МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА «ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ІМЕНІ А. В. ДАБАГЯНА»

ЗВІТ

З лабораторної роботи №4

«СТВОРЕННЯ GUI-ЗАСТОСУНКІВ»

ВИКОНАВ:

студент групи КН-422ч

Максим БЕЛОШИЦЬКИЙ

ПЕРЕВІРИВ:

Асистент каф. ПІІТУ

Олексій КОНДРАТОВ

Харків – 2023

**ЗМІСТ**

[МЕТА 3](#_Toc167219850)

[Завдання 1.1 4](#_Toc167219851)

[Код завдання 1.1: 4](#_Toc167219852)

[Main.java 4](#_Toc167219853)

[Conference.java 12](#_Toc167219854)

[Meeting.java 14](#_Toc167219855)

[Тестування за допомогою JUnit: 16](#_Toc167219856)

[ConferenceTest.java 16](#_Toc167219857)

[MeetingTest.java 18](#_Toc167219858)

[Результати виконання програми 19](#_Toc167219859)

[conference.xml 22](#_Toc167219860)

[Завдання 1.2 27](#_Toc167219861)

[Код завдання 1.2: 27](#_Toc167219862)

[Main.java 27](#_Toc167219863)

[Тестування за допомогою JUnit: 28](#_Toc167219864)

[MainTest.java 28](#_Toc167219865)

[Результати виконання програми 29](#_Toc167219866)

[Завдання 1.3 31](#_Toc167219867)

[Код завдання 1.3: 31](#_Toc167219868)

[Kyivstar.java 31](#_Toc167219869)

[Тестування за допомогою JUnit: 33](#_Toc167219870)

[CalculatorAppTest.java 33](#_Toc167219871)

[Результати виконання програм 35](#_Toc167219872)

[Завдання 1.4 36](#_Toc167219873)

[Код завдання 1.4: 36](#_Toc167219874)

[DictionaryApp.java 36](#_Toc167219875)

[Тестування за допомогою JUnit: 39](#_Toc167219876)

[DictionaryAppTest.java 39](#_Toc167219877)

[Результати виконання програми 40](#_Toc167219878)

[Завдання для самоконтролю 43](#_Toc167219879)

[Код завдання 2.1: 43](#_Toc167219880)

[Результати виконання програми 44](#_Toc167219881)

[Код завдання 2.2: 45](#_Toc167219882)

[Результати виконання програми 46](#_Toc167219883)

[Код завдання 2.3: 47](#_Toc167219884)

[Результати виконання програми 48](#_Toc167219885)

[ВИСНОВОК 49](#_Toc167219886)

# **МЕТА**

Мета лабораторної роботи: розробити графічний інтерфейс користувача (GUI) за допомогою JavaFX для виконання завдань, що включають обробку даних, введених користувачем, роботу зі списками та таблицями, а також використання різноманітних компонентів, таких як кнопки, текстові поля та меню.

У рамках лабораторної роботи передбачено:

1. Реалізацію головного вікна з меню, яке містить функції створення нового набору даних, завантаження та збереження даних в XML-форматі, пошук та сортування даних, а також вікно "Про програму".
2. Розміщення у лівій частині вікна рядків для введення скалярних даних, області відображення результатів пошуку та кнопок для виконання основних функцій програми.
3. Розміщення у середній частині вікна таблиці для відображення та редагування даних.
4. Використання ObservableList для роботи зі списками дійсних чисел, змінюючи їх початковий порядок розташування.
5. Реалізацію міні-калькулятора, де користувач може ввести числа у два поля та обрати одну з чотирьох арифметичних операцій.
6. Розробку програми для перегляду словника англо-українських слів з можливістю пошуку та додавання нових слів, збереження даних в Map.

Ці завдання передбачають використання різноманітних елементів графічного інтерфейсу, роботу зі списками, таблицями, текстовими полями та кнопками, а також збереження та завантаження даних у форматі XML.

# Завдання 1.1

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 (десеріалізація) даних з XML-документа для редагування;
* зберігання змінених даних в XML-документі за допомогою засобів серіалізації XStream;
* пошук за ознаками, визначеними в [лабораторній роботі № 3](http://iwanoff.inf.ua/java_ua/LabTraining03.html) курсу "Основи програмування Java " попереднього семестру;
* здійснення сортування за ознаками, визначеними в [лабораторній роботі № 4](http://iwanoff.inf.ua/java_ua/LabTraining04.html) курсу "Основи програмування Java " попереднього семестру;
* отримання вікна "Про програму" з даними про програму й 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.

## Код завдання 1.1:

### Main.java

package *edu.lab03\_ind*;

import *javafx.application.Application*;

import *javafx.beans.property.SimpleIntegerProperty*;

import *javafx.geometry.Insets*;

import *javafx.scene.Scene*;

import *javafx.scene.control.\**;

import *javafx.scene.layout.BorderPane*;

import *javafx.scene.layout.HBox*;

import *javafx.stage.Modality*;

import *javafx.stage.Stage*;

import *javafx.scene.layout.VBox*;

import *java.util.ArrayList*;

import *java.util.List*;

import *java.util.stream.Collectors*;

import *com.thoughtworks.xstream.XStream*;

import *com.thoughtworks.xstream.io.xml.DomDriver*;

import *java.io.FileNotFoundException*;

import *java.io.FileReader*;

import *java.io.FileWriter*;

import *java.io.IOException*;

*/\*\**

*\* The main class that extends the Application class and serves as the entry point for the JavaFX application.*

*\*/*

*/\*\**

*\* The Main class is the entry point of the application and extends the Application class.*

*\* It provides the main method to launch the application and defines the UI components and event handlers.*

*\*/*

*public* *class* Main *extends* *Application* {

*private* *TableView*<*Conference*> conferenceTable;

*private* *TableView*<*Meeting*> meetingTable;

*private* *List*<*Conference*> conferences = new *ArrayList*<>();

*private* *boolean* showingConferences = true;

*/\*\**

*\* The main method is the entry point of the application.*

*\* It launches the JavaFX application by calling the launch method.*

*\**

*\* @param args the command line arguments*

*\*/*

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

        launch(args);

    }

    @*SuppressWarnings*("unchecked")

    @*Override*

*public* *void* start(*Stage* *primaryStage*) {

        primaryStage.setTitle("Conference Manager");

*// Initialize buttons*

*Button* addConferenceButton = new Button("Add Conference");

*Button* addMeetingButton = new Button("Add Meeting");

*Button* switchViewButton = new Button("Switch View");

*Button* searchMeetingButton = new Button("Search Meeting");

*// Initialize menu items*

*MenuItem* aboutMenuItem = new MenuItem("About");

        aboutMenuItem.setOnAction(e -> showAboutDialog());

*MenuItem* saveMenuItem = new MenuItem("Save to XML");

        saveMenuItem.setOnAction(e -> saveToXML());

*MenuItem* loadMenuItem = new MenuItem("Load from XML");

        loadMenuItem.setOnAction(e -> loadFromXML());

*// Initialize menus*

*Menu* fileMenu = new Menu("File");

        fileMenu.getItems().addAll(saveMenuItem, loadMenuItem);

*Menu* aboutMenu = new Menu("About");

        aboutMenu.getItems().addAll(aboutMenuItem);

*// Initialize menu bar*

*MenuBar* menuBar = new MenuBar();

        menuBar.getMenus().addAll(fileMenu, aboutMenu);

*// Set event handlers*

        addConferenceButton.setOnAction(e -> showAddConferenceDialog());

        addMeetingButton.setOnAction(e -> showAddMeetingDialog());

        switchViewButton.setOnAction(e -> switchView());

        searchMeetingButton.setOnAction(e -> showSearchMeetingDialog());

*// Initialize conference table*

        conferenceTable = new *TableView*<>();

*TableColumn*<*Conference*, *String*> nameColumn = new *TableColumn*<>("Name");

        nameColumn.setCellValueFactory(cellData -> cellData.getValue().nameProperty());

*TableColumn*<*Conference*, *String*> locationColumn = new *TableColumn*<>("Location");

        locationColumn.setCellValueFactory(cellData -> cellData.getValue().locationProperty());

        conferenceTable.getColumns().addAll(nameColumn, locationColumn);

*// Initialize meeting table*

        meetingTable = new *TableView*<>();

*TableColumn*<*Meeting*, *String*> topicColumn = new *TableColumn*<>("Topic");

        topicColumn.setCellValueFactory(cellData -> cellData.getValue().topicProperty());

*TableColumn*<*Meeting*, *String*> dateColumn = new *TableColumn*<>("Date");

        dateColumn.setCellValueFactory(cellData -> cellData.getValue().dateProperty());

*TableColumn*<*Meeting*, *Integer*> participantsColumn = new *TableColumn*<>("Participants");

        participantsColumn.setCellValueFactory(cellData -> new SimpleIntegerProperty(cellData.getValue().getParticipants()).asObject());

        meetingTable.getColumns().addAll(topicColumn, dateColumn, participantsColumn);

*// Layout*

*HBox* buttons = new HBox(10);

        buttons.setPadding(new Insets(10));

        buttons.getChildren().addAll(addConferenceButton, addMeetingButton, switchViewButton, searchMeetingButton);

*BorderPane* layout = new BorderPane();

        layout.setTop(menuBar);

        layout.setCenter(conferenceTable);

        layout.setBottom(buttons);

*Scene* scene = new Scene(layout, 800, 600);

        primaryStage.setScene(scene);

        primaryStage.show();

    }

*/\*\**

*\* The switchView method is called when the switchViewButton is clicked.*

*\* It switches between displaying the conference table and the meeting table.*

*\*/*

*private* *void* switchView() {

*BorderPane* layout = (BorderPane) conferenceTable.getScene().getRoot();

        System.out.println("Switching view. Showing conferences: " + showingConferences);

        if (showingConferences) {

*// Collect all meetings from the conferences*

*List*<*Meeting*> meetings = conferences.stream()

                    .flatMap(conference -> conference.getMeetings().stream())

                    .collect(Collectors.toList());

            System.out.println("Number of meetings: " + meetings.size());

*// Check if there are any meetings to display*

            if (meetings.isEmpty()) {

                showAlert("No Meetings", "There are no meetings to display.");

                return;

            }

*// Populate the meeting table with the meetings*

            if (meetingTable != null) {

                System.out.println("Meeting table is not null");

                meetingTable.getItems().setAll(meetings);

                layout.setCenter(meetingTable);

            } else {

                System.out.println("Meeting table is null");

            }

        } else {

            System.out.println("Number of conferences: " + conferences.size());

*// Check if there are any conferences to display*

            if (conferences.isEmpty()) {

                showAlert("No Conferences", "There are no conferences to display.");

                return;

            }

*// Populate the conference table with the conferences*

            if (conferenceTable != null) {

                System.out.println("Conference table is not null");

                conferenceTable.getItems().setAll(conferences);

                layout.setCenter(conferenceTable);

            } else {

                System.out.println("Conference table is null");

            }

        }

        showingConferences = !showingConferences;

    }

*/\*\**

*\* The showAlert method displays an alert dialog with the given title and message.*

*\**

*\* @param title   the title of the alert*

*\* @param message the message of the alert*

*\*/*

*private* *void* showAlert(*String* *title*, *String* *message*) {

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

        alert.setTitle(title);

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();

    }

*/\*\**

*\* The showAddConferenceDialog method displays a dialog for adding a new conference.*

*\* It prompts the user to enter the name and location of the conference.*

*\*/*

*private* *void* showAddConferenceDialog() {

*Stage* dialog = new Stage();

        dialog.initModality(Modality.APPLICATION\_MODAL);

        dialog.setTitle("Add Conference");

*TextField* nameField = new TextField();

*TextField* locationField = new TextField();

*Button* submitButton = new Button("Submit");

        submitButton.setOnAction(e -> {

*String* name = nameField.getText();

*String* location = locationField.getText();

            conferences.add(new Conference(name, location));

            conferenceTable.getItems().setAll(conferences);

            dialog.close();

        });

*VBox* dialogVBox = new VBox(10, new Label("Name"), nameField, new Label("Location"), locationField, submitButton);

        dialogVBox.setPadding(new Insets(10));

*Scene* dialogScene = new Scene(dialogVBox, 300, 200);

        dialog.setScene(dialogScene);

        dialog.show();

    }

*/\*\**

*\* The showAddMeetingDialog method displays a dialog for adding a new meeting.*

*\* It prompts the user to select a conference, enter the topic, date, and number of participants of the meeting.*

*\*/*

*private* *void* showAddMeetingDialog() {

*Stage* dialog = new Stage();

        dialog.initModality(Modality.APPLICATION\_MODAL);

        dialog.setTitle("Add Meeting");

*// Create ComboBox for selecting a conference*

*ComboBox*<*String*> conferenceComboBox = new *ComboBox*<>();

        conferenceComboBox.getItems().setAll(conferences.stream().map(Conference::getName).collect(Collectors.toList()));

*TextField* topicField = new TextField();

*DatePicker* datePicker = new DatePicker();

*TextField* participantsField = new TextField();

*Button* submitButton = new Button("Submit");

        submitButton.setOnAction(e -> {

*String* selectedConferenceName = conferenceComboBox.getValue();

            if (selectedConferenceName != null) {

*Conference* selectedConference = conferences.stream()

                        .filter(conference -> conference.getName().equals(selectedConferenceName))

                        .findFirst()

                        .orElse(null);

                if (selectedConference != null) {

*String* topic = topicField.getText();

*String* date = datePicker.getValue().toString();

*int* participants = Integer.parseInt(participantsField.getText());

                    selectedConference.addMeeting(new Meeting(topic, date, participants));

                    if (!showingConferences) {

*List*<*Meeting*> meetings = conferences.stream()

                                .flatMap(conference -> conference.getMeetings().stream())

                                .collect(Collectors.toList());

                        meetingTable.getItems().setAll(meetings);

                    }

                    conferenceTable.refresh();

                    dialog.close();

                }

            }

        });

*VBox* dialogVBox = new VBox(10,

                new Label("Conference"), conferenceComboBox,

                new Label("Topic"), topicField,

                new Label("Date"), datePicker,

                new Label("Participants"), participantsField,

                submitButton);

        dialogVBox.setPadding(new Insets(10));

*Scene* dialogScene = new Scene(dialogVBox, 300, 300);

        dialog.setScene(dialogScene);

        dialog.show();

    }

*/\*\**

*\* The showSearchMeetingDialog method displays a dialog for searching meetings.*

*\* It prompts the user to enter a search query and displays the search results.*

*\*/*

*private* *void* showSearchMeetingDialog() {

*Stage* dialog = new Stage();

        dialog.initModality(Modality.APPLICATION\_MODAL);

        dialog.setTitle("Search Meeting");

*TextField* searchField = new TextField();

*Button* searchButton = new Button("Search");

        searchButton.setOnAction(e -> {

*String* searchQuery = searchField.getText().toLowerCase();

*List*<*Meeting*> searchResults = conferences.stream()

                    .flatMap(conference -> conference.getMeetings().stream())

                    .filter(meeting -> meeting.getTopic().toLowerCase().contains(searchQuery))

                    .collect(Collectors.toList());

            meetingTable.getItems().setAll(searchResults);

            dialog.close();

*BorderPane* layout = (BorderPane) conferenceTable.getScene().getRoot();

            layout.setCenter(meetingTable);

            showingConferences = false;

        });

*VBox* dialogVBox = new VBox(10, new Label("Search Topic"), searchField, searchButton);

        dialogVBox.setPadding(new Insets(10));

*Scene* dialogScene = new Scene(dialogVBox, 300, 150);

        dialog.setScene(dialogScene);

        dialog.show();

    }

*private* *void* showAboutDialog() {

*Stage* dialog = new Stage();

        dialog.initModality(Modality.APPLICATION\_MODAL);

        dialog.setTitle("About");

*VBox* dialogVBox = new VBox(10, new Label("Conference Manager v1.0"), new Label("Author: Maksym Beloshytskiy"));

        dialogVBox.setPadding(new Insets(10));

*Scene* dialogScene = new Scene(dialogVBox, 200, 100);

        dialog.setScene(dialogScene);

        dialog.show();

    }

*private* *void* serializeToXML(*Object* *objectToSerialize*, *String* *fileName*) {

*XStream* xStream = new XStream(new DomDriver());

        try {

*FileWriter* writer = new FileWriter(fileName);

            xStream.toXML(objectToSerialize, writer);

            writer.close();

        } catch (*IOException* *e*) {

            e.printStackTrace();

        }

    }

    @*SuppressWarnings*({ "unchecked" })

*private* *List*<*Conference*> deserializeFromXML(*String* *fileName*) {

*XStream* xStream = new XStream(new DomDriver());

        try {

            return (*List*<Conference>) xStream.fromXML(new FileReader(fileName));

        } catch (*FileNotFoundException* *e*) {

            e.printStackTrace();

            return new *ArrayList*<>();

        }

    }

*private* *void* saveToXML() {

        serializeToXML(conferences, "conferences.xml");

    }

*private* *void* loadFromXML() {

        conferences = deserializeFromXML("conferences.xml");

        conferenceTable.getItems().setAll(conferences);

    }

}

### Conference.java

package *edu.lab03\_ind*;

import *java.util.ArrayList*;

import *java.util.Comparator*;

import *java.util.List*;

import *javafx.beans.property.SimpleStringProperty*;

import *javafx.beans.property.StringProperty*;

*/\*\**

*\* Represents a conference.*

*\*/*

*public* *class* Conference {

*private* *StringProperty* name;

*private* *StringProperty* location;

*private* *List*<*Meeting*> meetings;

*/\*\**

*\* Constructs a conference with the given name and location.*

*\**

*\* @param name     the name of the conference*

*\* @param location the location of the conference*

*\*/*

*public* Conference(*String* *name*, *String* *location*) {

*this*.name = new SimpleStringProperty(name);

*this*.location = new SimpleStringProperty(location);

*this*.meetings = new *ArrayList*<>();

    }

*/\*\**

*\* Returns the name of the conference.*

*\**

*\* @return the name of the conference*

*\*/*

*public* *String* getName() {

        return name.get();

    }

*/\*\**

*\* Sets the name of the conference.*

*\**

*\* @param name the name of the conference*

*\*/*

*public* *void* setName(*String* *name*) {

*this*.name.set(name);

    }

*/\*\**

*\* Returns the property representing the name of the conference.*

*\**

*\* @return the property representing the name of the conference*

*\*/*

*public* *StringProperty* nameProperty() {

        return name;

    }

*/\*\**

*\* Returns the location of the conference.*

*\**

*\* @return the location of the conference*

*\*/*

*public* *String* getLocation() {

        return location.get();

    }

*/\*\**

*\* Sets the location of the conference.*

*\**

*\* @param location the location of the conference*

*\*/*

*public* *void* setLocation(*String* *location*) {

*this*.location.set(location);

    }

*/\*\**

*\* Returns the property representing the location of the conference.*

*\**

*\* @return the property representing the location of the conference*

*\*/*

*public* *StringProperty* locationProperty() {

        return location;

    }

*/\*\**

*\* Returns the list of meetings associated with the conference.*

*\**

*\* @return the list of meetings associated with the conference*

*\*/*

*public* *List*<*Meeting*> getMeetings() {

        return meetings;

    }

*/\*\**

*\* Adds a meeting to the conference.*

*\**

*\* @param meeting the meeting to be added*

*\*/*

*public* *void* addMeeting(*Meeting* *meeting*) {

        meetings.add(meeting);

    }

*/\*\**

*\* Sorts the meetings in the conference by their topic name.*

*\*/*

*public* *void* sortMeetingsByName() {

        meetings.sort(Comparator.comparing(Meeting::getTopic));

    }

*/\*\**

*\* Sorts the meetings in the conference by the number of participants.*

*\*/*

*public* *void* sortMeetingsByParticipants() {

        meetings.sort(Comparator.comparingInt(Meeting::getParticipants));

    }

}

### Meeting.java

package *edu.lab03\_ind*;

import *javafx.beans.property.SimpleStringProperty*;

import *javafx.beans.property.StringProperty*;

*/\*\**

*\* Represents a meeting with a topic, date, and number of participants.*

*\*/*

*public* *class* Meeting {

*private* *StringProperty* topic;

*private* *StringProperty* date;

*private* *int* participants;

*/\*\**

*\* Constructs a Meeting object with the specified topic, date, and number of participants.*

*\**

*\* @param topic        the topic of the meeting*

*\* @param date         the date of the meeting*

*\* @param participants the number of participants in the meeting*

*\*/*

*public* Meeting(*String* *topic*, *String* *date*, *int* *participants*) {

*this*.topic = new SimpleStringProperty(topic);

*this*.date = new SimpleStringProperty(date);

*this*.participants = participants;

    }

*/\*\**

*\* Returns the topic of the meeting.*

*\**

*\* @return the topic of the meeting*

*\*/*

*public* *String* getTopic() {

        return topic.get();

    }

*/\*\**

*\* Sets the topic of the meeting.*

*\**

*\* @param topic the topic of the meeting*

*\*/*

*public* *void* setTopic(*String* *topic*) {

*this*.topic.set(topic);

    }

*/\*\**

*\* Returns the StringProperty representing the topic of the meeting.*

*\**

*\* @return the StringProperty representing the topic of the meeting*

*\*/*

*public* *StringProperty* topicProperty() {

        return topic;

    }

*/\*\**

*\* Returns the date of the meeting.*

*\**

*\* @return the date of the meeting*

*\*/*

*public* *String* getDate() {

        return date.get();

    }

*/\*\**

*\* Sets the date of the meeting.*

*\**

*\* @param date the date of the meeting*

*\*/*

*public* *void* setDate(*String* *date*) {

*this*.date.set(date);

    }

*/\*\**

*\* Returns the StringProperty representing the date of the meeting.*

*\**

*\* @return the StringProperty representing the date of the meeting*

*\*/*

*public* *StringProperty* dateProperty() {

        return date;

    }

*/\*\**

*\* Returns the number of participants in the meeting.*

*\**

*\* @return the number of participants in the meeting*

*\*/*

*public* *int* getParticipants() {

        return participants;

    }

*/\*\**

*\* Sets the number of participants in the meeting.*

*\**

*\* @param participants the number of participants in the meeting*

*\*/*

*public* *void* setParticipants(*int* *participants*) {

*this*.participants = participants;

    }

}

## Тестування за допомогою JUnit:

### ConferenceTest.java

package *edu.lab03\_ind*;

import *static* *org.junit.Assert.assertEquals*;

import *static* *org.junit.Assert.assertTrue*;

import *java.util.List*;

import *org.junit.Before*;

import *org.junit.Test*;

*public* *class* ConferenceTest {

*private* *Conference* conference;

    @*Before*

*public* *void* setUp() {

        conference = new Conference("Tech Conference", "New York");

        conference.addMeeting(new Meeting("Workshop", "2024-05-21", 50));

        conference.addMeeting(new Meeting("Panel Discussion", "2024-05-22", 100));

        conference.addMeeting(new Meeting("Keynote Speech", "2024-05-23", 200));

    }

    @*Test*

*public* *void* testGetName() {

        assertEquals("Tech Conference", conference.getName());

    }

    @*Test*

*public* *void* testSetName() {

        conference.setName("Science Conference");

        assertEquals("Science Conference", conference.getName());

    }

    @*Test*

*public* *void* testGetLocation() {

        assertEquals("New York", conference.getLocation());

    }

    @*Test*

*public* *void* testSetLocation() {

        conference.setLocation("San Francisco");

        assertEquals("San Francisco", conference.getLocation());

    }

    @*Test*

*public* *void* testGetMeetings() {

*List*<*Meeting*> meetings = conference.getMeetings();

        assertEquals(3, meetings.size());

    }

    @*Test*

*public* *void* testAddMeeting() {

        conference.addMeeting(new Meeting("Seminar", "2024-05-24", 150));

*List*<*Meeting*> meetings = conference.getMeetings();

        assertEquals(4, meetings.size());

    }

    @*Test*

*public* *void* testSortMeetingsByName() {

        conference.sortMeetingsByName();

*List*<*Meeting*> meetings = conference.getMeetings();

        assertTrue(meetings.get(0).getTopic().equals("Keynote Speech"));

    }

    @*Test*

*public* *void* testSortMeetingsByParticipants() {

        conference.sortMeetingsByParticipants();

*List*<*Meeting*> meetings = conference.getMeetings();

        assertTrue(meetings.get(0).getParticipants() == 50);

    }

}

### MeetingTest.java

package *edu.lab03\_ind*;

import *static* *org.junit.Assert.assertEquals*;

import *org.junit.Test*;

*public* *class* MeetingTest {

    @*Test*

*public* *void* testGetTopic() {

*Meeting* meeting = new Meeting("Project Discussion", "2024-05-21", 10);

        assertEquals("Project Discussion", meeting.getTopic());

    }

    @*Test*

*public* *void* testSetTopic() {

*Meeting* meeting = new Meeting("Project Discussion", "2024-05-21", 10);

        meeting.setTopic("Team Meeting");

        assertEquals("Team Meeting", meeting.getTopic());

    }

    @*Test*

*public* *void* testGetDate() {

*Meeting* meeting = new Meeting("Project Discussion", "2024-05-21", 10);

        assertEquals("2024-05-21", meeting.getDate());

    }

    @*Test*

*public* *void* testSetDate() {

*Meeting* meeting = new Meeting("Project Discussion", "2024-05-21", 10);

        meeting.setDate("2024-05-22");

        assertEquals("2024-05-22", meeting.getDate());

    }

    @*Test*

*public* *void* testGetParticipants() {

*Meeting* meeting = new Meeting("Project Discussion", "2024-05-21", 10);

        assertEquals(10, meeting.getParticipants());

    }

    @*Test*

*public* *void* testSetParticipants() {

*Meeting* meeting = new Meeting("Project Discussion", "2024-05-21", 10);

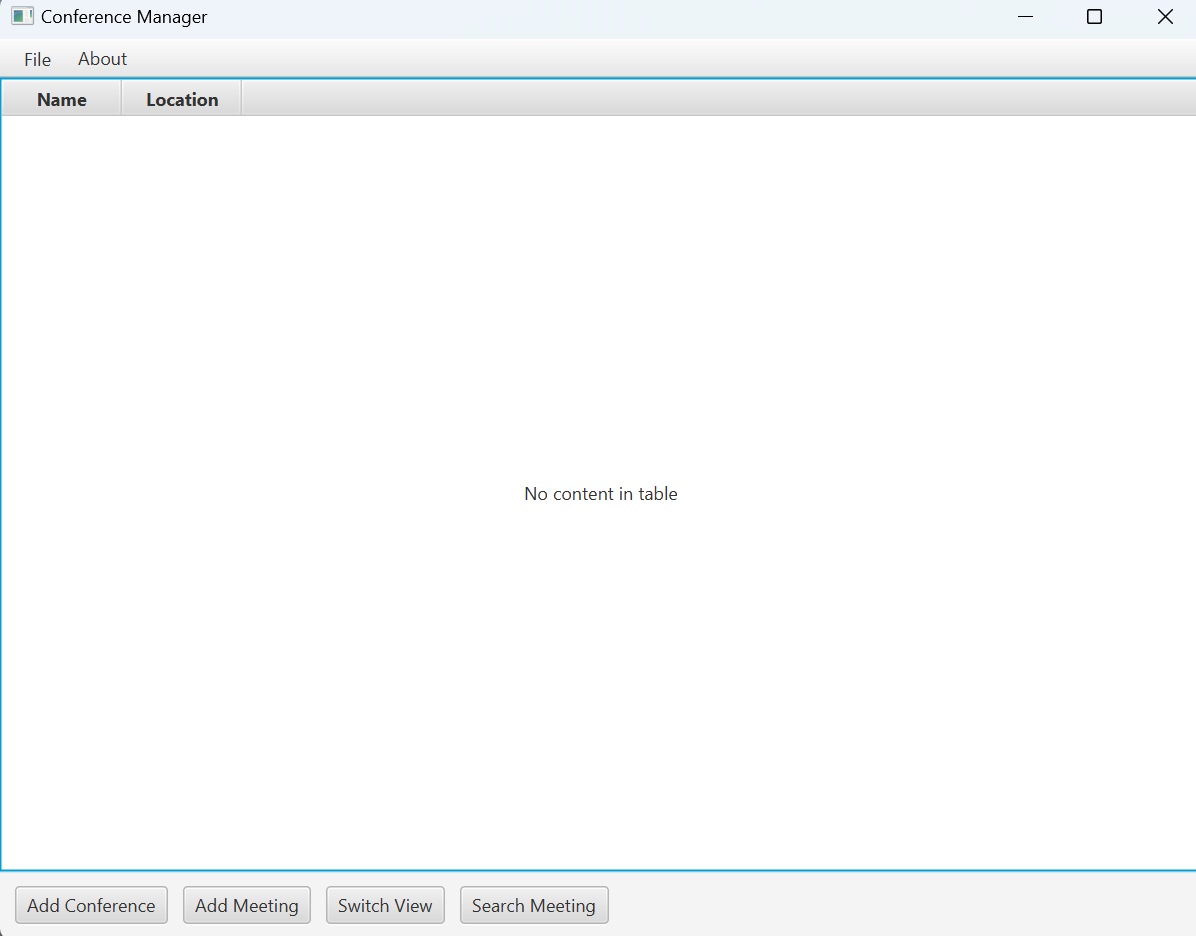
        meeting.setParticipants(15);

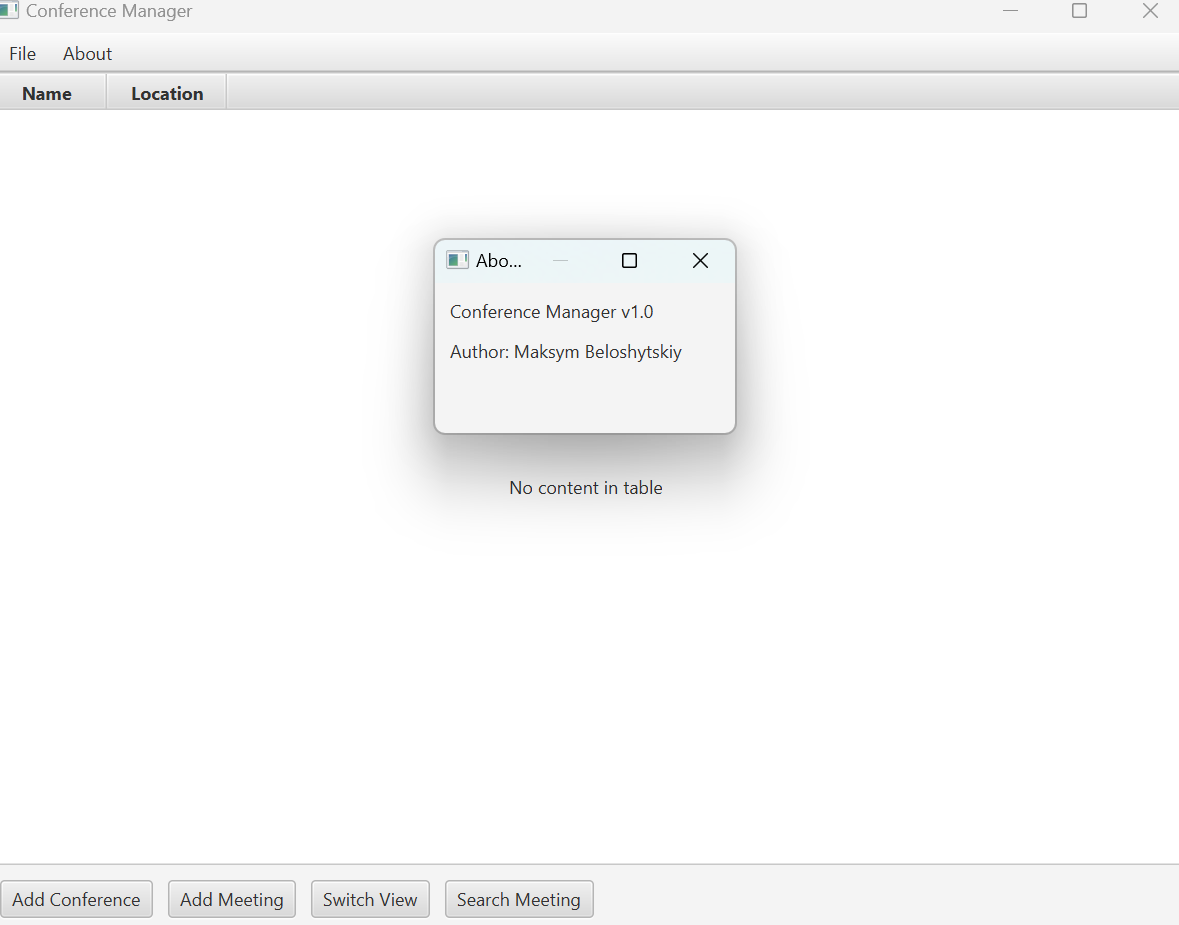
        assertEquals(15, meeting.getParticipants());

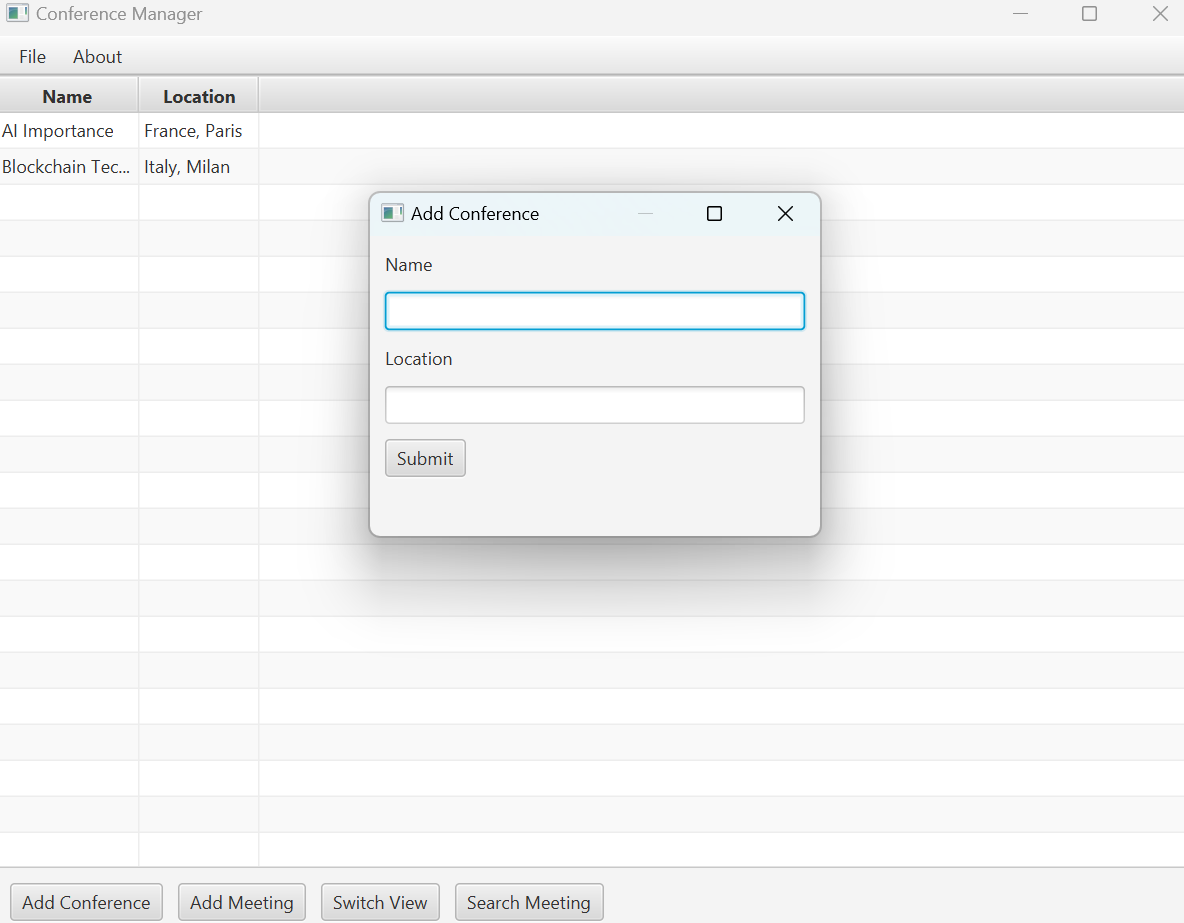
    }

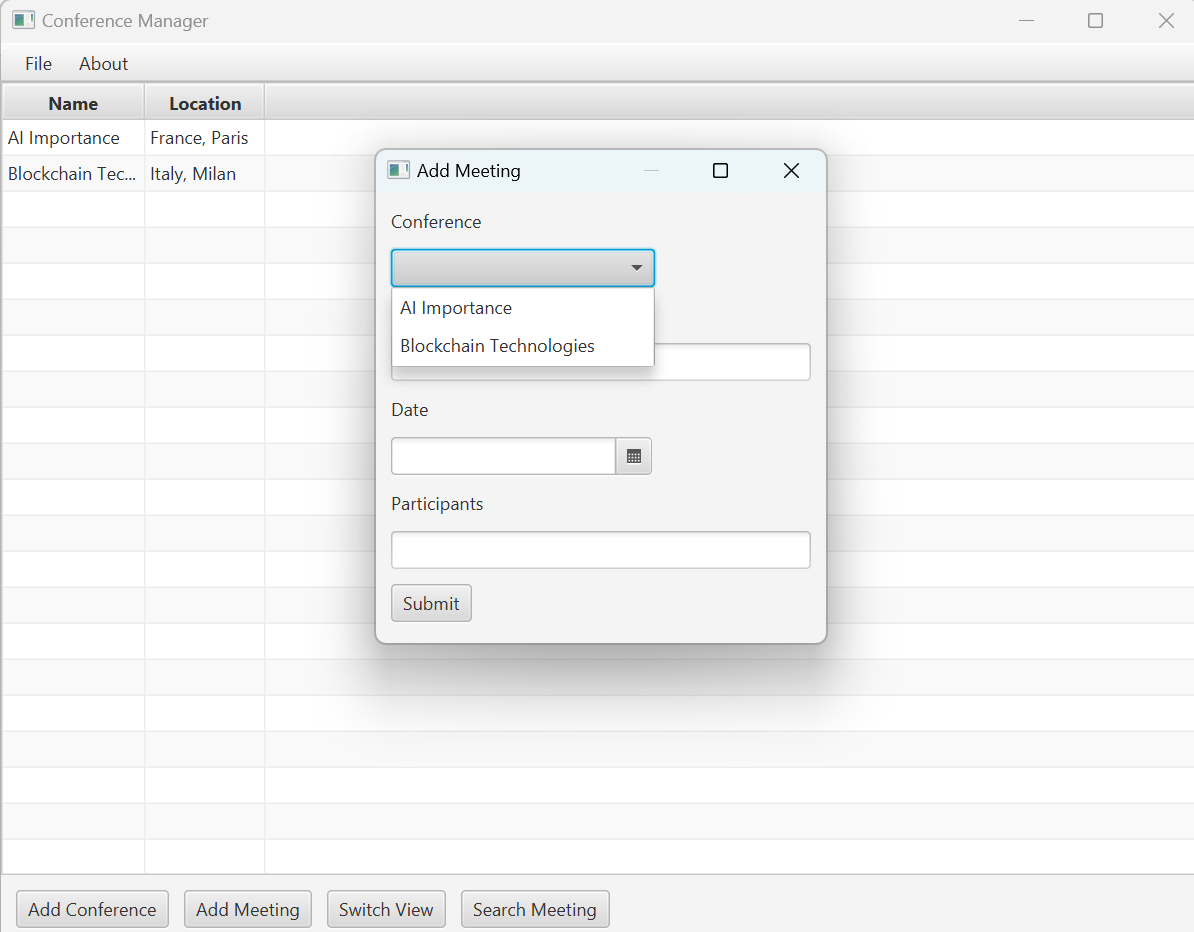
}

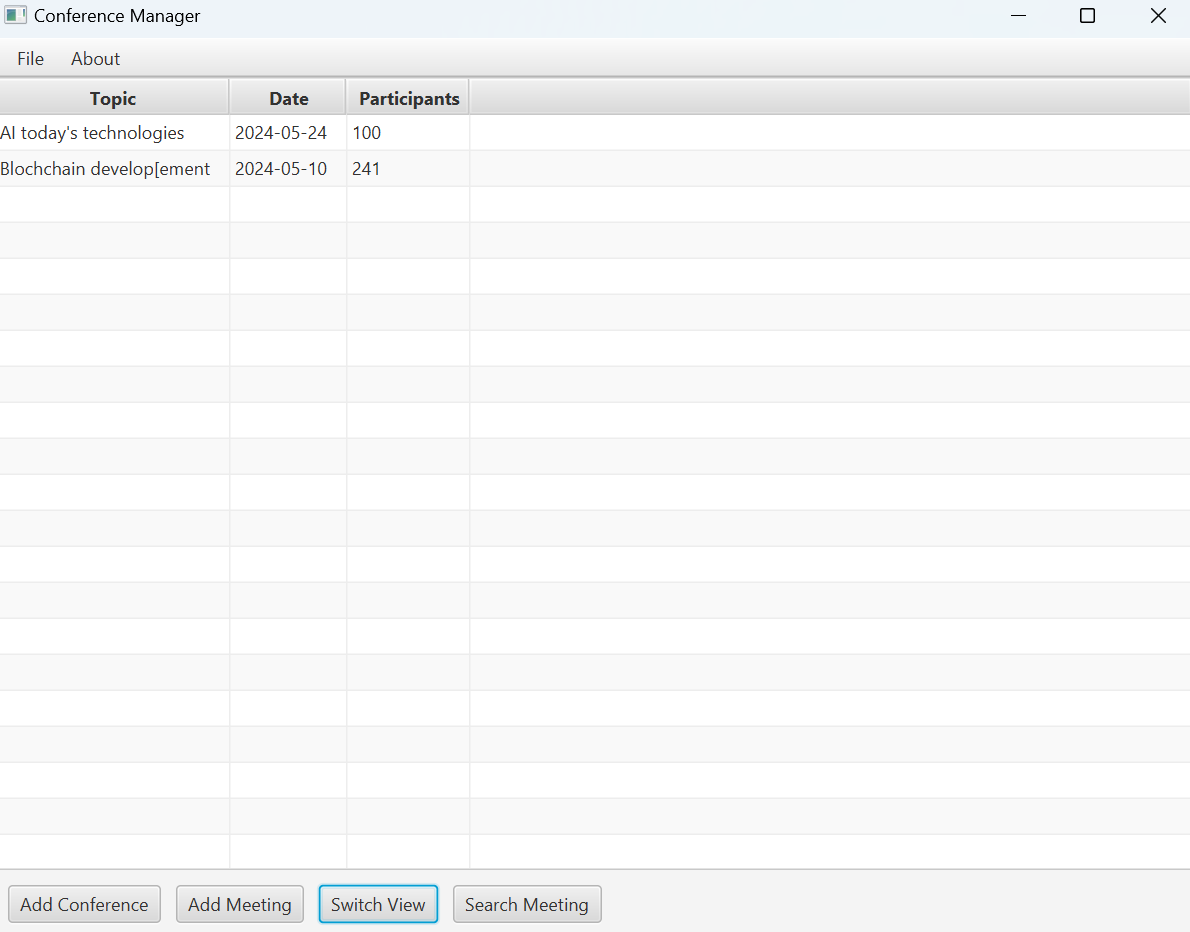
## Результати виконання програми

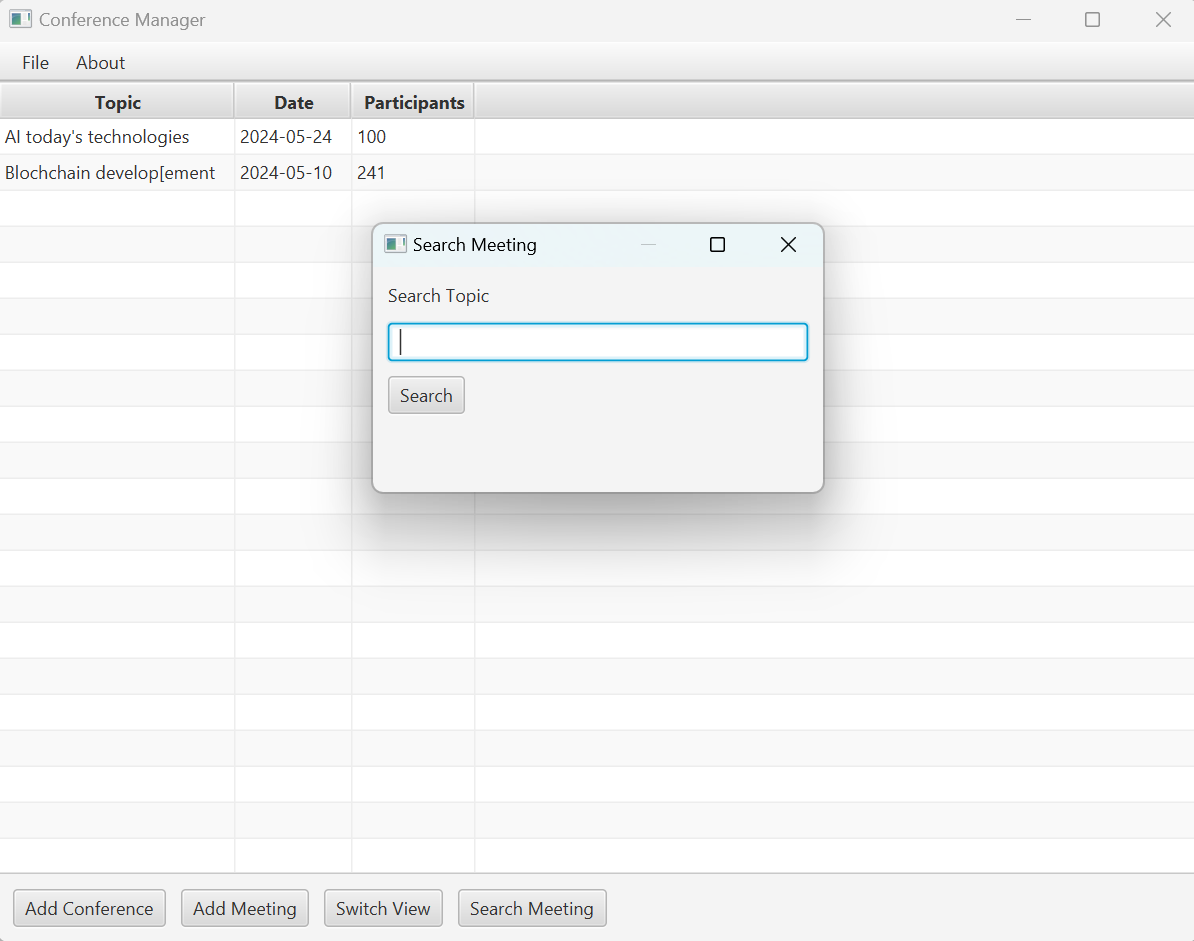


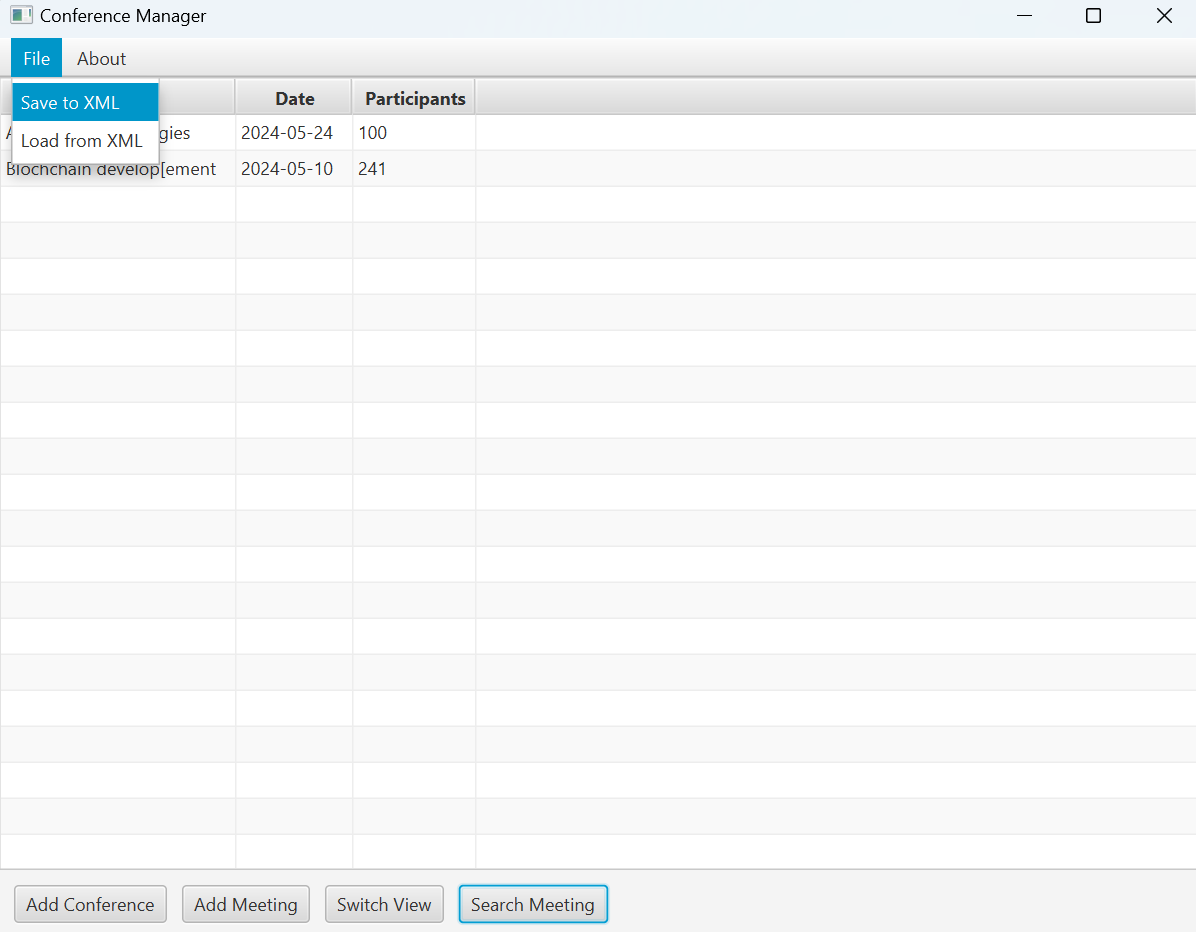












Рисунки 1.1-1.8 – «Результат створення конференцій»

### conference.xml

<list>

  <edu.lab03\_\_ind.Conference>

    <name *class*="javafx.beans.property.SimpleStringProperty">

      <value>AI Importance</value>

      <valid>true</valid>

      <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

        <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

        <listener *class*="javafx.beans.WeakInvalidationListener">

          <ref>

            <referent *class*="null"/>

            <queue *class*="java.lang.ref.ReferenceQueue$Null">

              <lock/>

              <queueLength>0</queueLength>

            </queue>

          </ref>

        </listener>

      </helper>

      <name></name>

    </name>

    <location *class*="javafx.beans.property.SimpleStringProperty">

      <value>France, Paris</value>

      <valid>true</valid>

      <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

        <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

        <listener *class*="javafx.beans.WeakInvalidationListener">

          <ref>

            <referent *class*="null"/>

            <queue *class*="java.lang.ref.ReferenceQueue$Null" *reference*="../../../../../name/helper/listener/ref/queue"/>

          </ref>

        </listener>

      </helper>

      <name></name>

    </location>

    <meetings>

      <edu.lab03\_\_ind.Meeting>

        <topic *class*="javafx.beans.property.SimpleStringProperty">

          <value>AI today&apos;s technologies</value>

          <valid>true</valid>

          <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

            <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

            <listener *class*="javafx.beans.WeakInvalidationListener">

              <ref>

                <referent *class*="null"/>

                <queue *class*="java.lang.ref.ReferenceQueue$Null" *reference*="../../../../../../../name/helper/listener/ref/queue"/>

              </ref>

            </listener>

          </helper>

          <name></name>

        </topic>

        <date *class*="javafx.beans.property.SimpleStringProperty">

          <value>2024-05-24</value>

          <valid>true</valid>

          <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

            <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

            <listener *class*="javafx.beans.WeakInvalidationListener">

              <ref>

                <referent *class*="null"/>

                <queue *class*="java.lang.ref.ReferenceQueue$Null" *reference*="../../../../../../../name/helper/listener/ref/queue"/>

              </ref>

            </listener>

          </helper>

          <name></name>

        </date>

        <participants>100</participants>

      </edu.lab03\_\_ind.Meeting>

    </meetings>

  </edu.lab03\_\_ind.Conference>

  <edu.lab03\_\_ind.Conference>

    <name *class*="javafx.beans.property.SimpleStringProperty">

      <value>Blockchain Technologies</value>

      <valid>true</valid>

      <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

        <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

        <listener *class*="javafx.beans.WeakInvalidationListener">

          <ref>

            <referent *class*="null"/>

            <queue *class*="java.lang.ref.ReferenceQueue$Null" *reference*="../../../../../../edu.lab03\_\_ind.Conference/name/helper/listener/ref/queue"/>

          </ref>

        </listener>

      </helper>

      <name></name>

    </name>

    <location *class*="javafx.beans.property.SimpleStringProperty">

      <value>Italy, Milan</value>

      <valid>true</valid>

      <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

        <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

        <listener *class*="javafx.beans.WeakInvalidationListener">

          <ref>

            <referent *class*="null"/>

            <queue *class*="java.lang.ref.ReferenceQueue$Null" *reference*="../../../../../../edu.lab03\_\_ind.Conference/name/helper/listener/ref/queue"/>

          </ref>

        </listener>

      </helper>

      <name></name>

    </location>

    <meetings>

      <edu.lab03\_\_ind.Meeting>

        <topic *class*="javafx.beans.property.SimpleStringProperty">

          <value>Blochchain develop[ement</value>

          <valid>true</valid>

          <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

            <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

            <listener *class*="javafx.beans.WeakInvalidationListener">

              <ref>

                <referent *class*="null"/>

                <queue *class*="java.lang.ref.ReferenceQueue$Null" *reference*="../../../../../../../../edu.lab03\_\_ind.Conference/name/helper/listener/ref/queue"/>

              </ref>

            </listener>

          </helper>

          <name></name>

        </topic>

        <date *class*="javafx.beans.property.SimpleStringProperty">

          <value>2024-05-10</value>

          <valid>true</valid>

          <helper *class*="com.sun.javafx.binding.ExpressionHelper$SingleInvalidation">

            <observable *class*="javafx.beans.property.SimpleStringProperty" *reference*="../.."/>

            <listener *class*="javafx.beans.WeakInvalidationListener">

              <ref>

                <referent *class*="null"/>

                <queue *class*="java.lang.ref.ReferenceQueue$Null" *reference*="../../../../../../../../edu.lab03\_\_ind.Conference/name/helper/listener/ref/queue"/>

              </ref>

            </listener>

          </helper>

          <name></name>

        </date>

        <participants>241</participants>

      </edu.lab03\_\_ind.Meeting>

    </meetings>

  </edu.lab03\_\_ind.Conference>

</list>

У даному коді реалізовано графічний інтерфейс користувача (GUI) за допомогою JavaFX для обробки даних, що включають індивідуальні завдання попередніх лабораторних робіт. Основні елементи та функції програми відображені у вікні, які було визначено у завданні:

1. Меню:
   * Реалізовано меню з наступними пунктами:
     + "File" з можливістю збереження та завантаження даних з XML-документа.
     + "About" з можливістю перегляду інформації про програму та автора.
2. Кнопки:
   * "Add Conference": для додавання нової конференції.
   * "Add Meeting": для додавання нової зустрічі.
   * "Switch View": для перемикання між відображенням таблиці конференцій та зустрічей.
   * "Search Meeting": для пошуку зустрічей за ознаками.
3. Таблиці:
   * Реалізовано таблиці для відображення та редагування даних про конференції та зустрічі.
4. Обробка даних:
   * Дані про конференції та зустрічі зберігаються у списку.
   * Реалізовано функції для додавання, пошуку та відображення даних.
5. Робота з XML:
   * Дані можуть бути збережені у XML-форматі за допомогою серіалізації XStream та завантажені з XML-документа.

Цей код відповідає вимогам завдання та забезпечує базовий функціонал для роботи з даними та їх відображення в графічному інтерфейсі.

# Завдання 1.2

Створити список (ObservableList) дійсних чисел типу Double. Змінити початковий порядок розташування чисел так, щоб спочатку були розташовані додатні числа (без зміни їх відносного порядку), а потім – від'ємні у порядку, протилежному початковому. Кожна зміна стану списку повинна обумовлювати виведення елементів списку в консольне вікно.

## Код завдання 1.2:

### Main.java

package *edu.lab03\_t00*;

import *javafx.collections.FXCollections*;

import *javafx.collections.ObservableList*;

*public* *class* Main {

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

*// Створення початкового списку дійсних чисел*

*ObservableList*<*Double*> numbers = FXCollections.observableArrayList(3.5, -2.0, 1.0, -4.5, 2.5, -1.5);

*// Виведення початкового списку*

        System.out.println("Starter List:");

        System.out.println(numbers);

*// Створення нового списку, який міститиме додатні числа, а потім від'ємні у протилежному порядку*

*ObservableList*<*Double*> sortedNumbers = FXCollections.observableArrayList();

*// Додавання додатних чисел до нового списку*

        for (*Double* number : numbers) {

            if (number > 0) {

                sortedNumbers.add(number);

            }

        }

*// Додавання від'ємних чисел у протилежному порядку до нового списку*

        for (*int* i = numbers.size() - 1; i >= 0; i--) {

            if (numbers.get(i) < 0) {

                sortedNumbers.add(numbers.get(i));

            }

        }

*// Виведення відсортованого списку*

        System.out.println("Sorted List:");

        System.out.println(sortedNumbers);

    }

}

## Тестування за допомогою JUnit:

### MainTest.java

package *edu.lab03\_t00*;

import *javafx.collections.FXCollections*;

import *javafx.collections.ObservableList*;

import *org.junit.Test*;

import *static* *org.junit.Assert.assertEquals*;

*public* *class* MainTest {

    @*Test*

*public* *void* testSortingNumbers() {

*ObservableList*<*Double*> numbers = FXCollections.observableArrayList(3.5, -2.0, 1.0, -4.5, 2.5, -1.5);

*ObservableList*<*Double*> expectedSortedNumbers = FXCollections.observableArrayList(3.5, 1.0, 2.5, -1.5, -2.0, -4.5);

*ObservableList*<*Double*> sortedNumbers = Main.sortNumbers(numbers);

        assertEquals(expectedSortedNumbers, sortedNumbers);

    }

    @*Test*

*public* *void* testSortingEmptyList() {

*ObservableList*<*Double*> numbers = FXCollections.observableArrayList();

*ObservableList*<*Double*> expectedSortedNumbers = FXCollections.observableArrayList();

*ObservableList*<*Double*> sortedNumbers = Main.sortNumbers(numbers);

        assertEquals(expectedSortedNumbers, sortedNumbers);

    }

    @*Test*

*public* *void* testSortingOnlyPositiveNumbers() {

*ObservableList*<*Double*> numbers = FXCollections.observableArrayList(3.5, 2.0, 1.0, 4.5, 2.5, 1.5);

*ObservableList*<*Double*> expectedSortedNumbers = FXCollections.observableArrayList(3.5, 2.0, 1.0, 4.5, 2.5, 1.5);

*ObservableList*<*Double*> sortedNumbers = Main.sortNumbers(numbers);

        assertEquals(expectedSortedNumbers, sortedNumbers);

    }

    @*Test*

*public* *void* testSortingOnlyNegativeNumbers() {

*ObservableList*<*Double*> numbers = FXCollections.observableArrayList(-3.5, -2.0, -1.0, -4.5, -2.5, -1.5);

*ObservableList*<*Double*> expectedSortedNumbers = FXCollections.observableArrayList(-3.5, -2.0, -1.0, -4.5, -2.5, -1.5);

*ObservableList*<*Double*> sortedNumbers = Main.sortNumbers(numbers);

        assertEquals(expectedSortedNumbers, sortedNumbers);

    }

    @*Test*

*public* *void* testSortingWithZero() {

*ObservableList*<*Double*> numbers = FXCollections.observableArrayList(3.5, -2.0, 0.0, -4.5, 2.5, -1.5);

*ObservableList*<*Double*> expectedSortedNumbers = FXCollections.observableArrayList(3.5, 0.0, 2.5, -1.5, -2.0, -4.5);

*ObservableList*<*Double*> sortedNumbers = Main.sortNumbers(numbers);

        assertEquals(expectedSortedNumbers, sortedNumbers);

    }

}

## Результати виконання програми

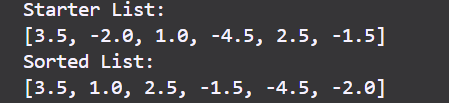


Рисунок 1.9 – «Результат роботи програми»

У цьому завданні було створено Java програму, яка працює з JavaFX для роботи зі списками дійсних чисел. Основні кроки, що виконуються у програмі:

1. Створення початкового списку дійсних чисел:
   * Використовується клас ObservableList з пакету javafx.collections.FXCollections.
   * Числа додаються до списку за допомогою методу FXCollections.observableArrayList().
2. Виведення початкового списку:
   * Виводиться початковий список чисел у консоль.
3. Створення нового списку, який міститиме додатні числа, а потім від'ємні у протилежному порядку:
   * Створюється порожній список sortedNumbers.
   * Проходиться по кожному числу в початковому списку та додаються додатні числа до нового списку.
   * Потім проходиться по початковому списку в зворотньому порядку та додаються від'ємні числа до нового списку.
4. Виведення відсортованого списку:
   * Виводиться відсортований список чисел у консоль.

Ця програма виконує завдання, описане в лабораторній, тобто створює новий список чисел, в якому спочатку розташовані додатні числа, а потім від'ємні у протилежному порядку.

# Завдання 1.3

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.

## Код завдання 1.3:

### Kyivstar.java

package *edu.lab03\_t01*;

import *javafx.application.Application*;

import *javafx.geometry.Insets*;

import *javafx.geometry.Pos*;

import *javafx.scene.Scene*;

import *javafx.scene.control.Button*;

import *javafx.scene.control.RadioButton*;

import *javafx.scene.control.TextField*;

import *javafx.scene.control.ToggleGroup*;

import *javafx.scene.layout.GridPane*;

import *javafx.scene.layout.HBox*;

import *javafx.stage.Stage*;

*public* *class* CalculatorApp *extends* *Application* {

*private* *TextField* numberField1;

*private* *TextField* numberField2;

*private* *TextField* resultField;

*private* *ToggleGroup* operationGroup; *// Define ToggleGroup as a class variable*

    @*Override*

*public* *void* start(*Stage* *primaryStage*) {

        primaryStage.setTitle("Simple Calculator");

*GridPane* grid = new GridPane();

        grid.setAlignment(Pos.CENTER);

        grid.setHgap(10);

        grid.setVgap(10);

        grid.setPadding(new Insets(25, 25, 25, 25));

        numberField1 = new TextField();

        numberField2 = new TextField();

        resultField = new TextField();

        resultField.setEditable(false);

        operationGroup = new ToggleGroup(); *// Initialize ToggleGroup*

*RadioButton* addRadioButton = new RadioButton("Add");

        addRadioButton.setToggleGroup(operationGroup);

        addRadioButton.setSelected(true);

*RadioButton* subtractRadioButton = new RadioButton("Subtract");

        subtractRadioButton.setToggleGroup(operationGroup);

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

        multiplyRadioButton.setToggleGroup(operationGroup);

*RadioButton* divideRadioButton = new RadioButton("Divide");

        divideRadioButton.setToggleGroup(operationGroup);

*Button* calculateButton = new Button("Calculate");

        calculateButton.setOnAction(e -> calculate());

        grid.add(numberField1, 0, 0);

        grid.add(numberField2, 0, 1);

        grid.add(new HBox(10, addRadioButton, subtractRadioButton, multiplyRadioButton, divideRadioButton), 1, 0);

        grid.add(calculateButton, 1, 1);

        grid.add(resultField, 0, 2, 2, 1);

*Scene* scene = new Scene(grid, 400, 250);

        primaryStage.setScene(scene);

        primaryStage.show();

    }

*private* *void* calculate() {

*double* num1 = Double.parseDouble(numberField1.getText());

*double* num2 = Double.parseDouble(numberField2.getText());

*double* result = 0.0;

*RadioButton* selectedRadioButton = (RadioButton) operationGroup.getSelectedToggle(); *// Use operationGroup directly*

*String* operation = selectedRadioButton.getText();

        switch (operation) {

            case "Add":

                result = num1 + num2;

                break;

            case "Subtract":

                result = num1 - num2;

                break;

            case "Multiply":

                result = num1 \* num2;

                break;

            case "Divide":

                if (num2 != 0) {

                    result = num1 / num2;

                } else {

                    resultField.setText("Error: Division by zero");

                    return;

                }

                break;

        }

        resultField.setText(String.valueOf(result));

    }

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

        launch(args);

    }

}

## Тестування за допомогою JUnit:

### CalculatorAppTest.java

package *edu.lab03\_t01*;

import *org.junit.Test*;

import *static* *org.junit.Assert.assertEquals*;

*public* *class* CalculatorAppTest {

    @*Test*

*public* *void* testAddition() {

*CalculatorApp* calculator = new CalculatorApp();

        calculator.numberField1.setText("5");

        calculator.numberField2.setText("3");

        calculator.calculate();

        assertEquals("8.0", calculator.resultField.getText());

    }

    @*Test*

*public* *void* testSubtraction() {

*CalculatorApp* calculator = new CalculatorApp();

        calculator.numberField1.setText("5");

        calculator.numberField2.setText("3");

        calculator.operationGroup.selectToggle(calculator.subtractRadioButton);

        calculator.calculate();

        assertEquals("2.0", calculator.resultField.getText());

    }

    @*Test*

*public* *void* testMultiplication() {

*CalculatorApp* calculator = new CalculatorApp();

        calculator.numberField1.setText("5");

        calculator.numberField2.setText("3");

        calculator.operationGroup.selectToggle(calculator.multiplyRadioButton);

        calculator.calculate();

        assertEquals("15.0", calculator.resultField.getText());

    }

    @*Test*

*public* *void* testDivision() {

*CalculatorApp* calculator = new CalculatorApp();

        calculator.numberField1.setText("10");

        calculator.numberField2.setText("2");

        calculator.operationGroup.selectToggle(calculator.divideRadioButton);

        calculator.calculate();

        assertEquals("5.0", calculator.resultField.getText());

    }

    @*Test*

*public* *void* testDivisionByZero() {

*CalculatorApp* calculator = new CalculatorApp();

        calculator.numberField1.setText("10");

        calculator.numberField2.setText("0");

        calculator.operationGroup.selectToggle(calculator.divideRadioButton);

        calculator.calculate();

        assertEquals("Error: Division by zero", calculator.resultField.getText());

    }

}

## Результати виконання програм

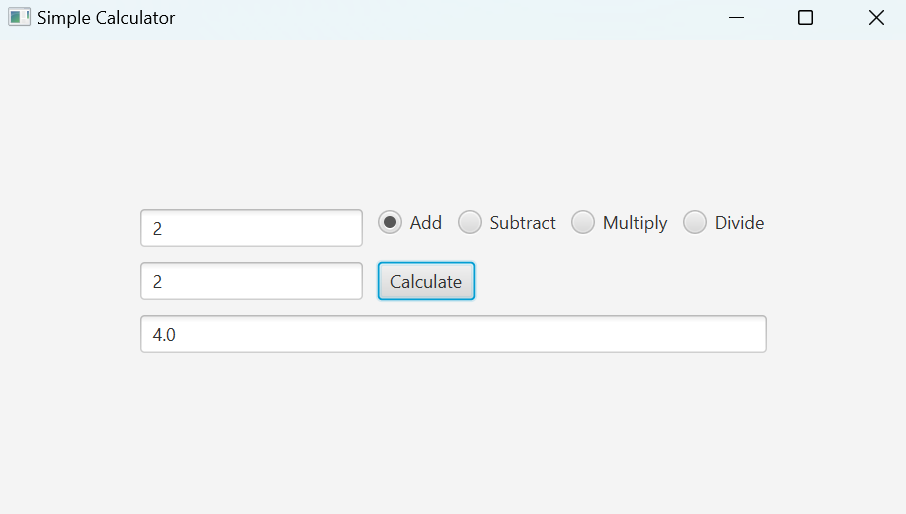


Рисунок 2.1 – «Результат роботи калькулятору»

Ця програма є простим калькулятором, який використовує графічний інтерфейс користувача на основі JavaFX. Основні елементи програми:

1. Графічний інтерфейс (GUI):
   * Вікно програми має назву "Simple Calculator".
   * Використовується GridPane для розташування елементів у вигляді таблиці.
   * У полі numberField1 вводиться перше число.
   * У полі numberField2 вводиться друге число.
   * Результат обчислення виводиться у полі resultField, яке є нередагованим.
   * Є чотири перемикачі RadioButton, що визначають операцію для обчислення: "Add" (додавання), "Subtract" (віднімання), "Multiply" (множення) та "Divide" (ділення).
   * Кнопка calculateButton викликає метод calculate(), який виконує обчислення.
2. Обробка подій:
   * При кліку на кнопку "Calculate" викликається метод calculate().
   * У методі calculate() виконується обчислення значення залежно від вибраної операції.
   * Результат обчислення виводиться у відповідному полі resultField.
3. Операції калькулятора:
   * Вибір операції визначається через групу переключачів ToggleGroup.
   * Після натискання кнопки "Calculate" виконується обчислення відповідно до обраної операції.
4. Запуск програми:
   * Метод main() викликає launch(args), що запускає додаток.

Ця програма демонструє створення простого калькулятора з графічним інтерфейсом користувача за допомогою JavaFX.

# Завдання 1.4

Розробити програму графічного інтерфейсу користувача перегляду слів невеличкого англо-українського словника. Реалізувати функції пошуку слова, додавання нових слів. Для зберігання даних використати Map.

## Код завдання 1.4:

### DictionaryApp.java

package *edu.lab03\_t02*;

import *javafx.application.Application*;

import *javafx.geometry.Insets*;

import *javafx.geometry.Pos*;

import *javafx.scene.Scene*;

import *javafx.scene.control.\**;

import *javafx.scene.layout.GridPane*;

import *javafx.scene.layout.HBox*;

import *javafx.stage.Stage*;

import *java.util.HashMap*;

import *java.util.Map*;

*public* *class* DictionaryApp *extends* *Application* {

*private* *Map*<*String*, *String*> dictionary;

*private* *TextField* searchField;

*private* *TextField* englishField;

*private* *TextField* ukrainianField;

*private* *TextArea* resultArea;

    @*Override*

*public* *void* start(*Stage* *primaryStage*) {

        primaryStage.setTitle("English-Ukrainian Dictionary");

        dictionary = new *HashMap*<>();

        populateDictionary();

*GridPane* grid = new GridPane();

        grid.setAlignment(Pos.CENTER);

        grid.setHgap(10);

        grid.setVgap(10);

        grid.setPadding(new Insets(25, 25, 25, 25));

        searchField = new TextField();

        searchField.setPromptText("Search word...");

*Button* searchButton = new Button("Search");

        searchButton.setOnAction(e -> searchWord());

        englishField = new TextField();

        englishField.setPromptText("English word");

        ukrainianField = new TextField();

        ukrainianField.setPromptText("Ukrainian translation");

*Button* addButton = new Button("Add Word");

        addButton.setOnAction(e -> addWord());

        resultArea = new TextArea();

        resultArea.setEditable(false);

        resultArea.setWrapText(true);

*HBox* searchBox = new HBox(10);

        searchBox.getChildren().addAll(searchField, searchButton);

*HBox* addBox = new HBox(10);

        addBox.getChildren().addAll(englishField, ukrainianField, addButton);

        grid.add(searchBox, 0, 0);

        grid.add(addBox, 0, 1);

        grid.add(resultArea, 0, 2, 2, 1);

*Scene* scene = new Scene(grid, 500, 300);

        primaryStage.setScene(scene);

        primaryStage.show();

    }

*private* *void* populateDictionary() {

        dictionary.put("hello", "привіт");

        dictionary.put("world", "світ");

        dictionary.put("apple", "яблуко");

        dictionary.put("book", "книга");

    }

*private* *void* searchWord() {

*String* searchKey = searchField.getText().trim().toLowerCase();

*String* result = dictionary.getOrDefault(searchKey, "Word not found.");

        resultArea.setText("Search result:\n" + searchKey + " - " + result);

    }

*private* *void* addWord() {

*String* englishWord = englishField.getText().trim().toLowerCase();

*String* ukrainianTranslation = ukrainianField.getText().trim();

        if (!englishWord.isEmpty() && !ukrainianTranslation.isEmpty()) {

            dictionary.put(englishWord, ukrainianTranslation);

            resultArea.setText("Word added:\n" + englishWord + " - " + ukrainianTranslation);

            englishField.clear();

            ukrainianField.clear();

        } else {

            resultArea.setText("Error: Both fields must be filled.");

        }

    }

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

        launch(args);

    }

}

## Тестування за допомогою JUnit:

### DictionaryAppTest.java

package *edu.lab03\_t02*;

import *org.junit.Before*;

import *org.junit.Test*;

import *static* *org.junit.Assert.assertEquals*;

*public* *class* DictionaryAppTest {

*private* *DictionaryApp* dictionaryApp;

    @*Before*

*public* *void* setUp() {

        dictionaryApp = new DictionaryApp();

        dictionaryApp.populateDictionary(); *// Populate dictionary with initial values*

    }

    @*Test*

*public* *void* testSearchExistingWord() {

        dictionaryApp.searchField.setText("hello");

        dictionaryApp.searchWord();

        assertEquals("Search result:\nhello - привіт", dictionaryApp.resultArea.getText());

    }

    @*Test*

*public* *void* testSearchNonExistingWord() {

        dictionaryApp.searchField.setText("xyz");

        dictionaryApp.searchWord();

        assertEquals("Search result:\nxyz - Word not found.", dictionaryApp.resultArea.getText());

    }

    @*Test*

*public* *void* testAddWord() {

        dictionaryApp.englishField.setText("test");

        dictionaryApp.ukrainianField.setText("тест");

        dictionaryApp.addWord();

        assertEquals("Word added:\ntest - тест", dictionaryApp.resultArea.getText());

*// Check if the word was added to the dictionary*

        assertEquals("тест", dictionaryApp.dictionary.get("test"));

    }

    @*Test*

*public* *void* testAddEmptyWord() {

        dictionaryApp.englishField.setText("");

        dictionaryApp.ukrainianField.setText("тест");

        dictionaryApp.addWord();

        assertEquals("Error: Both fields must be filled.", dictionaryApp.resultArea.getText());

*// Check if the dictionary remains unchanged*

        assertEquals(4, dictionaryApp.dictionary.size());

    }

    @*Test*

*public* *void* testAddEmptyTranslation() {

        dictionaryApp.englishField.setText("test");

        dictionaryApp.ukrainianField.setText("");

        dictionaryApp.addWord();

        assertEquals("Error: Both fields must be filled.", dictionaryApp.resultArea.getText());

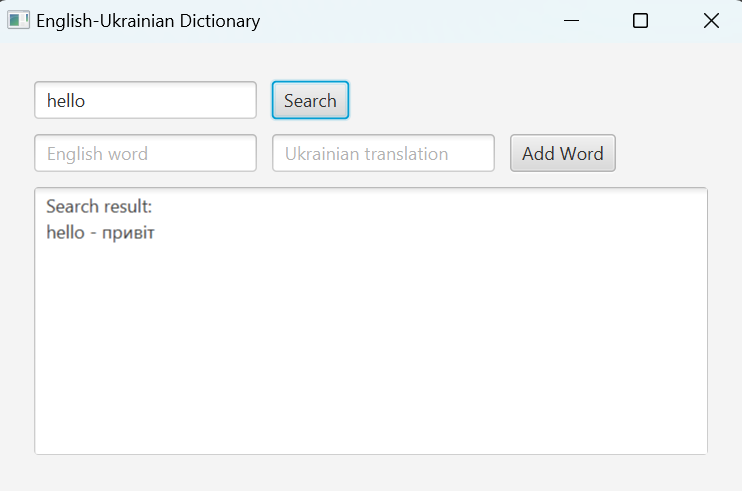
*// Check if the dictionary remains unchanged*

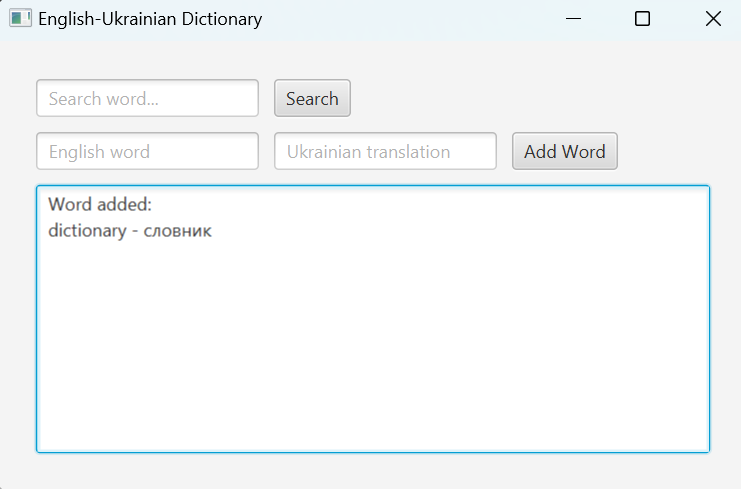
        assertEquals(4, dictionaryApp.dictionary.size());

    }

}

## Результати виконання програми





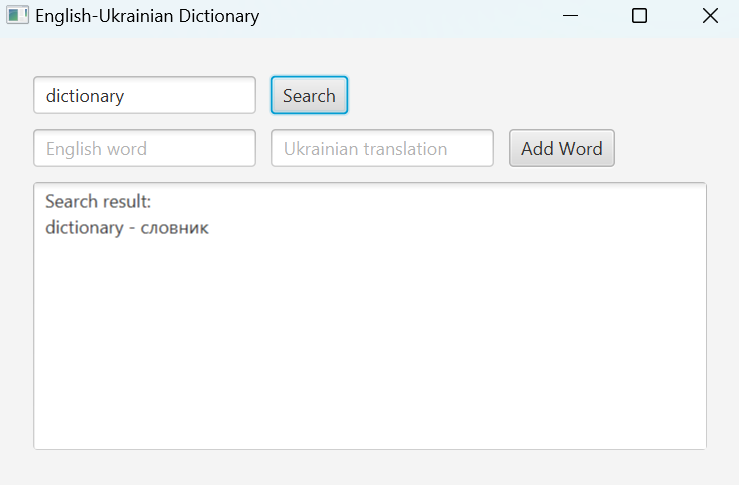


Рисунок 2.2-2.5 – «Пошук та додавання слів в словник»

У цій програмі створено простий словник англійської та української мов. Основні елементи програми:

1. Графічний інтерфейс (GUI):
   * Вікно програми має назву "English-Ukrainian Dictionary".
   * Використовується GridPane для розташування елементів у вигляді таблиці.
   * Поле searchField призначене для введення слова для пошуку.
   * Кнопка searchButton викликає метод searchWord(), який виконує пошук слова в словнику.
   * Поля englishField та ukrainianField призначені для введення нового слова та його перекладу.
   * Кнопка addButton викликає метод addWord(), який додає нове слово та його переклад до словника.
   * resultArea відображає результати пошуку або додавання слова.
2. Словник:
   * Словник реалізований за допомогою Map<String, String>, де ключ - англійське слово, а значення - його український переклад.
   * Слова та їх переклади ініціалізуються в методі populateDictionary().
3. Обробка подій:
   * При натисканні кнопки "Search" викликається метод searchWord(), який знаходить переклад введеного слова.
   * При натисканні кнопки "Add Word" викликається метод addWord(), який додає нове слово та переклад до словника.
4. Запуск програми:
   * Метод main() викликає launch(args), що запускає додаток.

Ця програма демонструє створення простого словника англійської та української мов з графічним інтерфейсом користувача за допомогою JavaFX.

# Завдання для самоконтролю

Створити програму графічного інтерфейсу користувача, у якій у двох рядках уведення користувач задає два рядки й після натискання кнопки одержує в третьому рядку результат зшивання вихідних рядків.

## Код завдання 2.1:

package *edu.lab03\_add00*;

import *javafx.application.Application*;

import *javafx.geometry.Insets*;

import *javafx.scene.Scene*;

import *javafx.scene.control.Button*;

import *javafx.scene.control.Label*;

import *javafx.scene.control.TextField*;

import *javafx.scene.layout.GridPane*;

import *javafx.stage.Stage*;

*public* *class* StringConcatenationGUI *extends* *Application* {

    @*Override*

*public* *void* start(*Stage* *primaryStage*) {

        primaryStage.setTitle("String Concatenation");

*// Create UI components*

*Label* inputLabel1 = new Label("Input 1:");

*TextField* inputField1 = new TextField();

*Label* inputLabel2 = new Label("Input 2:");

*TextField* inputField2 = new TextField();

*Button* concatenateButton = new Button("Concatenate");

*Label* resultLabel = new Label("Result:");

*// Event handler for button click*

        concatenateButton.setOnAction(e -> {

*String* input1 = inputField1.getText();

*String* input2 = inputField2.getText();

*String* result = input1 + " " + input2;

            resultLabel.setText("Result: " + result);

        });

*// Layout setup*

*GridPane* grid = new GridPane();

        grid.setPadding(new Insets(10, 10, 10, 10));

        grid.setVgap(5);

        grid.setHgap(5);

        grid.add(inputLabel1, 0, 0);

        grid.add(inputField1, 1, 0);

        grid.add(inputLabel2, 0, 1);

        grid.add(inputField2, 1, 1);

        grid.add(concatenateButton, 0, 2);

        grid.add(resultLabel, 1, 2);

*Scene* scene = new Scene(grid, 300, 150);

        primaryStage.setScene(scene);

        primaryStage.show();

    }

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

        launch(args);

    }

}

## Результати виконання програми

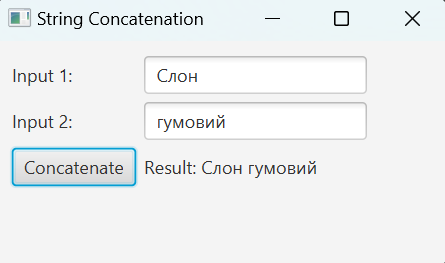


Рисунок 2.6 – «Канкатенація строк»

Створити програму графічного інтерфейсу користувача, у якій у двох рядках уведення користувач задає два дійсних числа і після натискання кнопки одержує в третьому рядку введення добуток цих чисел.

## Код завдання 2.2:

package *edu.lab03\_add01*;

import *javafx.application.Application*;

import *javafx.geometry.Insets*;

import *javafx.scene.Scene*;

import *javafx.scene.control.Button*;

import *javafx.scene.control.Label*;

import *javafx.scene.control.TextField*;

import *javafx.scene.layout.GridPane*;

import *javafx.stage.Stage*;

*public* *class* MultiplyNumbersGUI *extends* *Application* {

    @*Override*

*public* *void* start(*Stage* *primaryStage*) {

        primaryStage.setTitle("Multiply Numbers");

*// Create UI components*

*Label* inputLabel1 = new Label("Number 1:");

*TextField* inputField1 = new TextField();

*Label* inputLabel2 = new Label("Number 2:");

*TextField* inputField2 = new TextField();

*Button* multiplyButton = new Button("Multiply");

*Label* resultLabel = new Label("Result:");

*// Event handler for button click*

        multiplyButton.setOnAction(e -> {

            try {

*double* number1 = Double.parseDouble(inputField1.getText());

*double* number2 = Double.parseDouble(inputField2.getText());

*double* result = number1 \* number2;

                resultLabel.setText("Result: " + result);

            } catch (*NumberFormatException* *ex*) {

                resultLabel.setText("Invalid input");

            }

        });

*// Layout setup*

*GridPane* grid = new GridPane();

        grid.setPadding(new Insets(10, 10, 10, 10));

        grid.setVgap(5);

        grid.setHgap(5);

        grid.add(inputLabel1, 0, 0);

        grid.add(inputField1, 1, 0);

        grid.add(inputLabel2, 0, 1);

        grid.add(inputField2, 1, 1);

        grid.add(multiplyButton, 0, 2);

        grid.add(resultLabel, 1, 2);

*Scene* scene = new Scene(grid, 300, 150);

        primaryStage.setScene(scene);

        primaryStage.show();

    }

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

        launch(args);

    }

}

## Результати виконання програми

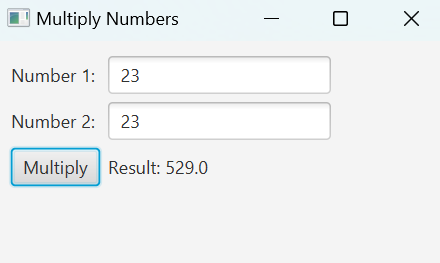


Рисунок 2.7 – «Множення числа на число»

Створити програму графічного інтерфейсу користувача, у якій у двох рядках уведення користувач задає два цілих числа і після натискання кнопки одержує в окремому діалоговому вікні введення добуток цих чисел.

## Код завдання 2.3:

package *edu.lab03\_add02*;

import *javafx.application.Application*;

import *javafx.geometry.Insets*;

import *javafx.scene.Scene*;

import *javafx.scene.control.Button*;

import *javafx.scene.control.Label*;

import *javafx.scene.control.TextField*;

import *javafx.scene.layout.GridPane*;

import *javafx.stage.Stage*;

import *javafx.scene.control.Alert*;

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

*public* *class* MultiplyNumbersGUI *extends* *Application* {

    @*Override*

*public* *void* start(*Stage* *primaryStage*) {

        primaryStage.setTitle("Multiply Numbers");

*// Create UI components*

*Label* inputLabel1 = new Label("Number 1:");

*TextField* inputField1 = new TextField();

*Label* inputLabel2 = new Label("Number 2:");

*TextField* inputField2 = new TextField();

*Button* multiplyButton = new Button("Multiply");

*// Event handler for button click*

        multiplyButton.setOnAction(e -> {

            try {

*int* number1 = Integer.parseInt(inputField1.getText());

*int* number2 = Integer.parseInt(inputField2.getText());

*int* result = number1 \* number2;

*// Display result in a dialog window*

                showAlert("Result", "The product of the numbers is: " + result, AlertType.INFORMATION);

            } catch (*NumberFormatException* *ex*) {

                showAlert("Error", "Please enter valid integer numbers", AlertType.ERROR);

            }

        });

*// Layout setup*

*GridPane* grid = new GridPane();

        grid.setPadding(new Insets(10, 10, 10, 10));

        grid.setVgap(5);

        grid.setHgap(5);

        grid.add(inputLabel1, 0, 0);

        grid.add(inputField1, 1, 0);

        grid.add(inputLabel2, 0, 1);

        grid.add(inputField2, 1, 1);

        grid.add(multiplyButton, 0, 2);

*Scene* scene = new Scene(grid, 300, 150);

        primaryStage.setScene(scene);

        primaryStage.show();

    }

*private* *void* showAlert(*String* *title*, *String* *message*, *AlertType* *type*) {

*Alert* alert = new Alert(type);

        alert.setTitle(title);

        alert.setHeaderText(null);

        alert.setContentText(message);

        alert.showAndWait();

    }

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

        launch(args);

    }

}

## Результати виконання програми

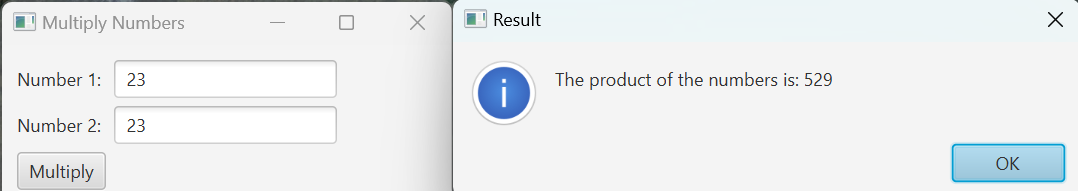


Рисунок 2.8 – «Множення чисел з використанням попередження»

# **ВИСНОВОК**

Під час виконання цієї лабораторної роботи ми ознайомилися з реалізацією GUI-застосунків за допомогою JavaFX, що дозволило нам створити програму з інтерактивним інтерфейсом для обробки даних з попередніх лабораторних робіт. Основні аспекти, які було засвоєно та вивчено: створення GUI засобами JavaFX, обробка подій, використання колекцій та серіалізація, робота з XML та використання JavaFX меню. Ця лабораторна робота дала нам можливість поєднати різноманітні аспекти програмування на Java, такі як робота з інтерфейсом користувача, обробка подій, колекції та робота з XML, та поглибити розуміння роботи з JavaFX та практично застосувати знання, отримані на попередніх лабораторних роботах.