FXMLLoader();

loader.setLocation(ReceptionsFX.class.getResource("Recep.fxml"));

recep = (BorderPane) loader.load();

Scene scene = new Scene(recep);

scene.setRoot(recep);

primaryStage.setScene(scene);

primaryStage.setTitle("Информация о докторе");

primaryStage.show();

}

catch(Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

launch(args);

}

}

**ReceptionsController.java**

**package sample;**

**import java.awt.Checkbox;**

**import java.io.File;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.StringTokenizer;**

**import javafx.fxml.FXML;**

**import javafx.scene.control.Alert;**

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

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

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

**import javafx.scene.control.TextArea;**

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

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

**import javafx.scene.control.Alert.AlertType;**

**import javafx.scene.control.CheckBox;**

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

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

**import javafx.stage.FileChooser;**

**import javafx.util.converter.IntegerStringConverter;**

**import seclab.DocListXML;**

**import seclab.Recept;**

**import javafx.application.Platform;**

**import javafx.collections.FXCollections;**

**import javafx.collections.ObservableList;**

**import javafx.event.ActionEvent;**

**public class ReceptionsController {**

**@FXML TextField TextFieldDoctor;**

**@FXML TextField TextFieldSpec;**

**@FXML TextField TextFieldSearch;**

**@FXML TextArea TextAreaResults;@FXML TextArea textAreaAll;**

**@FXML CheckBox CheckBoxch;**

**@FXML TableView<Recept> tableViewReceptions;**

**@FXML TableColumn<Recept, Integer> TableColumnVisitors;**

**@FXML TableColumn<Recept, String> TableColumnShift;**

**@FXML TableColumn<Recept, String> TableColumnDate;**

**private DocListXML doctor = new DocListXML();**

**private ObservableList<Recept> observableList;**

**@FXML**

**public void initialize() {**

**tableViewReceptions.setPlaceholder(new Label(""));**

**}**

**@FXML**

**public void doNew(ActionEvent event) {**

**doctor = new DocListXML();**

**observableList = null;**

**TextFieldDoctor.setText("");**

**TextFieldSpec.setText("");**

**TextFieldSearch.setText("");**

**TextAreaResults.setText("");**

**tableViewReceptions.setItems(null);**

**tableViewReceptions.setPlaceholder(new Label(""));**

**}**

**@FXML**

**public void doOpen(ActionEvent event) {**

**FileChooser fileChooser = getFileChooser("Открыть XML-файл");**

**File file;**

**try {**

**if ((file = fileChooser.showOpenDialog(null)) != null) {**

**doctor.fromXml(file.getCanonicalPath());**

**TextFieldDoctor.setText(doctor.getName());**

**TextFieldSpec.setText(doctor.getSpec());**

**TextAreaResults.setText("");**

**tableViewReceptions.setItems(null);**

**updateTable();**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**@FXML**

**public void doSave(ActionEvent event) {**

**FileChooser fileChooser = getFileChooser("Сохранить XML-файл");**

**File file;**

**if ((file = fileChooser.showSaveDialog(null)) != null) {**

**try {**

**updateSourceData();**

**doctor.toXml(file.getCanonicalPath());**

**showMessage("Результаты успешно сохранены");**

**}**

**catch (Exception e) {**

**showError("Ошибка записи файла");**

**}**

**}**

**}**

**@FXML**

**public void doExit(ActionEvent event) {**

**Platform.exit();**

**}**

**@FXML**

**public void doAdd(ActionEvent event) {**

**if (observableList == null) {**

**updateTable();**

**}**

**observableList.add(new Recept(0,null, null));**

**}**

**@FXML**

**public void doRemove(ActionEvent event) {**

**if (observableList == null) {**

**return;**

**}**

**if (observableList.size() > 0) {**

**observableList.remove(observableList.size() - 1);**

**}**

**if (observableList.size() <= 0) {**

**observableList = null;**

**}**

**}**

**@FXML**

**public void doSortByDay(ActionEvent event) {**

**updateSourceData();**

**doctor.sortByDay();**

**updateTable();**

**}**

**@FXML**

**public void doSortByVisitors(ActionEvent event) {**

**updateSourceData();**

**doctor.sortByVisitors();**

**updateTable();**

**}**

**@FXML**

**public void doAbout(ActionEvent event) {**

**Alert alert = new Alert(AlertType.INFORMATION);**

**alert.setTitle("О программе");**

**alert.setHeaderText("Индивидуальное задание");**

**alert.setContentText("Автор: Владимир Кулик");**

**alert.showAndWait();**

**}**

**@FXML**

**public void nameChanged(ActionEvent event) {**

**doctor.setName(TextFieldDoctor.getText());**

**}**

**@FXML**

**public void specChanged(ActionEvent event) {**

**doctor.setSpec(TextFieldSpec.getText());**

**}**

**@FXML**

**public void doSearchByWord(ActionEvent event) {**

**updateSourceData();**

**TextAreaResults.clear();**

**showResults(doctor.minOfVisitors());**

**}**

**@FXML**

**public void doSearchByAll(ActionEvent event) {**

**updateSourceData();**

**textAreaAll.clear();**

**if(CheckBoxch.isSelected()) {**

**showResult();}**

**}**

**@FXML**

**public void doSearchBySubstring(ActionEvent event) {**

**updateSourceData();**

**TextAreaResults.clear();**

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

**Recept h = doctor.getList().get(i);**

**if (h.getShift().contains(TextFieldSearch.getText())||h.getDay().contains(TextFieldSearch.getText())||Integer.toString(h.getNumOfVisit()).contains(TextFieldSearch.getText())) {**

**showResults(h);**

**}**

**}**

**}**

**private void showResults(Recept recep) {**

**TextAreaResults.appendText("День: " + recep.getDay() + "\n");**

**TextAreaResults.appendText("Кол-во поситителей :" + recep.getNumOfVisit() + "\n");**

**TextAreaResults.appendText("Смена :" + recep.getShift() + "\n");**

**TextAreaResults.appendText("\n");**

**}**

**private void showResult() {**

**textAreaAll.appendText(""+doctor.sumOfVisitors());**

**}**

**private void updateSourceData() {**

**doctor.getList().clear();**

**for (Recept h : observableList) {**

**doctor.getList().add(h);**

**}**

**}**

**private void updateVisitors(CellEditEvent<Recept, Integer> t) {**

**Recept h = (Recept) t.getTableView().getItems().get(t.getTablePosition().getRow());**

**h.setNumOfVisit(t.getNewValue());**

**}**

**private void updateShift(CellEditEvent<Recept, String> t) {**

**Recept h = (Recept) t.getTableView().getItems().get(t.getTablePosition().getRow());**

**h.setShift(t.getNewValue());**

**}**

**private void updateTable() {**

**List<seclab.Recept> sList = new ArrayList<>();**

**observableList = FXCollections.observableList(sList);**

**for (Recept rec : doctor.getList()) {**

**sList.add(rec);**

**}**

**tableViewReceptions.setItems(observableList);**

**TableColumnVisitors.setCellValueFactory(new PropertyValueFactory<>("numOfVisit"));**

**TableColumnVisitors**

**.setCellFactory(TextFieldTableCell.<Recept, Integer>forTableColumn(new IntegerStringConverter()));**

**TableColumnVisitors.setOnEditCommit(t -> updateVisitors(t));**

**TableColumnDate.setCellValueFactory(new PropertyValueFactory<>("day"));**

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

**TableColumnDate.setOnEditCommit(t -> updateSourceData());**

**TableColumnShift.setCellValueFactory(new PropertyValueFactory<>("shift"));**

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

**TableColumnShift.setOnEditCommit(t -> updateShift(t));**

**}**

**public static void showMessage(String message) {**

**Alert alert = new Alert(AlertType.INFORMATION);**

**alert.setTitle("");**

**alert.setHeaderText(message);**

**alert.showAndWait();**

**}**

**public static void showError(String message) {**

**Alert alert = new Alert(AlertType.ERROR);**

**alert.setTitle("Ошибка");**

**alert.setHeaderText(message);**

**alert.showAndWait();**

**}**

**public static FileChooser getFileChooser(String title) {**

**FileChooser fileChooser = new FileChooser();**

**fileChooser.setInitialDirectory(new File("."));**

**fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("XML-файлы (\*.xml)", "\*.xml"));**

**fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Все файлы (\*.\*)", "\*.\*"));**

**fileChooser.setTitle(title);**

**return fileChooser;**

**}**

**private boolean isNumber(String str) {**

**if (str == null || str.isEmpty())**

**return false;**

**for (int i = 0; i < str.length(); i++) {**

**if (!Character.isDigit(str.charAt(i)))**

**return false;**

**}**

**return true;**

**}**

**}**

**Controller.java**

**package sample;**

**public class Controller {**

**}**

**Recep.fxml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<?import javafx.scene.effect.\*?>**

**<?import javafx.scene.text.\*?>**

**<?import javafx.geometry.\*?>**

**<?import javafx.scene.control.\*?>**

**<?import java.lang.\*?>**

**<?import javafx.scene.layout.\*?>**

**<?import javafx.scene.layout.BorderPane?>**

**<BorderPane xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.ReceptionsController">**

**<center>**

**<Pane prefHeight="486.0" prefWidth="643.0" BorderPane.alignment="CENTER">**

**<children>**

**<MenuBar prefHeight="24.0" prefWidth="643.0">**

**<menus>**

**<Menu mnemonicParsing="false" text="File">**

**<items>**

**<MenuItem mnemonicParsing="false" onAction="#doNew" text="New" />**

**<MenuItem mnemonicParsing="false" onAction="#doOpen" onMenuValidation="#doOpen" text="Open..." />**

**<MenuItem mnemonicParsing="false" onAction="#doSave" text="Save..." />**

**<SeparatorMenuItem mnemonicParsing="false" />**

**<MenuItem mnemonicParsing="false" onAction="#doExit" text="Exit" />**

**</items>**

**</Menu>**

**<Menu mnemonicParsing="false" text="Sorts">**

**<items>**

**<MenuItem mnemonicParsing="false" onAction="#doSortByVisitors" text="Sort by visitors" />**

**<MenuItem mnemonicParsing="false" onAction="#doSortByDay" text="Sort by days" />**

**</items>**

**</Menu>**

**<Menu mnemonicParsing="false" text="Help">**

**<items>**

**<MenuItem mnemonicParsing="false" onAction="#doAbout" text="About App" />**

**</items>**

**</Menu>**

**</menus>**

**</MenuBar>**

**<AnchorPane layoutY="24.0" prefHeight="463.0" prefWidth="239.0">**

**<children>**

**<Label layoutX="16.0" layoutY="155.0" text="Search:" AnchorPane.topAnchor="135.0">**

**<font>**

**<Font />**

**</font>**

**</Label>**

**<TextField fx:id="TextFieldSearch" layoutX="15.0" layoutY="160.0" prefHeight="24.0" prefWidth="196.0" />**

**<Button layoutX="17.0" layoutY="385.0" mnemonicParsing="false" onAction="#doSearchByWord" prefHeight="24.0" prefWidth="205.0" text="Search minimum number of visits" textAlignment="CENTER">**

**<font>**

**<Font size="10.0" />**

**</font>**

**<effect>**

**<Blend />**

**</effect>**

**</Button>**

**<Button layoutX="91.0" layoutY="194.0" mnemonicParsing="false" onAction="#doSearchBySubstring" prefHeight="24.0" prefWidth="51.0" text="Search" textAlignment="CENTER">**

**<font>**

**<Font size="10.0" />**

**</font>**

**<effect>**

**<Blend />**

**</effect>**

**</Button>**

**<TextArea fx:id="TextAreaResults" layoutX="11.0" layoutY="211.0" prefHeight="164.0" prefWidth="173.0" AnchorPane.bottomAnchor="88.0" AnchorPane.leftAnchor="11.0" AnchorPane.rightAnchor="10.0" AnchorPane.topAnchor="228.0" />**

**<TextField id="TextFieldDoctor" fx:id="TextFieldDoctor" layoutX="14.0" layoutY="38.0" onAction="#nameChanged" prefHeight="24.0" prefWidth="198.0" />**

**<TextField fx:id="TextFieldSpec" layoutX="14.0" layoutY="99.0" onAction="#specChanged" prefHeight="24.0" prefWidth="198.0" />**

**<Label layoutX="14.0" layoutY="14.0" text="Doctor:" />**

**<Label layoutX="14.0" layoutY="75.0" text="Specialization:" />**

**<CheckBox fx:id="CheckBoxch" layoutX="24.0" layoutY="428.0" mnemonicParsing="false" onAction="#doSearchByAll" text="Total Visitors" />**

**<TextArea fx:id="textAreaAll" layoutX="141.0" layoutY="418.0" prefHeight="37.0" prefWidth="71.0" />**

**</children>**

**</AnchorPane>**

**<TableView fx:id="tableViewReceptions" layoutX="241.0" layoutY="24.0" prefHeight="473.0" prefWidth="405.0">**

**<columns>**

**<TableColumn fx:id="TableColumnDate" editable="true" prefWidth="105.0" text="Date" />**

**<TableColumn fx:id="TableColumnVisitors" editable="true" prefWidth="153.0" text="Number of visits" />**

**<TableColumn fx:id="TableColumnShift" editable="true" prefWidth="146.0" text="Shift" />**

**</columns>**

**</TableView>**

**</children>**

**</Pane>**

**</center>**

**</BorderPane>**

*Файл Doctor.xml містить такі дані:*

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<Doc name="Kurchenko Sergey" spec="Family doctor">

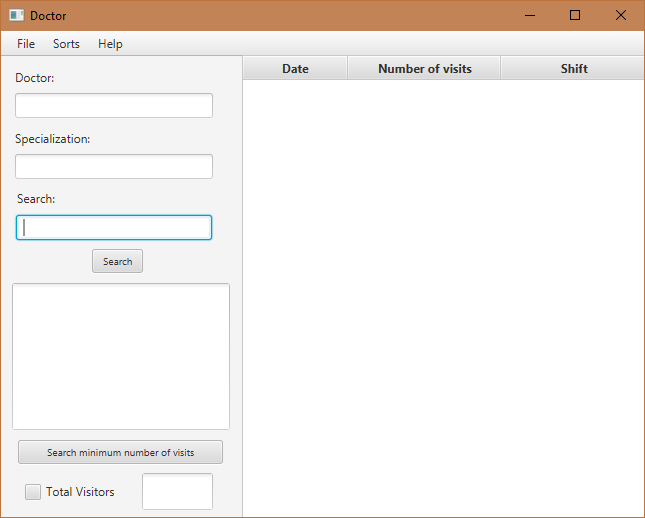
<Recept day="15.10.15" numOfVisit="30" shift="ночная смена"></Recept>

<Recept day="13.10.15" numOfVisit="18" shift="дневная смена"></Recept>

<Recept day="05.10.15" numOfVisit="15" shift="утренняя смена"></Recept>

</Doc>

*Після запуску програми відкривається наступне вікно:*



### 1.2 Перелік для опису місяців року

*Код програми 1.2:*

**Month.java**

**package Describe;**

**enum Month {**

**JANUARY(31),**

**FEBRUARY(28),**

**MARCH(31),**

**APRIL(30),**

**MAY(31),**

**JUNE(30),**

**JULY(31),**

**AUGUST(31),**

**SEPTEMBER(30),**

**OCTOBER(31),**

**NOVEMBER(30),**

**DECEMBER(31);**

**private int daysCount;**

**private Month(int daysCount){**

**this.daysCount=daysCount;**

**}**

**double getMonth() {**

**return daysCount;**

**}**

**@Override**

**public String toString() {**

**return "Month: "+name()+"\nDays in month: " + daysCount;**

**}**

**Month next() {**

**Month month = values()[(ordinal() + 1) % values().length];**

**return month;**

**}**

**Month previous() {**

**Month month = values()[(ordinal()-1) % values().length];**

**return month;**

**}**

**public String toStr() {**

**switch (this) {**

**case JANUARY:**

**return "Январь"+"\nДней в месяце: " + daysCount;**

**case FEBRUARY:**

**return "Февраль"+"\nДней в месяце: " + daysCount;**

**case MARCH:**

**return "Март"+"\nДней в месяце: " + daysCount;**

**case APRIL:**

**return "Апрель"+"\nДней в месяце: " + daysCount;**

**case MAY:**

**return "Май"+"\nДней в месяце: " + daysCount;**

**case JUNE:**

**return "Июнь"+"\nДней в месяце: " + daysCount;**

**case JULY:**

**return "Июль"+"\nДней в месяце: " + daysCount;**

**case AUGUST:**

**return "Август"+"\nДней в месяце: " + daysCount;**

**case SEPTEMBER:**

**return "Сентябрь"+"\nДней в месяце: " + daysCount;**

**case OCTOBER:**

**return "Октябрь"+"\nДней в месяце: " + daysCount;**

**case NOVEMBER:**

**return "Ноябрь"+"\nДней в месяце: " + daysCount;**

**case DECEMBER:**

**return "Декабрь"+"\nДней в месяце: " + daysCount;**

**}**

**return "Месяцы закончились!";**

**}**

**@SuppressWarnings("incomplete-switch")**

**String Season(){**

**String s="";**

**switch (this) {**

**case DECEMBER:**

**case JANUARY:**

**case FEBRUARY:**

**return s="Winter";}**

**switch(this){**

**case MARCH:**

**case MAY:**

**case APRIL:**

**return s="Spring";**

**}**

**switch(this){**

**case JUNE:**

**case JULY:**

**case AUGUST:**

**return s="Summer";**

**}**

**switch(this){**

**case SEPTEMBER:**

**case NOVEMBER:**

**case OCTOBER:**

**return s="Autumn";**

**}**

**return s;**

**}**

**@SuppressWarnings("incomplete-switch")**

**String SeasonRus(){**

**String s="";**

**switch (this) {**

**case DECEMBER:**

**case JANUARY:**

**case FEBRUARY:**

**return s="Зима";}**

**switch(this){**

**case MARCH:**

**case MAY:**

**case APRIL:**

**return s="Весна";**

**}**

**switch(this){**

**case JUNE:**

**case JULY:**

**case AUGUST:**

**return s="Лето";**

**}**

**switch(this){**

**case SEPTEMBER:**

**case NOVEMBER:**

**case OCTOBER:**

**return s="Осень";**

**}**

**return s;**

**}**

**static void printAll() {**

**for (Month d : values()) {**

**System.out.println("--English--\n"+d + "\nSeason: "+d.Season());**

**System.out.println("--Russian--\nМесяц: "+d.toStr()+ "\nСезон: "+d.SeasonRus());**

**System.out.println();**

**}**

**}**

**}**

**Test.java**

**package Describe;**

**public class Test {**

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

**Month f=Month.APRIL.previous();**

**System.out.println("Previous month for April:");**

**System.out.println(f);**

**System.out.println();**

**System.out.println("Next month for April: ");**

**f=Month.APRIL.next();**

**System.out.println(f);**

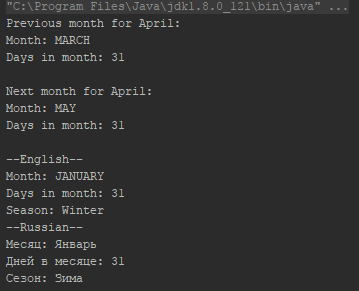
**System.out.println();**

**Month.printAll();**

**}**

**}**

*Виведення даних у консольне вікно:*



### 1.3 Дані про користувачів

*Код програми 1.3:*

**User.java**

**package User;**

**import java.util.\*;**

**import java.util.regex.Matcher;**

**import java.util.regex.Pattern;**

**public class User{**

**@SuppressWarnings("resource")**

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

**System.out.println("Введите количество пользователей в массиве: ");**

**String s=new Scanner(System.in).nextLine();**

**Pattern p = Pattern.compile("-[0-9]+|[0-1]");**

**Matcher m = p.matcher(s);**

**if(s.matches("(\\D|\\s)+")||m.find()){**

**do{**

**System.out.print("Не корректное значение. Введите еще раз: ");**

**s=new Scanner(System.in).nextLine();**

**m = p.matcher(s);**

**}**

**while(s.matches("(\\D|\\s)+")||m.find());**

**}**

**int size = Integer.parseInt(s);**

**SortedMap<String, String> users = new TreeMap<>();**

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

**System.out.println("Введите имя пользователя под № "+(i+1)+": ");**

**String key=new Scanner(System.in).nextLine();**

**if(users.containsKey(key)) {**

**do {**

**System.out.println("Такой пользователь уже существует!");**

**System.out.println("Введите имя пользователя под № "+(i+1)+": ");**

**key=new Scanner(System.in).nextLine();}while(users.containsKey(key));**

**}**

**System.out.println("Введите пароль: ");**

**String value=new Scanner(System.in).nextLine();**

**users.put(key,value);**

**}**

**System.out.println("\nПользователи с длинной пароля больше 6 цифр: \n");**

**for (Map.Entry<?, ?> entry : users.entrySet()){**

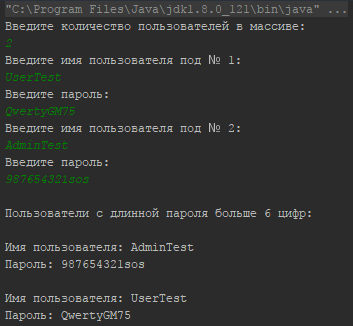
**if(entry.getValue().toString().length()>6)**

**System.out.println("Имя пользователя: "+entry.getKey() + " \nПароль: " + entry.getValue()+"\n");}**

**}**

**}**

*Виведення даних у консольне вікно:*



### 1.4  Міні-калькулятор

*Код програми 1.4:*

**FileTest.java**

**package sample;**

**import javafx.application.Application;**

**import javafx.geometry.Pos;**

**import javafx.scene.Scene;**

**import javafx.scene.control.Alert;**

**import javafx.scene.control.Alert.AlertType;**

**import javafx.scene.control.RadioButton;**

**import javafx.scene.control.ToggleGroup;**

**import javafx.scene.layout.FlowPane;**

**import javafx.event.Event;**

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

**import javafx.stage.Stage;**

**public class Calc extends Application{**

**private TextField f1, f2, f3;**

**private RadioButton r1,r2,r3,r4;**

**@Override**

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

**stage.setTitle("Calc");**

**r1 = new RadioButton("Plus");**

**r2 = new RadioButton("Minus");**

**r3 = new RadioButton("Multiply");**

**r4 = new RadioButton("Division");**

**r1.setOnAction(this::buttonClick1);**

**r2.setOnAction(this::buttonClick2);**

**r3.setOnAction(this::buttonClick3);**

**r4.setOnAction(this::buttonClick4);**

**ToggleGroup radioGroup = new ToggleGroup();**

**r1.setToggleGroup(radioGroup);**

**r2.setToggleGroup(radioGroup);**

**r3.setToggleGroup(radioGroup);**

**r4.setToggleGroup(radioGroup);**

**FlowPane rootNode = new FlowPane(10, 10);**

**rootNode.setAlignment(Pos.CENTER);**

**Scene scene = new Scene(rootNode, 200, 200);**

**stage.setScene(scene);**

**f1 = new TextField();**

**f2 = new TextField();**

**f3 = new TextField(""); f3.setEditable(false);**

**rootNode.getChildren().addAll(f1, f2, r1,r2,r3,r4, f3);**

**stage.show();**

**}**

**private void buttonClick1(Event event) {**

**try {**

**double i = Double.parseDouble(f1.getText());**

**double j = Double.parseDouble(f2.getText());**

**double k = i + j;**

**f3.setText(k + "");**

**}**

**catch (NumberFormatException e1) {**

**Alert alert = new Alert(AlertType.ERROR);**

**alert.setTitle("Error");**

**alert.setHeaderText("Please enter values!");**

**alert.showAndWait();**

**}**

**}**

**private void buttonClick2(Event event) {**

**try {**

**double i = Double.parseDouble(f1.getText());**

**double j = Double.parseDouble(f2.getText());**

**double k = i - j;**

**f3.setText(k + "");**

**}**

**catch (NumberFormatException e1) {**

**Alert alert = new Alert(AlertType.ERROR);**

**alert.setTitle("Error");**

**alert.setHeaderText("Please enter values!");**

**alert.showAndWait();**

**}**

**}**

**private void buttonClick3(Event event) {**

**try {**

**double i = Double.parseDouble(f1.getText());**

**double j = Double.parseDouble(f2.getText());**

**double k = i \* j;**

**f3.setText(k + "");**

**}**

**catch (NumberFormatException e1) {**

**Alert alert = new Alert(AlertType.ERROR);**

**alert.setTitle("Error");**

**alert.setHeaderText("Please enter values!");**

**alert.showAndWait();**

**}**

**}**

**private void buttonClick4(Event event) {**

**try {**

**double i = Double.parseDouble(f1.getText());**

**double j = Double.parseDouble(f2.getText());**

**double k = i / j;**

**f3.setText(k + "");**

**}**

**catch (NumberFormatException e1) {**

**Alert alert = new Alert(AlertType.ERROR);**

**alert.setTitle("Error");**

**alert.setHeaderText("Please enter values!");**

**alert.showAndWait();**

**}**

**}**

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

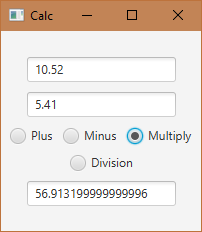
**launch(args);**

**}**

**}**

**Sample.fxml**

*Після запуску програми відкривається наступне вікно:*



**1.5 Словник**

*Код програми 1.5:*

**Mainn.java**

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.AbstractMap;

import java.util.Comparator;

import java.util.Map;

import java.util.Map.Entry;

import java.util.TreeMap;

import javafx.application.Application;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.value.ObservableValue;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.scene.Scene;

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.control.TextField;

import javafx.scene.control.cell.TextFieldTableCell;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import javafx.util.Callback;

public class Mainn extends Application {

private VBox vBox;private VBox vBox1;

private Map<String, String> map = new TreeMap<>();

private ObservableList<Map.Entry<String, String>> observable;

private TableView<Map.Entry<String, String>> table = new TableView<>();

private TableColumn<Map.Entry<String, String>, String> russian = new TableColumn<>("Russian");

private TableColumn<Map.Entry<String, String>, String> english = new TableColumn<>("English");

private TextArea tArea = new TextArea();

@Override

public void start(Stage primaryStage) throws Exception {

table.setEditable(true);

fillList();

initColumns();

initPane();

find();

observable = FXCollections.observableArrayList(map.entrySet());

BorderPane root = new BorderPane();

table.getColumns().clear();

table.setItems(observable);

table.getColumns().setAll(english, russian);

root.setCenter(table);

root.setLeft(vBox);

root.setRight(vBox1);

Scene scene = new Scene(root);

primaryStage.setTitle("Vocabulary");

primaryStage.setScene(scene);

primaryStage.show();

}

private void initColumns() {

russian.setCellValueFactory(

new Callback<TableColumn.CellDataFeatures<Map.Entry<String, String>, String>, ObservableValue<String>>() {

@Override

public ObservableValue<String> call(

TableColumn.CellDataFeatures<Map.Entry<String, String>, String> p) {

return new SimpleStringProperty(p.getValue().getValue());

}

});

english.setCellValueFactory(

new Callback<TableColumn.CellDataFeatures<Map.Entry<String, String>, String>, ObservableValue<String>>() {

@Override

public ObservableValue<String> call(

TableColumn.CellDataFeatures<Map.Entry<String, String>, String> p) {

return new SimpleStringProperty(p.getValue().getKey());

}

});

russian.setCellFactory(TextFieldTableCell.forTableColumn());

english.setCellFactory(TextFieldTableCell.forTableColumn());

}

@Override

public void stop() {

FXCollections.sort(observable, new CompareFXCollect());

try (PrintWriter pWriter = new PrintWriter(new FileWriter("dictionary.txt"));) {

for (Entry<String, String> entry : observable) {

pWriter.println(entry.getKey() + " - " + entry.getValue());

}

} catch (Exception e) {

System.out.println("error");

e.printStackTrace();

}

}

private void find() {

Label l =new Label("Search");

TextField tField = new TextField();

vBox1=new VBox(l,tField, tArea);

vBox1.setMaxSize(200,100);

vBox1.setPadding(new Insets(20, 30, 30, 30));

tField.textProperty().addListener((obs, oldValue, newValue) -> {

tArea.clear();

for (Entry<String, String> entry : observable) {

if (entry.getKey().contains(newValue) || contain(newValue, entry.getValue())) {

tArea.setText(entry.toString());

}

}

});

}

private void initPane() {

Button buttonAdd = new Button("Add word");

TextField tField = new TextField();

TextField result = new TextField();

Label l1=new Label("Englsih word");

Label l2=new Label("Russian word");

vBox = new VBox(buttonAdd,l1, tField,l2, result);

vBox.setPadding(new Insets(7, 7, 7, 7));

buttonAdd.setOnAction(event -> {

observable.add(new AbstractMap.SimpleEntry(tField.getText(),result.getText()));

table.scrollTo(observable.size());

FXCollections.sort(observable, new CompareFXCollect());

});

}

private boolean contain(String value, String entryValues) {

String[] strings = entryValues.split(",");

for (int i=0;i<strings.length;i++) {

char[] m=strings[i].toCharArray();

for(int j=0;j<m.length;j++) {

if (strings[i].equals(value)||strings[i].equals(m[j])) {

return true;

}

}}

return false;

}

private void fillList() {

try {

int count = 0;

FileReader fReader = new FileReader("dictionary.txt");

BufferedReader bReader = new BufferedReader(fReader);

String[] strings;

String string;

while (count++ < 200) {

string = bReader.readLine();

if (string == null) {

continue;

}

strings = string.split(" - ");

if (strings == null || strings[0].contains(":")) {

continue;

}

for (int i = 1; i < strings.length; i++) {

map.put(strings[0], strings[i]);

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

launch(args);

}

class CompareFXCollect implements Comparator<Map.Entry<String, String>> {

@Override

public int compare(Entry<String, String> o1, Entry<String, String> o2) {

int res = String.CASE\_INSENSITIVE\_ORDER.compare(o1.getKey(), o2.getKey());

if (res == 0) {

res = o1.getKey().compareTo(o2.getKey());

}

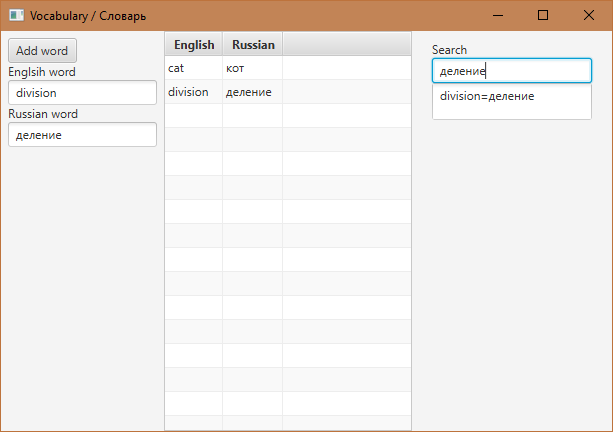
return res;

}

}

}

*Після запуску програми відкривається наступне вікно:*



**Висновок:**

У даній роботі за допомогою мови програмування Java та використання попереднього досвіду було cтворено програми графічного інтерфейсу користувача, під час розробки було опрацьовано JavaFX та переліки.