МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Практична робота №3**

з дисципліни «Розробка крос-платформного програмного забезпечення»

тема: Взаємодія з базами даних. Параметризація запитів

**Виконав:**

Студент гр. БС-03

Затуловський Г. А.

**Перевірела:**

ст.викл. Сердаковський В. С.

ac. Рисін. С. В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2023

**Мета:** ознайомитися з принципами роботи з базами даних із програмного коду; усвідомити технології підключення програми до бази даних на прикладі її конкретної реалізації; сформувати вміння написання коду для виконання запитів до бази даних з використанням мови SQL; придбати навики використання параметризованих операторів; ознайомитися з можливостями пакетування операторів запитів SQL; організувати зберігання об'єктів, що розроблені у попередніх роботах, у базі даних та отримування їх з неї за критеріями користувача.

**Завдання:**

1. До проекту, розробленого у попередній роботі додати підключення до бази даних MySQL (SQLite тощо)\*. Шлях до бази даних має надаватися як параметр.

\* п.р. № 4, яка використовує код п.р. № 3, виконується на віддаленному сервері бази даних MySQL

2. При з'єднанні з базою даних у разі її відсутності створити її та таблиці даних та журналу (таблиця журналу має містити поля дати-часу та повідомлення; *дата-час має встановлюватися базою даних)*. При виникнення помилок створення таблиць вивести повідомлення та припинити роботу. Усі подальші повідомлення про помилки записувати в т.ч. в таблицю журналу.

3. Після з'єднання з базою даних завантажити зміст таблиць у відповідні фрейми.

4. Забезпечити можливість операцій над таблицею даних з використанням *підготованих* операторів:

4.1. при читанні даних з вхідного файлу завантаження (додавання, а не заміщення) їх до таблиці даних;

4.2. перегляду списку об'єктів, у тому числі відфільтрованого з умовою за критерієм користувача;

4.3. зміни значень полів вибраного об'єкта;

4.4. додавання у колекцію нового об'єкта;

4.5. видалення вибраного об'єкта з запитом підтвердження користувача;

4.6. видалення всіх об'єктів (очистки таблиці даних) з запитом підтвердження користувача.

## Варіанти:

| **Номер варіанту** | **Об'єкт** |
| --- | --- |
| 6. | Вистава: назва п'єси, автор, театр, дата вистави, кількість актів, ціна квитка, музикальний супровід (так/ні) |

## **Лістинг програми:**

**Spectacle.java:**

package com.example.work3;  
  
import javafx.scene.control.Alert;  
  
import java.text.SimpleDateFormat;  
import java.util.Date;  
public class Spectacle implements Comparable<Spectacle> {  
 private String piece\_name;  
 private String author;  
 private String theatre;  
 private Date performance\_date;  
 private Integer act\_quantity;  
 private Float prise\_ticket;  
 private Boolean music;  
 final static SimpleDateFormat *form* = new SimpleDateFormat("dd/MM/yyyy");  
 static final String[] *spectacle\_field* = {"Name","Author","Theatre","Performance\_Date","Quantity\_of\_acts","Ticket\_price","Music"};  
 static int[] *spectacle\_order* = new int[7];  
 public static void SETindexParam(String parameters)  
 {  
 String[] parameter = parameters.split(",");  
 for(int i = 0; i < *spectacle\_field*.length; ++i) {  
 for(int j = 0; j < parameter.length; ++j) {  
 if(parameter[j].equals(*spectacle\_field*[i])) {  
  
 *spectacle\_order*[i] = j;  
 break;  
 }  
  
 }  
 }  
 }  
  
 public Spectacle(){  
  
 }  
  
  
 public Spectacle(String parameters) throws Exception  
 {  
 if(parameters == null)  
 throw new WorkException("EMPTY PARAMETERS");  
  
 String[] parameter = parameters.split(",");  
  
  
  
 SET\_piece\_name(parameter[*spectacle\_order*[0]]);  
  
 SET\_author(parameter[*spectacle\_order*[1]]);  
  
 SET\_theatre(parameter[*spectacle\_order*[2]]);  
  
 SET\_performance\_date(parameter[*spectacle\_order*[3]]);  
  
 SET\_act\_quantity(parameter[*spectacle\_order*[4]]);  
  
 SET\_prise\_ticket(parameter[*spectacle\_order*[5]]);  
  
 SET\_music(parameter[*spectacle\_order*[6]]);  
  
  
 }  
  
  
  
 public static SimpleDateFormat GET\_date\_form()  
 {  
 return *form*;  
 }  
  
  
  
 public String GET\_piece\_name()  
 {  
 return piece\_name;  
 }  
  
  
 public void SET\_piece\_name(String piece\_name) throws Exception  
 {  
 if (piece\_name.length() == 0)  
 {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT PIECE NAME: ITS CANT BE EMPTY");  
 alert.showAndWait();  
  
  
 throw new Exception("INCORRECT PIECE NAME: ITS CANT BE EMPTY");  
  
  
 }  
 this.piece\_name = piece\_name;  
 }  
  
 public String GET\_author()  
 {  
 return author;  
 }  
  
 public void SET\_author(String author) throws Exception  
 {  
 if (author.length() == 0)  
 {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT AUTHOR: ITS CANT BE EMPTY");  
 alert.showAndWait();  
 throw new Exception("INCORRECT AUTHOR: ITS CANT BE EMPTY");  
 }  
 this.author = author;  
 }  
  
 public String GET\_theatre()  
 {  
 return theatre;  
 }  
  
 public void SET\_theatre(String theatre) throws Exception  
 {  
 if (theatre.length() == 0)  
 {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT THEATRE: ITS CANT BE EMPTY");  
 alert.showAndWait();  
 throw new Exception("INCORRECT THEATRE: ITS CANT BE EMPTY");  
 }  
 this.theatre = theatre;  
 }  
  
 public Date GET\_performance\_date()  
 {  
  
 return performance\_date;  
 }  
  
  
 public void SET\_performance\_date(String performance\_date) throws Exception {  
 try {  
 this.performance\_date = *form*.parse(performance\_date);  
 }  
 catch (Exception ex)  
 {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT PERFORMANCE DATE");  
 alert.showAndWait();  
 throw new Exception(ex.getLocalizedMessage() + " INCORRECT PERFORMANCE DATE");  
 }  
 }  
  
 public Integer GET\_act\_quantity()  
 {  
 return act\_quantity;  
 }  
  
 public void SET\_act\_quantity(String act\_quantity) throws Exception  
 {  
 if (act\_quantity.length() == 0)  
 {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT ACT QUANTITY: ITS CANT BE 0");  
 alert.showAndWait();  
 throw new Exception("INCORRECT ACT QUANTITY: ITS CANT BE 0");  
 }  
 try  
 {  
 this.act\_quantity = (Integer.*parseInt*(act\_quantity));  
 }  
 catch (Exception ex) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT ACT QUANTITY: MUST BE A NUMBER");  
 alert.showAndWait();  
 throw new Exception("INCORRECT ACT QUANTITY: MUST BE A NUMBER");  
 }  
 }  
  
 public Float GET\_prise\_ticket()  
 {  
  
 return prise\_ticket;  
 }  
  
 public void SET\_prise\_ticket(String prise\_ticket) throws Exception  
 {  
 try {  
 this.prise\_ticket = (Float.*parseFloat*(prise\_ticket));  
 if (prise\_ticket.length() == 0)  
 {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT PRICE OF TICKET: ITS CANT BE 0");  
 alert.showAndWait();  
 throw new Exception("INCORRECT PRICE OF TICKET: ITS CANT BE 0");  
 }  
 }  
 catch (NumberFormatException Exeption)  
 {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Error");  
 alert.setHeaderText(null);  
 alert.setContentText("INCORRECT PRICE OF TICKET: MUST BE A NUMBER");  
 alert.showAndWait();  
 throw new Exception("INCORRECT PRICE OF TICKET: MUST BE A NUMBER");  
 }  
  
  
 }  
  
 public Boolean GET\_music()  
 {  
 return music;  
 }  
 public void SET\_music(Boolean music)  
 {  
  
 this.music = music;  
 }  
  
 public void SET\_music(String music) throws Exception  
  
 {  
  
 SET\_music(Boolean.*parseBoolean*(music));  
  
 }  
  
  
 @Override  
 public int compareTo(Spectacle S) {  
 int res = this.piece\_name.compareTo(S.piece\_name);  
 if(res == 0)  
 {  
 Integer temp = this.act\_quantity;  
 res = temp.compareTo(S.act\_quantity);  
 }  
 return res;  
 }  
  
  
 @Override  
 public String toString() {  
 return "Name: " + piece\_name +  
 ", Author: " + author +  
 ", Theatre: " + theatre+  
 ", Performance date: " + performance\_date+  
 ", Quantity of acts: " + act\_quantity+  
 ", Ticket price: " + prise\_ticket+  
 ", Music: " + music ;  
  
 }  
  
  
 public String toJson() {  
 return ("\"Name\": \'" + piece\_name + '\'' +  
 ", \"Author\": \'" + author + '\'' +  
 ", \"Theatre\": \'" + theatre+ '\'' +  
 ", \"Performance date\": \'" + performance\_date+ '\'' +  
 ", \"Quantity of acts\": " + act\_quantity +  
 ", \"Ticket price\": " + prise\_ticket +  
 ", \"Music\": " + music);  
  
 }  
}

**Error\_Controller.java:**

ackage com.example.work3;  
  
import javafx.collections.FXCollections;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.AnchorPane;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
import java.sql.\*;  
import java.sql.Connection;  
  
import java.io.File;  
import java.io.IOException;  
  
import java.util.ArrayList;  
import java.util.stream.Collectors;  
  
  
  
  
public class Main\_Controller {  
  
  
 @FXML  
 private TextField Author\_input;  
  
 @FXML  
 private AnchorPane Main\_Window;  
  
 @FXML  
 private TextField Music\_input;  
  
 @FXML  
 private TextField Name\_input;  
  
 @FXML  
 private TextField Performance\_input;  
  
 @FXML  
 private TextField Price\_input;  
  
 @FXML  
 private TextField Quantity\_input;  
  
 @FXML  
 private TextField Theatre\_input;  
  
 @FXML  
 private ComboBox<String> box\_search;  
  
 @FXML  
 private Button button\_add;  
  
 @FXML  
 private Button button\_save;  
  
 @FXML  
 private Button button\_update;  
  
 @FXML  
 private Button button\_add\_elem;  
  
 @FXML  
 private Button button\_delete\_all\_elem;  
  
 @FXML  
 private Button button\_delete\_elem;  
  
 @FXML  
 private TextField search\_input;  
  
 @FXML  
 private ListView<Spectacle> view\_data;  
  
 private static Container\_Spectacle *container\_spect* = new Container\_Spectacle();  
 public static Stage *main\_stage* = null;  
 private static Spectacle *spect* = null;  
 private static Error\_Controller *error\_log* = null;  
 private static Connection *connection*;  
 public static String *url* ;  
 public static String *username*;  
 public static String *password* ;  
  
  
  
 private void Output\_CSV() {  
  
 FileChooser file\_choose = new FileChooser();  
 File file = file\_choose.showOpenDialog(*main\_stage*);  
  
 try {  
 String path = file.getAbsolutePath();  
 *error\_log*.addListView("READING DATAS FROM FILE '" + path + "'");  
  
 *container\_spect*.Output(path);  
  
 } catch (Exception ex) {  
  
 *error\_log*.addListView\_error(ex);  
  
 }  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 for(Spectacle spect : *container\_spect*.GET\_arraylist())  
 Spect\_add\_db((ID\_Spectacle) spect);  
 }  
  
 private void Error\_log() throws IOException {  
 try {  
 FXMLLoader fxml = new FXMLLoader(getClass().getResource("view\_error\_log.fxml"));  
 Stage stage = new Stage();  
 stage.setScene(new Scene(fxml.load()));  
 stage.setResizable(false);  
 stage.show();  
 *error\_log* = fxml.getController();  
 } catch (IOException exception) {  
  
 }  
 }  
  
 private void Input\_JSON() {  
 FileChooser file\_c = new FileChooser();  
 File file = file\_c.showSaveDialog(*main\_stage*);  
 if (file == null) {  
 return;  
 }  
 try {  
 String path = file.getAbsolutePath();  
 *container\_spect*.Input\_json(file.getAbsolutePath());  
 *error\_log*.addListView("SAVE DATA IN JSON FILE '" + path + "'");  
  
 } catch (Exception ex) {  
 *error\_log*.addListView\_error(ex);  
  
 }  
  
 }  
  
 private void Fill\_spect() {  
 if (*spect* == null)  
 return;  
  
 Name\_input.setText(*spect*.GET\_piece\_name());  
 Author\_input.setText(*spect*.GET\_author());  
 Theatre\_input.setText(*spect*.GET\_theatre());  
 Performance\_input.setText(Spectacle.*form*.format(*spect*.GET\_performance\_date()));  
 Quantity\_input.setText(*spect*.GET\_act\_quantity().toString());  
 Price\_input.setText(*spect*.GET\_prise\_ticket().toString());  
 Music\_input.setText(*spect*.GET\_music().toString());  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
  
 }  
  
 private void Spect\_select() {  
  
 *spect* = view\_data.getSelectionModel().getSelectedItem();  
 Fill\_spect();  
  
 }  
  
 private void Spect\_save\_db(ID\_Spectacle spectacle)  
 {  
 String save\_spect = "UPDATE spectacle\_tabl SET Name = ?,Author = ?,Theatre = ?,Performance\_Date = ?,Quantity\_of\_acts = ?,Ticket\_price = ?,Music = ? WHERE ID = ?";  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 PreparedStatement Stat = conn.prepareStatement(save\_spect))  
 {  
  
 Stat.setString(1,spectacle.GET\_piece\_name());  
 Stat.setString(2,spectacle.GET\_author());  
 Stat.setString(3,spectacle.GET\_theatre());  
 Stat.setString(4,spectacle.*GET\_date\_form*().format(spectacle.GET\_performance\_date()));  
 Stat.setInt(5,spectacle.GET\_act\_quantity());  
 Stat.setFloat(6,spectacle.GET\_prise\_ticket());  
 Stat.setString(7,spectacle.GET\_music().toString());  
 Stat.setInt(8,spectacle.GET\_id());  
 Stat.executeUpdate();  
 }  
 catch(SQLException ex)  
 {  
 *error\_log*.addListView(  
 "Failed to update show data.");  
 }  
 }  
 private void Spect\_save() {  
 if (*spect* == null)  
 return;  
 try {  
 Spectacle s = *spect*;  
 *spect*.SET\_piece\_name(Name\_input.getText());  
 *spect*.SET\_author(Author\_input.getText());  
 *spect*.SET\_theatre(Theatre\_input.getText());  
 *spect*.SET\_performance\_date(Performance\_input.getText());  
 *spect*.SET\_act\_quantity(Quantity\_input.getText());  
 *spect*.SET\_prise\_ticket(Price\_input.getText());  
 *spect*.SET\_music(Music\_input.getText());  
 *error\_log*.addListView("SPECTACLE DATA ID("+ ((ID\_Spectacle)*spect*).GET\_id()+") HAS BEEN CHANGED");  
  
 } catch (Exception ex) {  
 *error\_log*.addListView\_error(ex);  
  
 }  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 Fill\_spect();  
 Spect\_save\_db((ID\_Spectacle) *spect*);  
 }  
  
 private void Spect\_search() {  
 ArrayList<Spectacle> arr = *container\_spect*.GET\_arraylist();  
 if (search\_input.getText().equals("")) {  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 return;  
 }  
 switch (box\_search.getSelectionModel().getSelectedItem().toString()) {  
 case ("Name"):  
  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_piece\_name().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Name'");  
  
 break;  
 case ("Author"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_author().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Author'");  
  
 break;  
 case ("Theatre"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_theatre().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Theatre'");  
  
 break;  
 case ("Performance\_Date"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_performance\_date().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Performance\_Date'");  
  
 break;  
 case ("Ticket\_price"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_prise\_ticket().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Ticket\_price'");  
  
 break;  
 case ("Music"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_music().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Music'");  
  
 break;  
 case ("Quantity\_of\_acts"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_act\_quantity().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Quantity\_of\_acts'");  
  
 break;  
 }  
 view\_data.setItems(FXCollections.*observableList*(arr));  
  
 }  
 private void Spect\_delete\_db(int ID)  
 {  
 String delete\_spectacle = "DELETE FROM spectacle\_tabl WHERE ID = ?";  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 PreparedStatement Stat = conn.prepareStatement(delete\_spectacle))  
 {  
 Stat.setInt(1,ID);  
 Stat.executeUpdate();  
 }  
 catch (SQLException ex)  
 {  
 *error\_log*.addListView("Failed to delete show data.");  
 }  
 }  
 private void Spect\_delete() {  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*);  
  
 alert.setTitle("Видалення об'єкту");  
 alert.setHeaderText("Видалити обраний об'єкт з бази?");  
 if (alert.showAndWait().get() != ButtonType.*OK*) {  
 return;  
 }  
  
 if (*spect* == null) {  
 return;  
 }  
 *error\_log*.addListView("DELETE DATA OF SPECTALCE ID ("+((ID\_Spectacle)*spect*).GET\_id()+")");  
 Spect\_delete\_db(((ID\_Spectacle)*spect*).GET\_id());  
 *container\_spect*.GET\_arraylist().remove(*spect*);  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 *spect* = null;  
  
 }  
  
 private void Delete\_all() {  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*);  
  
 alert.setTitle("Очищення бази даних");  
 alert.setHeaderText("Видалити всі об'єкти з бази?");  
 if (alert.showAndWait().get() != ButtonType.*OK*) {  
 return;  
 }  
 *error\_log*.addListView("DELETING ALL DATA FROM THE SPECTACLE DATABASE");  
 *container\_spect*.GET\_arraylist().clear();  
 String delete\_all\_spect = "DROP TABLE IF EXISTS spectacle\_tabl";  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 Statement Stat = conn.createStatement())  
 {  
 Stat.executeUpdate(delete\_all\_spect);  
  
 }  
 catch(SQLException e)  
 {  
 *error\_log*.addListView("Failed to delete all spectacle data");  
 }  
  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
  
 }  
 private void Spect\_add\_db(ID\_Spectacle spectacle)  
 {  
 String add\_spectacle = "INSERT INTO spectacle\_tabl(ID,Name,Author,Theatre,Performance\_Date,Quantity\_of\_acts,Ticket\_price,Music) VALUES (?,?,?,?,?,?,?,?)";  
  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 PreparedStatement Stat = conn.prepareStatement(add\_spectacle))  
 {  
 Stat.setInt(1,spectacle.GET\_id());  
 Stat.setString(2,spectacle.GET\_piece\_name());  
 Stat.setString(3,spectacle.GET\_author());  
 Stat.setString(4,spectacle.GET\_theatre());  
 Stat.setString(5,spectacle.*GET\_date\_form*().format(spectacle.GET\_performance\_date()));  
 Stat.setInt(6,spectacle.GET\_act\_quantity());  
 Stat.setFloat(7,spectacle.GET\_prise\_ticket());  
 Stat.setString(8,spectacle.GET\_music().toString());  
 Stat.executeUpdate();  
  
 }  
 catch(SQLException e)  
 {  
  
 }  
 }  
 public void Spect\_adding()  
 {  
  
 Spect\_add();  
 *error\_log*.addListView("ADDING NEW SPECTACLE DATA WITH ID ("+((ID\_Spectacle)*spect*).GET\_id()+")");  
  
 }  
 private void Spect\_add() {  
  
 *spect* = null;  
 Spectacle.*SETindexParam*("Name,Author,Theatre,Performance\_Date,Quantity\_of\_acts,Ticket\_price,Music");  
 StringBuilder new\_spectacle = new StringBuilder();  
 new\_spectacle.append(Name\_input.getText()).append(',').  
 append(Author\_input.getText()).append(',').  
 append(Theatre\_input.getText()).append(',').  
 append(Performance\_input.getText()).append(',').  
 append(Quantity\_input.getText()).append(',').  
 append(Price\_input.getText()).append(',').  
 append(Music\_input.getText());  
 try {  
 ID\_Spectacle spectacle = new ID\_Spectacle(new\_spectacle.toString());  
 Spect\_add\_db(spectacle);  
 *container\_spect*.GET\_arraylist().add(spectacle);  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 *spect* = spectacle;  
 view\_data.getSelectionModel().selectLast();  
 } catch (Exception ex) {  
 *error\_log*.addListView\_error(ex);  
  
 }  
  
  
 }  
  
  
 @FXML  
 void initialize() throws IOException {  
  
 Error\_log();  
 Database\_loading();  
  
 box\_search.getItems().addAll("Name", "Author", "Theatre", "Performance\_Date", "Quantity\_of\_acts", "Ticket\_price", "Music", "-");  
 button\_update.setOnAction(actionEvent -> Output\_CSV());  
 button\_add.setOnAction(actionEvent -> Input\_JSON());  
 view\_data.setOnMouseClicked(mouseEvent -> Spect\_select());  
 button\_save.setOnAction(actionEvent -> Spect\_save());  
 box\_search.setOnAction(actionEvent -> Spect\_search());  
 button\_delete\_elem.setOnAction(actionEvent -> Spect\_delete());  
 button\_delete\_all\_elem.setOnAction(actionEvent -> Delete\_all());  
 button\_add\_elem.setOnAction(actionEvent -> Spect\_adding());  
 }  
  
 static public Connection Connect\_db(String url, String username, String password) {  
 Connection connection = null;  
 try {  
 connection = DriverManager.*getConnection*(url, username, password);  
  
 } catch (SQLException e) {  
  
 *error\_log*.addListView("Failed to connect to databese spectacle data");  
 }  
  
 return connection;  
  
 }  
  
 private void CreateTable\_db(String url, String username, String password) {  
 String create\_spectacle = """  
 CREATE TABLE IF NOT EXISTS spectacle\_tabl( ID integer PRIMARY KEY, Name varchar(45),Author varchar(45),Theatre varchar(45),Performance\_Date varchar(45),Quantity\_of\_acts int,Ticket\_price float,Music varchar(45));""";  
 String create\_logs = """  
 CREATE TABLE IF NOT EXISTS log\_tabl(data varchar(45) PRIMARY KEY,message varchar(150));""";  
 try {  
 Connection conn = *Connect\_db*(url, username, password);  
 Statement stmt = conn.createStatement();  
 stmt.execute(create\_spectacle);  
 stmt.execute(create\_logs);  
  
 } catch (SQLException ex) {  
 }  
 }  
  
 private void Select\_db(String url, String username, String password, String filters)  
 {  
 String select\_spect = "SELECT \* FROM spectacle\_tabl" + filters;  
 try(Connection conn = *Connect\_db*(url, username, password);  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery(select\_spect)){  
 Spectacle.*SETindexParam*("Name,Author,Theatre,Performance\_Date,Quantity\_of\_acts,Ticket\_price,Music");  
 while(rs.next())  
 {  
 StringBuilder param\_str = new StringBuilder();  
 for (String param\_name : Spectacle.*spectacle\_field*) {  
 param\_str.append(rs.getString(param\_name)).append(',');  
 }  
 ID\_Spectacle spectacle = new ID\_Spectacle(param\_str.toString());  
 spectacle.SET\_ID(rs.getInt("id"));  
 *container\_spect*.GET\_arraylist().add(spectacle);  
  
  
 }  
  
 }  
 catch(SQLException ex)  
 {  
  
 }  
 catch (Exception ex)  
 {  
 *error\_log*.addListView\_error(ex);  
 }  
  
 }  
  
 private void Set\_maximum\_ID\_db(String url, String username, String password)  
 {  
 String max\_id = "SELECT MAX(id) From spectacle\_tabl";  
 try(Connection conn = *Connect\_db*(url, username, password);  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery(max\_id))  
 {  
 rs.next();  
 ID\_Spectacle.*SET\_ID\_start*(rs.getInt(1)+1);  
  
 }  
  
 catch(SQLException ex)  
 {  
 *error\_log*.addListView("Failed to find max id of spectacle ID");  
 }  
  
 }  
  
 private void Select\_log\_db(String url, String username, String password)  
 {  
  
 String select\_logs = "SELECT \* FROM log\_tabl";  
  
 try(Connection conn = *Connect\_db*(url, username, password);  
 Statement Stat = conn.createStatement();  
 ResultSet rs = Stat.executeQuery(select\_logs))  
  
 {  
 while(rs.next())  
 {  
  
  
  
  
 try{*error\_log*.addListView\_db(rs.getString(2),rs.getString(1));}  
 catch(SQLException ignore)  
 {  
  
 }  
 catch (Exception ex)  
 {  
 *error\_log*.addListView\_error(ex);  
  
 }  
 }  
 }  
 catch (SQLException ex)  
 {  
  
 }  
 }  
  
 private void Database\_loading()  
 {  
  
 try {  
 Class.*forName*("com.mysql.cj.Driver");  
 }  
 catch(ClassNotFoundException ex)  
 {  
  
 }  
 *Connect\_db*(*url*,*username*,*password*);  
 CreateTable\_db(*url*,*username*,*password*);  
 Select\_db(*url*,*username*,*password*, "");  
 Set\_maximum\_ID\_db(*url*,*username*,*password*);  
 Select\_log\_db(*url*,*username*,*password*);  
  
 }  
}

**Container\_Spectacle.java:**

package com.example.work3;  
  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Collections;  
  
  
public class Container\_Spectacle {  
 public ArrayList<Spectacle> Constain = new ArrayList<>();  
  
 public Container\_Spectacle(){}  
  
 public void Output(String name) throws Exception {  
 try{  
 FileReader file = new FileReader (name);  
 BufferedReader buff\_file = new BufferedReader(file);  
 String head = buff\_file.readLine();  
 Spectacle.*SETindexParam*(head);  
 String spect\_line;  
 int counter = 1;  
 while((spect\_line = buff\_file.readLine())!=null)  
 {  
 counter+=1;  
 try{  
 Spectacle spect = new ID\_Spectacle(spect\_line);  
 Constain.add(spect);  
  
 }  
  
 catch(WorkException Exeption){  
 Exeption.FileError(" ERROR IN READING CSV FILE IN ROW #" + counter +" : "+ Exeption.getLocalizedMessage());  
 }  
  
  
  
 }  
  
 }  
 catch(IOException Exeption)  
 {  
 throw new WorkException(Exeption.getLocalizedMessage());  
 }  
  
 }  
  
  
 public void Input\_json(String name) throws WorkException {  
 try(FileWriter Input\_in\_file = new FileWriter(name)) {  
 String text = "[\n";  
 for(int i = 0; i < Constain.size(); i++)  
 {  
 text = text + " {" + (Constain.get(i).toJson()) + " }";  
 if(i != (Constain.size() - 1) )  
 text = text + ",";  
 text = text + "\n";  
 }  
 text += "]";  
  
 Input\_in\_file.write(text);  
 Input\_in\_file.flush();  
 }  
 catch(IOException Exeption)  
 {  
 throw new WorkException(Exeption.getLocalizedMessage());  
 }  
 }  
 public ArrayList<Spectacle> GET\_arraylist()  
 {  
 return Constain;  
 }  
  
 @Override  
 public String toString() {  
 return "Spectacle : { " + Constain + "}";  
 }  
  
  
 public void Sort(){  
 Collections.*sort*(Constain);  
 }  
}

**ID\_Spectacle.java:**

package com.example.work3;  
public class ID\_Spectacle extends Spectacle {  
 private static int *id\_static* = 0;  
 private int id = 0;  
 public ID\_Spectacle() {  
 this.id = ++*id\_static*;  
 }  
  
 public ID\_Spectacle(String parametrs) throws Exception {  
 super(parametrs);  
 this.id = ++*id\_static*;  
 }  
 public int GET\_id() {  
 return id;  
 }  
  
 public void SET\_ID(Integer id){  
 this.id=id;  
 }  
  
 static public void SET\_ID\_start(Integer id\_start)  
 {  
 if (id\_start > *id\_static*)  
 id\_start = *id\_static*;  
 }  
 @Override  
 public String toString() {  
  
 return "{ID: " + GET\_id() +", "+ super.toString()+"}\n";  
 }  
  
 @Override  
 public String toJson(){  
 return ("\"ID\": " + id + ", " + super.toJson());  
 }  
  
  
}

**WorkException.java:**

package com.example.work3;  
  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.Writer;  
import java.time.LocalDateTime;  
  
public class WorkException extends Exception  
{  
  
 private static String *error\_fileName*= "Errors\_message.txt";;  
 private String info;  
  
  
  
  
 public WorkException(String info)  
 {  
 super(info);  
 *error\_fileName*= "Errors\_message.txt";  
  
  
 }  
  
  
 public void FileError(String error\_info)  
 {  
  
 try  
 {  
  
 Writer Input = new FileWriter("Errors\_message.txt",true);  
 String error = LocalDateTime.*now*() + ",TYPE OF ERROR: "+error\_info+"\n";  
 Input.write(error);  
 Input.close();  
  
 }  
 catch(IOException Exeption)  
 {  
  
 System.*out*.println(Exeption.getLocalizedMessage());  
 }  
 catch(NullPointerException Exeption)  
 {  
  
 System.*out*.println(Exeption.getLocalizedMessage());  
 }  
  
  
  
 }  
}

**Main\_Application:**

package com.example.work3;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class Main\_Application extends Application {  
  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(Main\_Application.class.getResource("view-main\_window.fxml"));  
 Scene scene = new Scene(fxmlLoader.load(), 706, 483);  
 stage.setResizable(false);  
 stage.setScene(scene);  
 Main\_Controller.*main\_stage* = stage;  
  
 stage.show();  
  
 }  
  
 public static void main(String[] args) {  
 Main\_Controller.*url* = args[0];  
 Main\_Controller.*username* =args[1];  
 Main\_Controller.*password* = args[2];  
 *launch*();  
 }  
}

**Main\_Controller:**

package com.example.work3;  
  
import javafx.collections.FXCollections;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.layout.AnchorPane;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
import java.sql.\*;  
import java.sql.Connection;  
  
import java.io.File;  
import java.io.IOException;  
  
import java.util.ArrayList;  
import java.util.stream.Collectors;  
  
  
  
  
public class Main\_Controller {  
  
  
 @FXML  
 private TextField Author\_input;  
  
 @FXML  
 private AnchorPane Main\_Window;  
  
 @FXML  
 private TextField Music\_input;  
  
 @FXML  
 private TextField Name\_input;  
  
 @FXML  
 private TextField Performance\_input;  
  
 @FXML  
 private TextField Price\_input;  
  
 @FXML  
 private TextField Quantity\_input;  
  
 @FXML  
 private TextField Theatre\_input;  
  
 @FXML  
 private ComboBox<String> box\_search;  
  
 @FXML  
 private Button button\_add;  
  
 @FXML  
 private Button button\_save;  
  
 @FXML  
 private Button button\_update;  
  
 @FXML  
 private Button button\_add\_elem;  
  
 @FXML  
 private Button button\_delete\_all\_elem;  
  
 @FXML  
 private Button button\_delete\_elem;  
  
 @FXML  
 private TextField search\_input;  
  
 @FXML  
 private ListView<Spectacle> view\_data;  
  
 private static Container\_Spectacle *container\_spect* = new Container\_Spectacle();  
 public static Stage *main\_stage* = null;  
 private static Spectacle *spect* = null;  
 private static Error\_Controller *error\_log* = null;  
 private static Connection *connection*;  
 public static String *url* ;  
 public static String *username*;  
 public static String *password* ;  
  
  
  
 private void Output\_CSV() {  
  
 FileChooser file\_choose = new FileChooser();  
 File file = file\_choose.showOpenDialog(*main\_stage*);  
  
 try {  
 String path = file.getAbsolutePath();  
 *error\_log*.addListView("READING DATAS FROM FILE '" + path + "'");  
  
 *container\_spect*.Output(path);  
  
 } catch (Exception ex) {  
  
 *error\_log*.addListView\_error(ex);  
  
 }  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 for(Spectacle spect : *container\_spect*.GET\_arraylist())  
 Spect\_add\_db((ID\_Spectacle) spect);  
 }  
  
 private void Error\_log() throws IOException {  
 try {  
 FXMLLoader fxml = new FXMLLoader(getClass().getResource("view\_error\_log.fxml"));  
 Stage stage = new Stage();  
 stage.setScene(new Scene(fxml.load()));  
 stage.setResizable(false);  
 stage.show();  
 *error\_log* = fxml.getController();  
 } catch (IOException exception) {  
  
 }  
 }  
  
 private void Input\_JSON() {  
 FileChooser file\_c = new FileChooser();  
 File file = file\_c.showSaveDialog(*main\_stage*);  
 if (file == null) {  
 return;  
 }  
 try {  
 String path = file.getAbsolutePath();  
 *container\_spect*.Input\_json(file.getAbsolutePath());  
 *error\_log*.addListView("SAVE DATA IN JSON FILE '" + path + "'");  
  
 } catch (Exception ex) {  
 *error\_log*.addListView\_error(ex);  
  
 }  
  
 }  
  
 private void Fill\_spect() {  
 if (*spect* == null)  
 return;  
  
 Name\_input.setText(*spect*.GET\_piece\_name());  
 Author\_input.setText(*spect*.GET\_author());  
 Theatre\_input.setText(*spect*.GET\_theatre());  
 Performance\_input.setText(Spectacle.*form*.format(*spect*.GET\_performance\_date()));  
 Quantity\_input.setText(*spect*.GET\_act\_quantity().toString());  
 Price\_input.setText(*spect*.GET\_prise\_ticket().toString());  
 Music\_input.setText(*spect*.GET\_music().toString());  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
  
 }  
  
 private void Spect\_select() {  
  
 *spect* = view\_data.getSelectionModel().getSelectedItem();  
 Fill\_spect();  
  
 }  
  
 private void Spect\_save\_db(ID\_Spectacle spectacle)  
 {  
 String save\_spect = "UPDATE spectacle\_tabl SET Name = ?,Author = ?,Theatre = ?,Performance\_Date = ?,Quantity\_of\_acts = ?,Ticket\_price = ?,Music = ? WHERE ID = ?";  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 PreparedStatement Stat = conn.prepareStatement(save\_spect))  
 {  
  
 Stat.setString(1,spectacle.GET\_piece\_name());  
 Stat.setString(2,spectacle.GET\_author());  
 Stat.setString(3,spectacle.GET\_theatre());  
 Stat.setString(4,spectacle.*GET\_date\_form*().format(spectacle.GET\_performance\_date()));  
 Stat.setInt(5,spectacle.GET\_act\_quantity());  
 Stat.setFloat(6,spectacle.GET\_prise\_ticket());  
 Stat.setString(7,spectacle.GET\_music().toString());  
 Stat.setInt(8,spectacle.GET\_id());  
 Stat.executeUpdate();  
 }  
 catch(SQLException ex)  
 {  
 *error\_log*.addListView(  
 "Failed to update show data.");  
 }  
 }  
 private void Spect\_save() {  
 if (*spect* == null)  
 return;  
 try {  
 Spectacle s = *spect*;  
 *spect*.SET\_piece\_name(Name\_input.getText());  
 *spect*.SET\_author(Author\_input.getText());  
 *spect*.SET\_theatre(Theatre\_input.getText());  
 *spect*.SET\_performance\_date(Performance\_input.getText());  
 *spect*.SET\_act\_quantity(Quantity\_input.getText());  
 *spect*.SET\_prise\_ticket(Price\_input.getText());  
 *spect*.SET\_music(Music\_input.getText());  
 *error\_log*.addListView("SPECTACLE DATA ID("+ ((ID\_Spectacle)*spect*).GET\_id()+") HAS BEEN CHANGED");  
  
 } catch (Exception ex) {  
 *error\_log*.addListView\_error(ex);  
  
 }  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 Fill\_spect();  
 Spect\_save\_db((ID\_Spectacle) *spect*);  
 }  
  
 private void Spect\_search() {  
 ArrayList<Spectacle> arr = *container\_spect*.GET\_arraylist();  
 if (search\_input.getText().equals("")) {  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 return;  
 }  
 switch (box\_search.getSelectionModel().getSelectedItem().toString()) {  
 case ("Name"):  
  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_piece\_name().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Name'");  
  
 break;  
 case ("Author"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_author().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Author'");  
  
 break;  
 case ("Theatre"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_theatre().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Theatre'");  
  
 break;  
 case ("Performance\_Date"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_performance\_date().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Performance\_Date'");  
  
 break;  
 case ("Ticket\_price"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_prise\_ticket().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Ticket\_price'");  
  
 break;  
 case ("Music"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_music().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Music'");  
  
 break;  
 case ("Quantity\_of\_acts"):  
 arr = (ArrayList<Spectacle>) *container\_spect*.GET\_arraylist().stream().filter(ar -> ar.GET\_act\_quantity().toString().toLowerCase().contains(search\_input.getText().toLowerCase())).collect(Collectors.*toList*());  
 *error\_log*.addListView("SEARCH FOR '" + search\_input.getText() + "' IN CASE 'Quantity\_of\_acts'");  
  
 break;  
 }  
 view\_data.setItems(FXCollections.*observableList*(arr));  
  
 }  
 private void Spect\_delete\_db(int ID)  
 {  
 String delete\_spectacle = "DELETE FROM spectacle\_tabl WHERE ID = ?";  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 PreparedStatement Stat = conn.prepareStatement(delete\_spectacle))  
 {  
 Stat.setInt(1,ID);  
 Stat.executeUpdate();  
 }  
 catch (SQLException ex)  
 {  
 *error\_log*.addListView("Failed to delete show data.");  
 }  
 }  
 private void Spect\_delete() {  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*);  
  
 alert.setTitle("Видалення об'єкту");  
 alert.setHeaderText("Видалити обраний об'єкт з бази?");  
 if (alert.showAndWait().get() != ButtonType.*OK*) {  
 return;  
 }  
  
 if (*spect* == null) {  
 return;  
 }  
 *error\_log*.addListView("DELETE DATA OF SPECTALCE ID ("+((ID\_Spectacle)*spect*).GET\_id()+")");  
 Spect\_delete\_db(((ID\_Spectacle)*spect*).GET\_id());  
 *container\_spect*.GET\_arraylist().remove(*spect*);  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 *spect* = null;  
  
 }  
  
 private void Delete\_all() {  
 Alert alert = new Alert(Alert.AlertType.*CONFIRMATION*);  
  
 alert.setTitle("Очищення бази даних");  
 alert.setHeaderText("Видалити всі об'єкти з бази?");  
 if (alert.showAndWait().get() != ButtonType.*OK*) {  
 return;  
 }  
 *error\_log*.addListView("DELETING ALL DATA FROM THE SPECTACLE DATABASE");  
 *container\_spect*.GET\_arraylist().clear();  
 String delete\_all\_spect = "DROP TABLE IF EXISTS spectacle\_tabl";  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 Statement Stat = conn.createStatement())  
 {  
 Stat.executeUpdate(delete\_all\_spect);  
  
 }  
 catch(SQLException e)  
 {  
 *error\_log*.addListView("Failed to delete all spectacle data");  
 }  
  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
  
 }  
 private void Spect\_add\_db(ID\_Spectacle spectacle)  
 {  
 String add\_spectacle = "INSERT INTO spectacle\_tabl(ID,Name,Author,Theatre,Performance\_Date,Quantity\_of\_acts,Ticket\_price,Music) VALUES (?,?,?,?,?,?,?,?)";  
  
 try(Connection conn = *Connect\_db*(*url*, *username*, *password*);  
 PreparedStatement Stat = conn.prepareStatement(add\_spectacle))  
 {  
 Stat.setInt(1,spectacle.GET\_id());  
 Stat.setString(2,spectacle.GET\_piece\_name());  
 Stat.setString(3,spectacle.GET\_author());  
 Stat.setString(4,spectacle.GET\_theatre());  
 Stat.setString(5,spectacle.*GET\_date\_form*().format(spectacle.GET\_performance\_date()));  
 Stat.setInt(6,spectacle.GET\_act\_quantity());  
 Stat.setFloat(7,spectacle.GET\_prise\_ticket());  
 Stat.setString(8,spectacle.GET\_music().toString());  
 Stat.executeUpdate();  
  
 }  
 catch(SQLException e)  
 {  
  
 }  
 }  
 public void Spect\_adding()  
 {  
  
 Spect\_add();  
 *error\_log*.addListView("ADDING NEW SPECTACLE DATA WITH ID ("+((ID\_Spectacle)*spect*).GET\_id()+")");  
  
 }  
 private void Spect\_add() {  
  
 *spect* = null;  
 Spectacle.*SETindexParam*("Name,Author,Theatre,Performance\_Date,Quantity\_of\_acts,Ticket\_price,Music");  
 StringBuilder new\_spectacle = new StringBuilder();  
 new\_spectacle.append(Name\_input.getText()).append(',').  
 append(Author\_input.getText()).append(',').  
 append(Theatre\_input.getText()).append(',').  
 append(Performance\_input.getText()).append(',').  
 append(Quantity\_input.getText()).append(',').  
 append(Price\_input.getText()).append(',').  
 append(Music\_input.getText());  
 try {  
 ID\_Spectacle spectacle = new ID\_Spectacle(new\_spectacle.toString());  
 Spect\_add\_db(spectacle);  
 *container\_spect*.GET\_arraylist().add(spectacle);  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 *spect* = spectacle;  
 view\_data.getSelectionModel().selectLast();  
 } catch (Exception ex) {  
 *error\_log*.addListView\_error(ex);  
  
 }  
  
  
 }  
  
  
 @FXML  
 void initialize() throws IOException {  
  
 Error\_log();  
 Database\_loading();  
  
 box\_search.getItems().addAll("Name", "Author", "Theatre", "Performance\_Date", "Quantity\_of\_acts", "Ticket\_price", "Music", "-");  
 button\_update.setOnAction(actionEvent -> Output\_CSV());  
 button\_add.setOnAction(actionEvent -> Input\_JSON());  
 view\_data.setOnMouseClicked(mouseEvent -> Spect\_select());  
 button\_save.setOnAction(actionEvent -> Spect\_save());  
 box\_search.setOnAction(actionEvent -> Spect\_search());  
 button\_delete\_elem.setOnAction(actionEvent -> Spect\_delete());  
 button\_delete\_all\_elem.setOnAction(actionEvent -> Delete\_all());  
 button\_add\_elem.setOnAction(actionEvent -> Spect\_adding());  
 }  
  
 static public Connection Connect\_db(String url, String username, String password) {  
 Connection connection = null;  
 try {  
 connection = DriverManager.*getConnection*(url, username, password);  
  
 } catch (SQLException e) {  
  
 *error\_log*.addListView("Failed to connect to databese spectacle data");  
 }  
  
 return connection;  
  
 }  
  
 private void CreateTable\_db(String url, String username, String password) {  
 String create\_spectacle = """  
 CREATE TABLE IF NOT EXISTS spectacle\_tabl( ID integer PRIMARY KEY, Name varchar(45),Author varchar(45),Theatre varchar(45),Performance\_Date varchar(45),Quantity\_of\_acts int,Ticket\_price float,Music varchar(45));""";  
 String create\_logs = """  
 CREATE TABLE IF NOT EXISTS log\_tabl(data varchar(45) PRIMARY KEY,message varchar(150));""";  
 try {  
 Connection conn = *Connect\_db*(url, username, password);  
 Statement stmt = conn.createStatement();  
 stmt.execute(create\_spectacle);  
 stmt.execute(create\_logs);  
  
 } catch (SQLException ex) {  
 }  
 }  
  
 private void Select\_db(String url, String username, String password, String filters)  
 {  
 String select\_spect = "SELECT \* FROM spectacle\_tabl" + filters;  
 try(Connection conn = *Connect\_db*(url, username, password);  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery(select\_spect)){  
 Spectacle.*SETindexParam*("Name,Author,Theatre,Performance\_Date,Quantity\_of\_acts,Ticket\_price,Music");  
 while(rs.next())  
 {  
 StringBuilder param\_str = new StringBuilder();  
 for (String param\_name : Spectacle.*spectacle\_field*) {  
 param\_str.append(rs.getString(param\_name)).append(',');  
 }  
 ID\_Spectacle spectacle = new ID\_Spectacle(param\_str.toString());  
 spectacle.SET\_ID(rs.getInt("id"));  
 *container\_spect*.GET\_arraylist().add(spectacle);  
  
  
 }  
  
 }  
 catch(SQLException ex)  
 {  
  
 }  
 catch (Exception ex)  
 {  
 *error\_log*.addListView\_error(ex);  
 }  
  
 }  
  
 private void Set\_maximum\_ID\_db(String url, String username, String password)  
 {  
 String max\_id = "SELECT MAX(id) From spectacle\_tabl";  
 try(Connection conn = *Connect\_db*(url, username, password);  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery(max\_id))  
 {  
 rs.next();  
 ID\_Spectacle.*SET\_ID\_start*(rs.getInt(1)+1);  
  
 }  
  
 catch(SQLException ex)  
 {  
 *error\_log*.addListView("Failed to find max id of spectacle ID");  
 }  
  
 }  
  
 private void Select\_log\_db(String url, String username, String password)  
 {  
  
 String select\_logs = "SELECT \* FROM log\_tabl";  
  
 try(Connection conn = *Connect\_db*(url, username, password);  
 Statement Stat = conn.createStatement();  
 ResultSet rs = Stat.executeQuery(select\_logs))  
  
 {  
 while(rs.next())  
 {  
  
  
  
  
 try{*error\_log*.addListView\_db(rs.getString(2),rs.getString(1));}  
 catch(SQLException ignore)  
 {  
  
 }  
 catch (Exception ex)  
 {  
 *error\_log*.addListView\_error(ex);  
  
 }  
 }  
 }  
 catch (SQLException ex)  
 {  
  
 }  
 }  
  
 private void Database\_loading()  
 {  
  
 try {  
 Class.*forName*("com.mysql.cj.Driver");  
 }  
 catch(ClassNotFoundException ex)  
 {  
  
 }  
 *Connect\_db*(*url*,*username*,*password*);  
 CreateTable\_db(*url*,*username*,*password*);  
 Select\_db(*url*,*username*,*password*, "");  
 Set\_maximum\_ID\_db(*url*,*username*,*password*);  
 Select\_log\_db(*url*,*username*,*password*);  
  
 }  
}

**view\_error\_log.fxml:**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ListView?>  
<?import javafx.scene.layout.AnchorPane?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="391.0" prefWidth="601.0" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.work3.Error\_Controller">  
 <children>  
 <ListView fx:id="view\_error" layoutX="17.0" layoutY="27.0" prefHeight="309.0" prefWidth="560.0" />  
 <Button fx:id="save\_button" layoutX="52.0" layoutY="352.0" mnemonicParsing="false" text="Save" />  
 <Button fx:id="delete\_button" layoutX="108.0" layoutY="352.0" mnemonicParsing="false" text="Delete" />  
 </children>  
</AnchorPane>

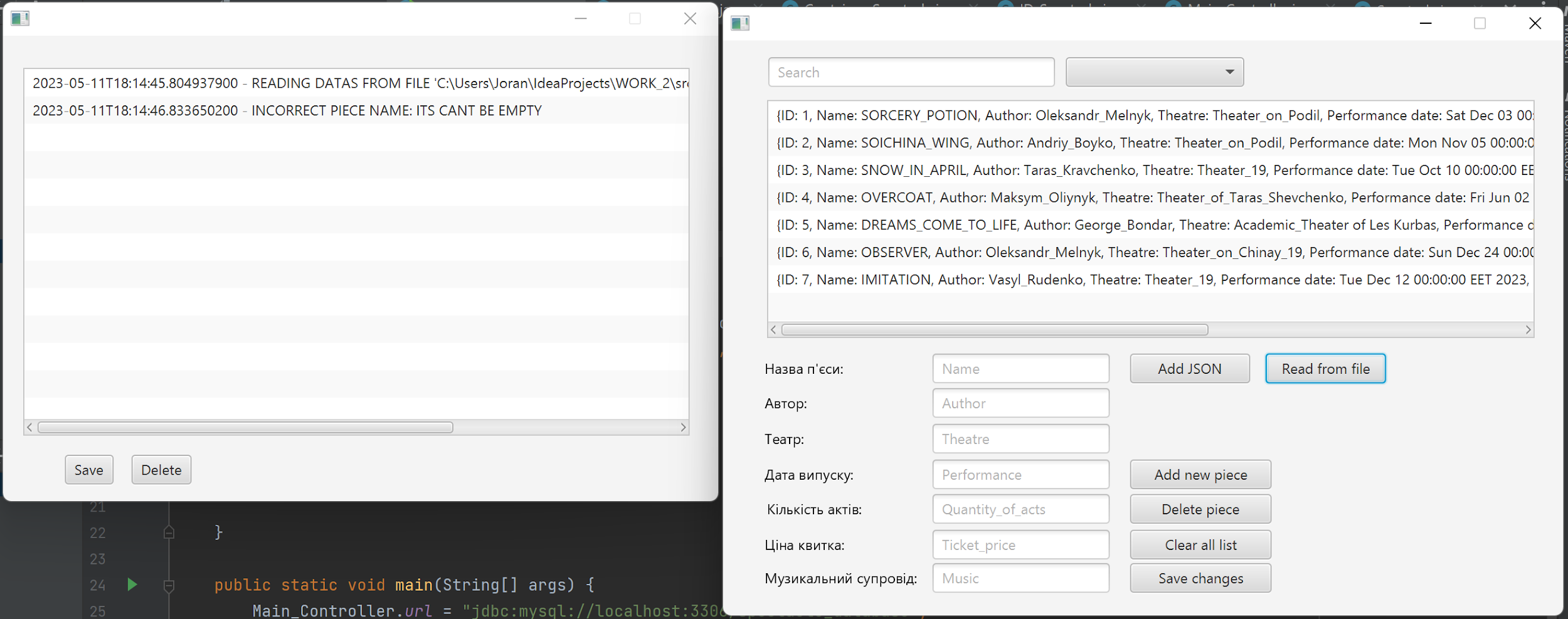
**view-main\_window.fxml:**

<?xml version="1.0" encoding="UTF-8"?>  
  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ComboBox?>  
<?import javafx.scene.control.ListView?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.text.Text?>  
  
<AnchorPane fx:id="Main\_Window" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="483.0" prefWidth="706.0" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.work3.Main\_Controller">  
 <children>  
 <Text layoutX="35.0" layoutY="280.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Назва п'єси:" wrappingWidth="132.13671875" />  
 <Text layoutX="35.0" layoutY="309.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Автор:" wrappingWidth="119.13671875" />  
 <Text layoutX="35.0" layoutY="339.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Театр:" wrappingWidth="119.13671875" />  
 <Text layoutX="35.0" layoutY="369.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Дата випуску:" wrappingWidth="119.13671875" />  
 <Text layoutX="37.0" layoutY="398.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Кількість актів:" wrappingWidth="186.63671875" />  
 <Text layoutX="35.0" layoutY="428.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Ціна квитка:" wrappingWidth="132.13671875" />  
 <Text layoutX="35.0" layoutY="456.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Музикальний супровід:" wrappingWidth="132.13671875" />  
 <Button fx:id="button\_update" layoutX="456.0" layoutY="263.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="101.0" text="Read from file" />  
 <Button fx:id="button\_add" layoutX="342.0" layoutY="263.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="101.0" text="Add JSON" />  
 <TextField fx:id="Name\_input" layoutX="176.0" layoutY="263.0" promptText="Name" />  
 <TextField fx:id="Author\_input" layoutX="176.0" layoutY="292.0" promptText="Author" />  
 <TextField fx:id="Theatre\_input" layoutX="176.0" layoutY="322.0" promptText="Theatre" />  
 <TextField fx:id="Performance\_input" layoutX="176.0" layoutY="352.0" promptText="Performance" />  
 <TextField fx:id="Quantity\_input" layoutX="176.0" layoutY="381.0" promptText="Quantity\_of\_acts" />  
 <TextField fx:id="Price\_input" layoutX="176.0" layoutY="411.0" promptText="Ticket\_price" />  
 <TextField fx:id="search\_input" layoutX="38.0" layoutY="14.0" prefHeight="25.0" prefWidth="241.0" promptText="Search" />  
 <ListView fx:id="view\_data" layoutX="37.0" layoutY="50.0" prefHeight="200.0" prefWidth="645.0" />  
 <Button fx:id="button\_save" layoutX="342.0" layoutY="439.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="119.0" text="Save changes" />  
 <ComboBox fx:id="box\_search" layoutX="288.0" layoutY="14.0" prefWidth="150.0" />  
 <TextField fx:id="Music\_input" layoutX="176.0" layoutY="439.0" promptText="Music" />  
 <Button fx:id="button\_add\_elem" layoutX="342.0" layoutY="352.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="119.0" text="Add new piece" />  
 <Button fx:id="button\_delete\_elem" layoutX="342.0" layoutY="381.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="119.0" text="Delete piece" />  
 <Button fx:id="button\_delete\_all\_elem" layoutX="342.0" layoutY="411.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="119.0" text="Clear all list" />  
 </children>  
</AnchorPane>

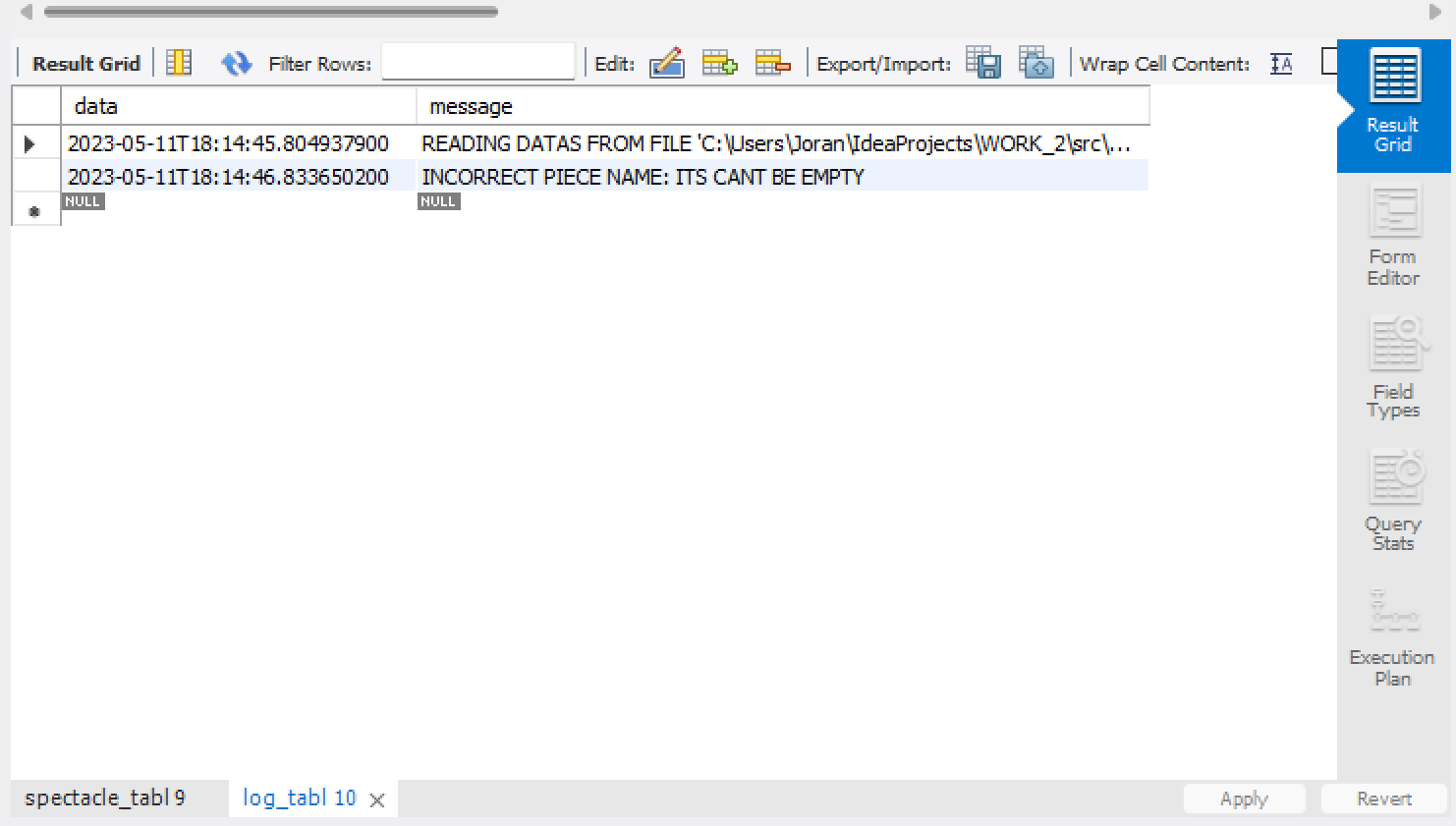
**Результати:**

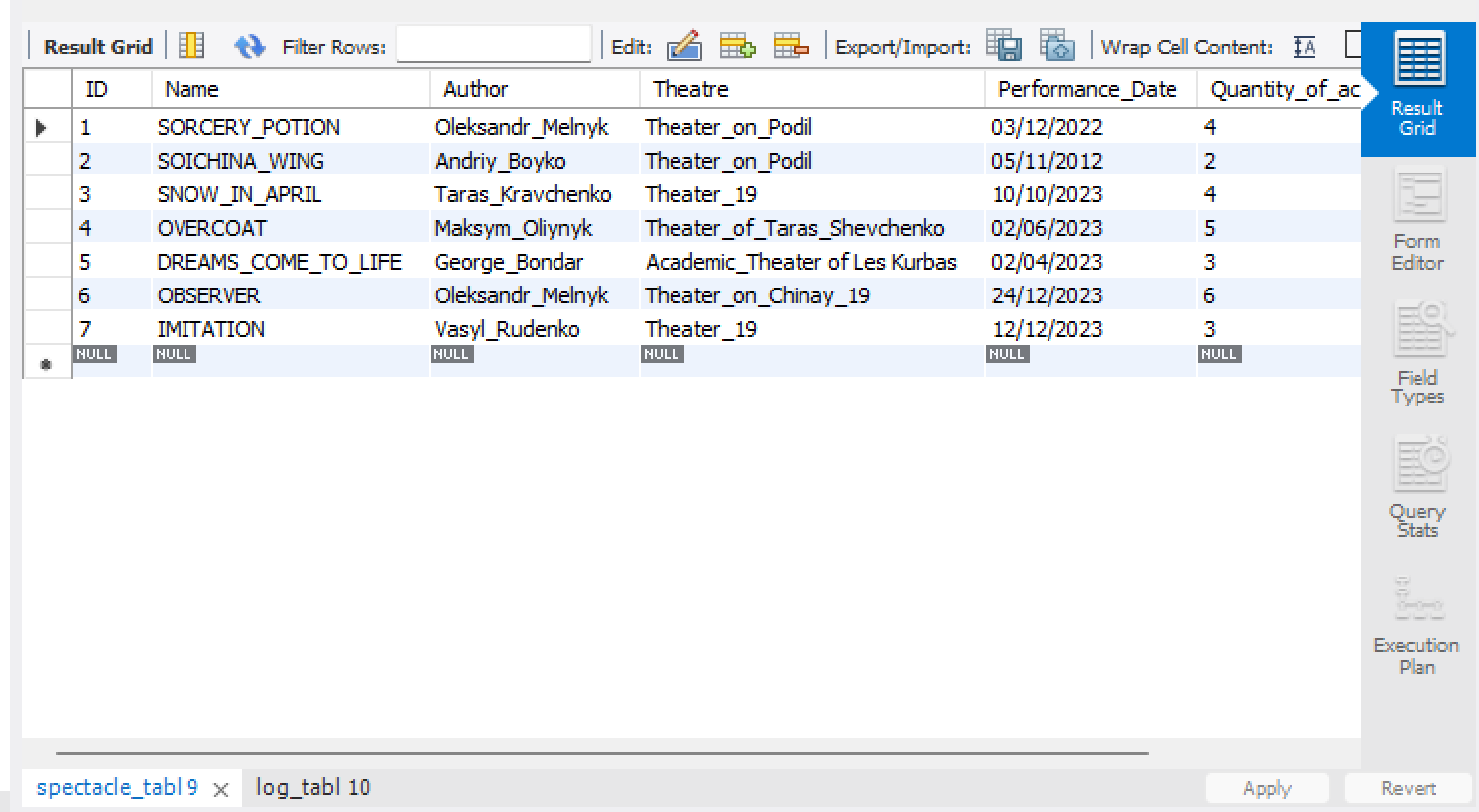
**Завантаження даних з файлу Spectacle\_data.cvs:**

* **Лістинг програми:**

****

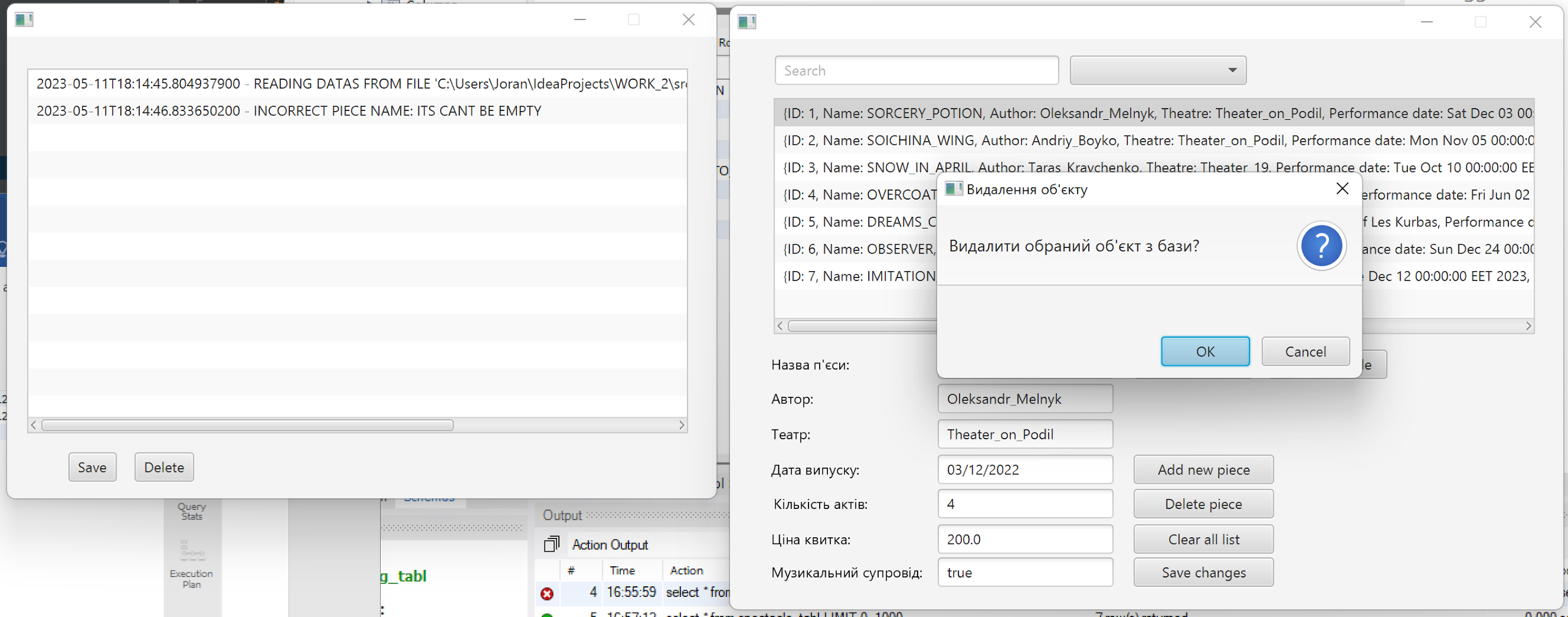
* **База даних:**

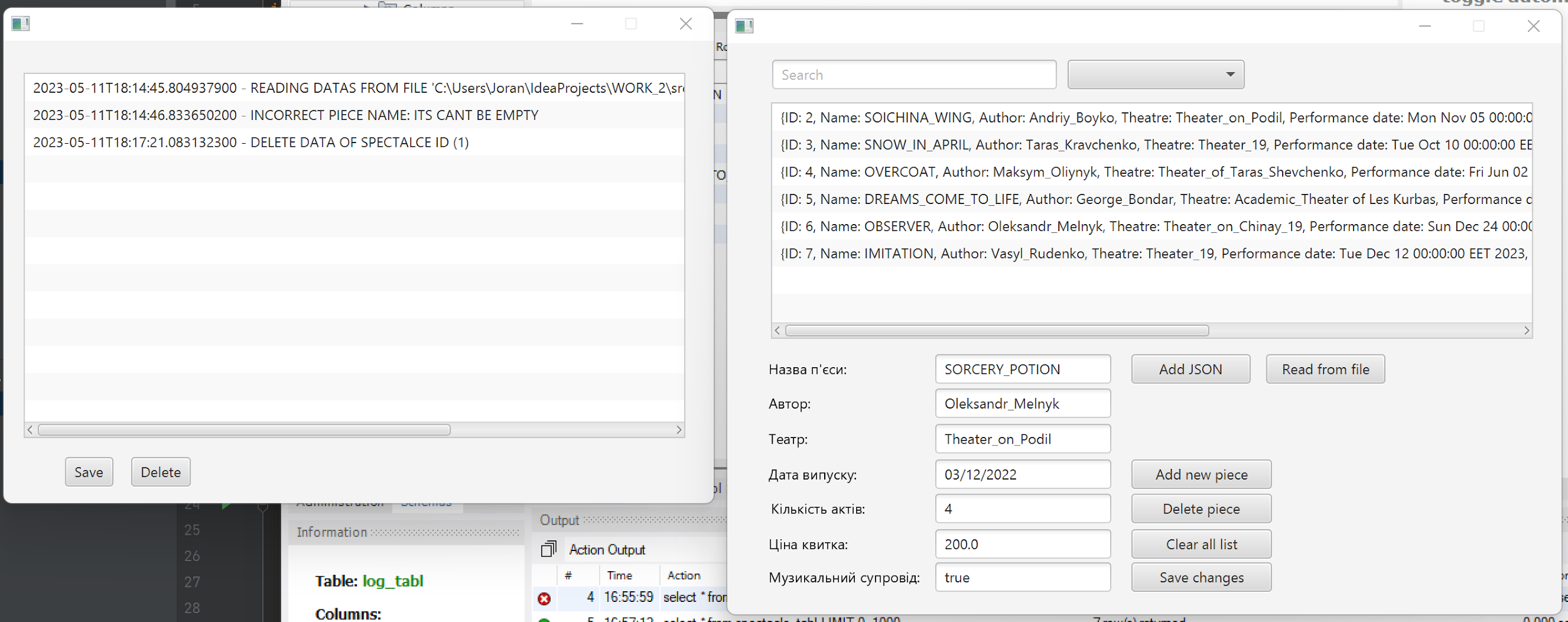
****

****

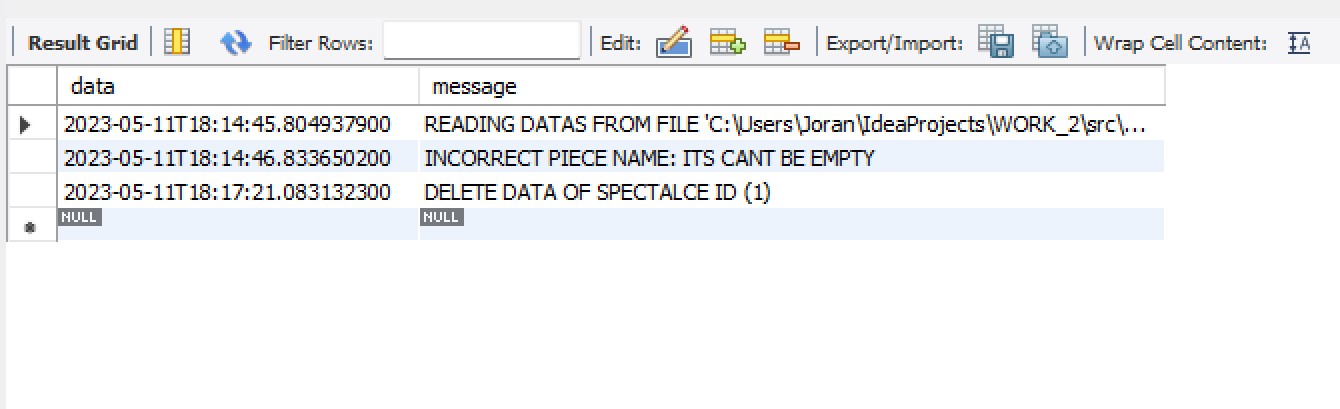
**Видалення даних вистави:**

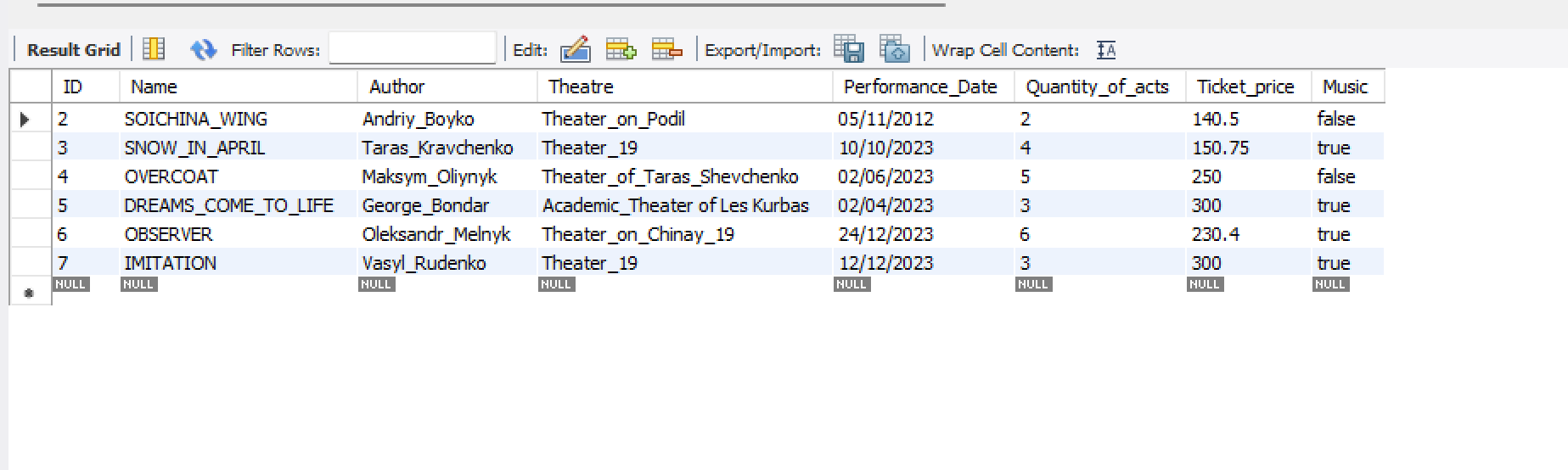
* **Лістинг програми:**

****

****

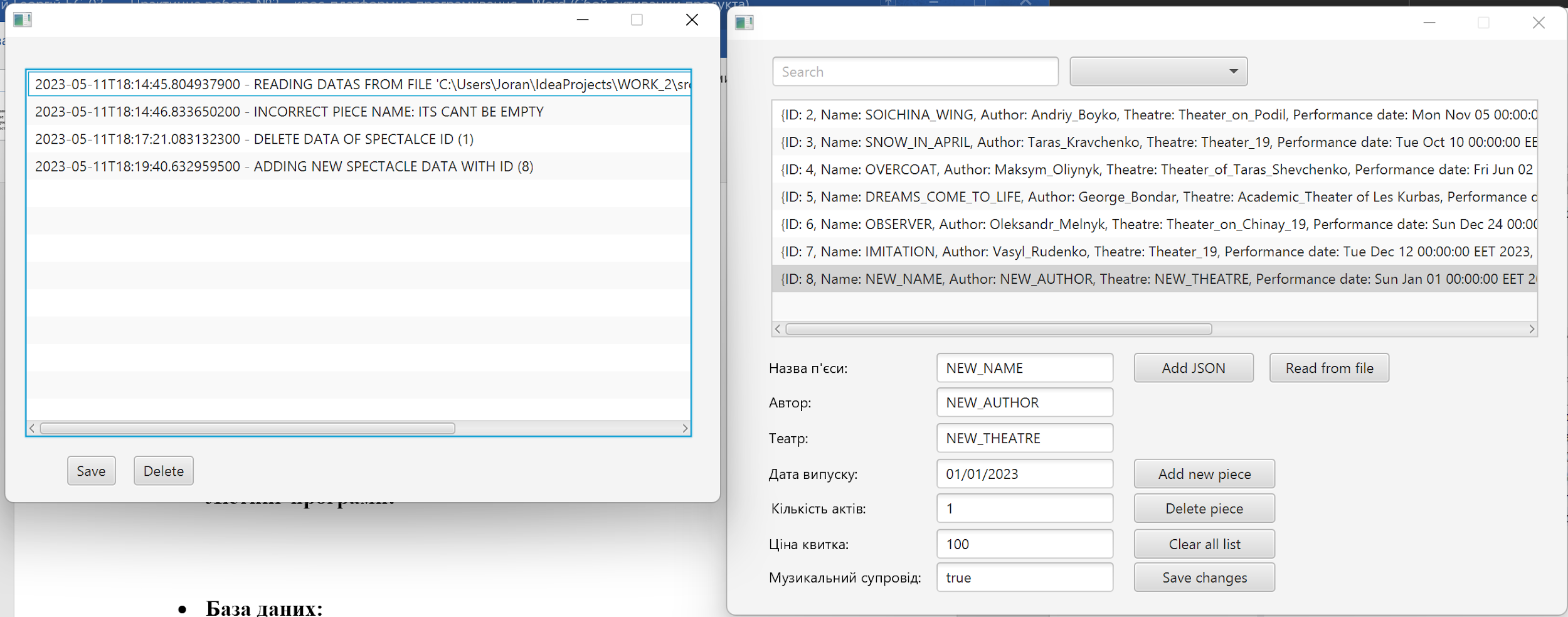
* **База даних:**

****

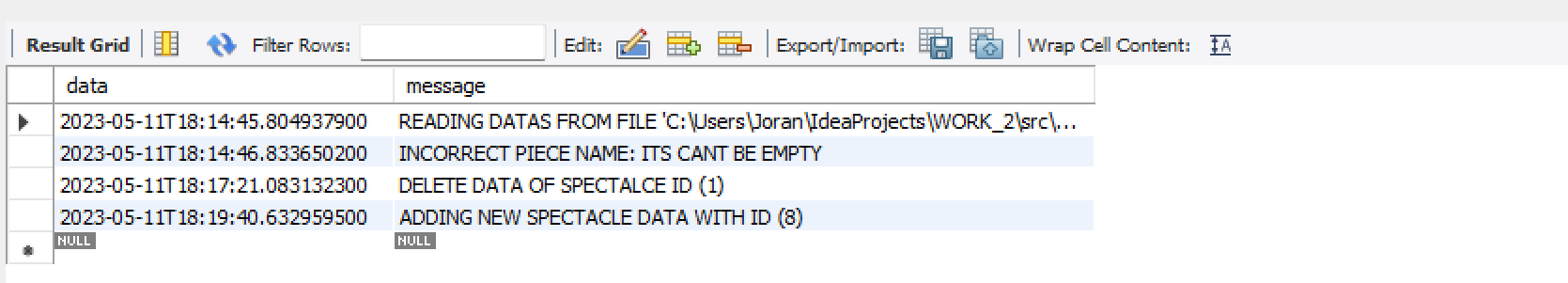
****

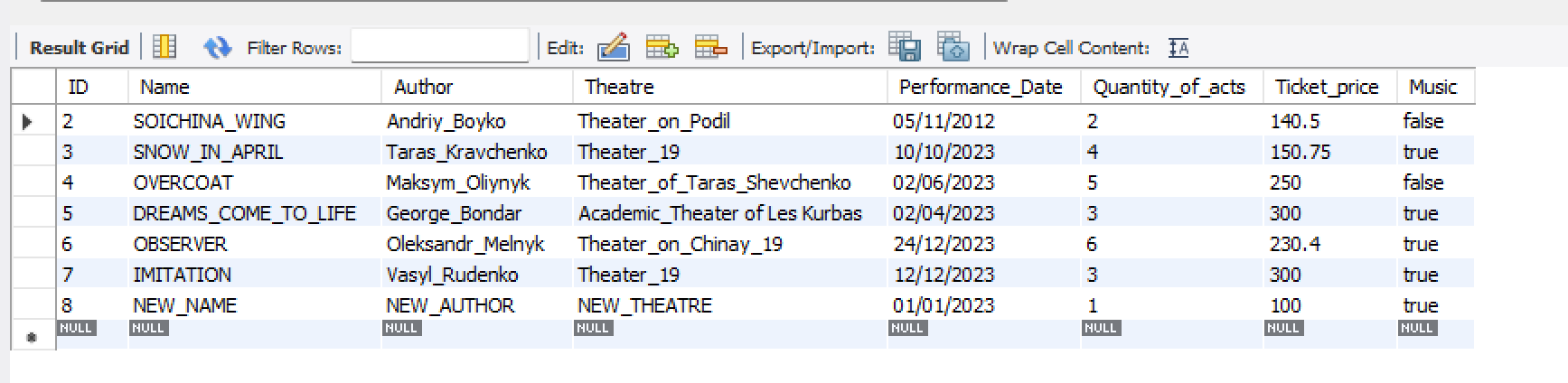
**Додавання даних вистави:**

* **Лістинг програми:**

****

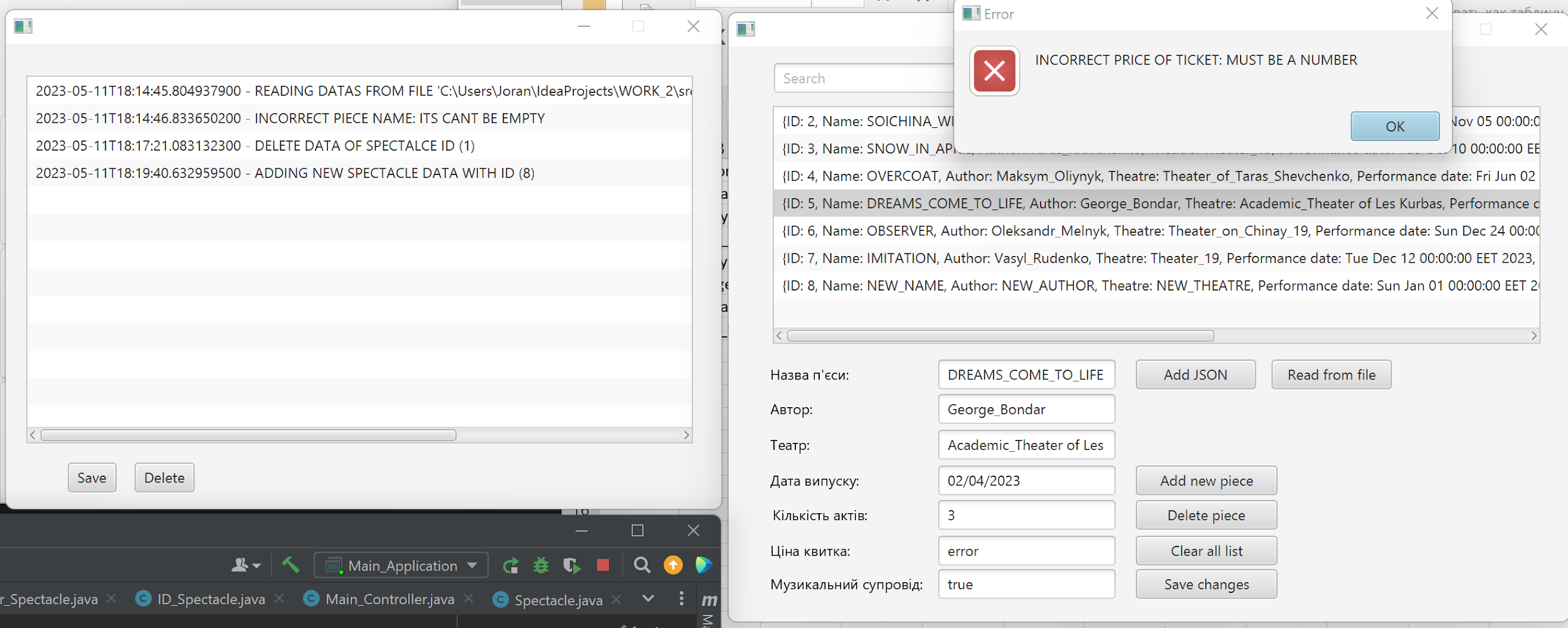
* **База даних:**

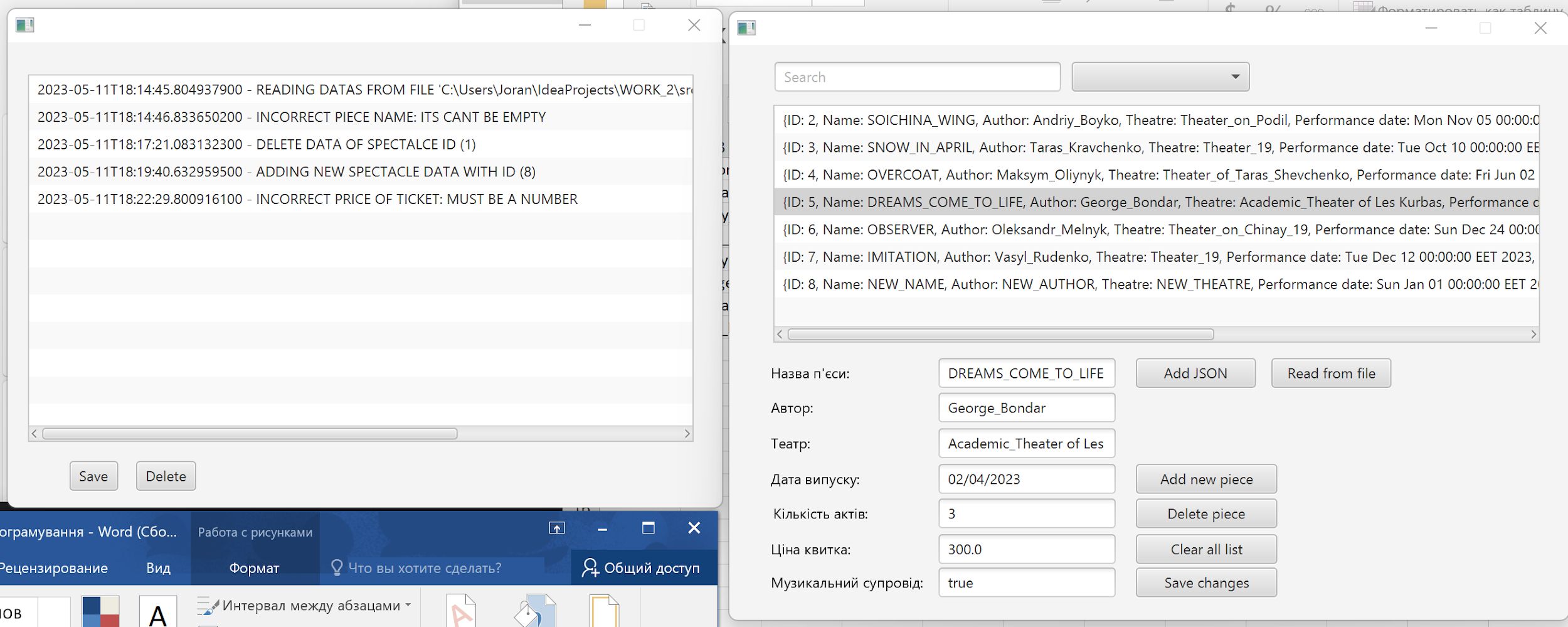
****

****

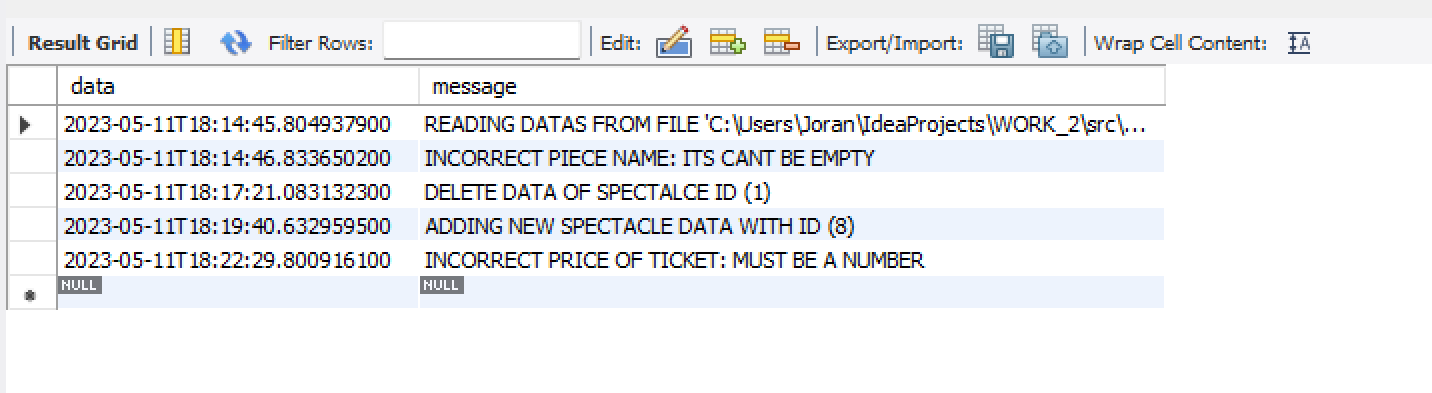
**Не вдала зміна даних вистави:**

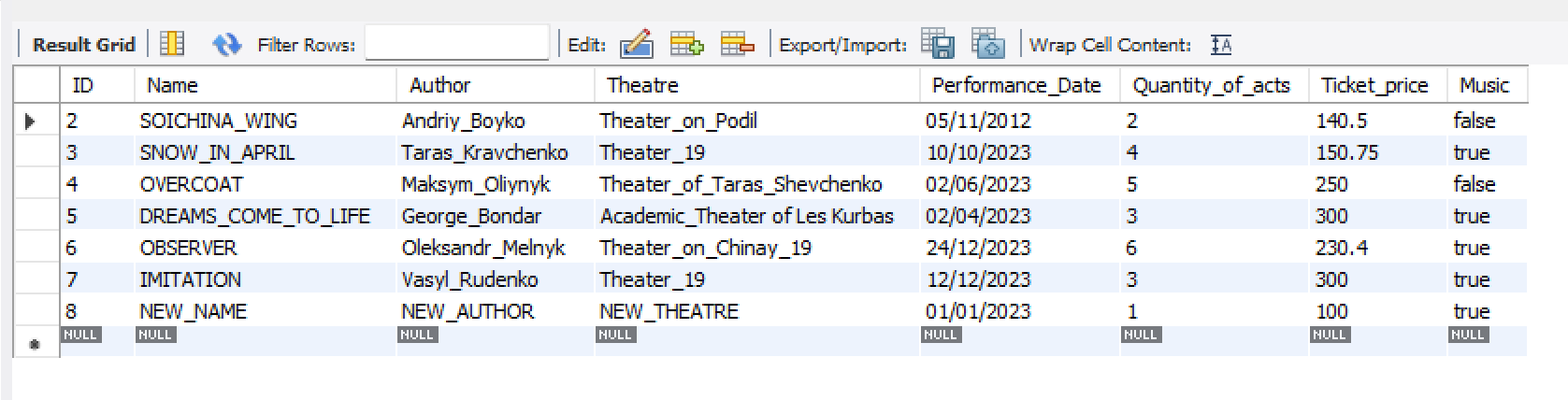
* **Лістинг програми:**

****

****

* **База даних:**

****

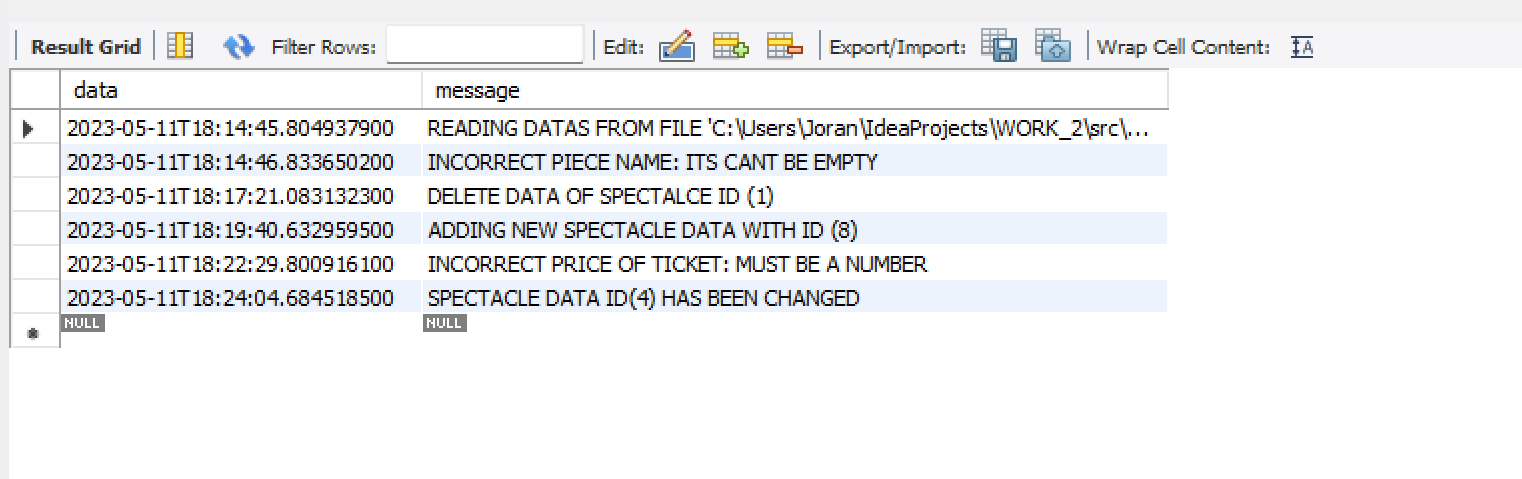
****

**Зміна даних вистави:**

* **Лістинг програми:**

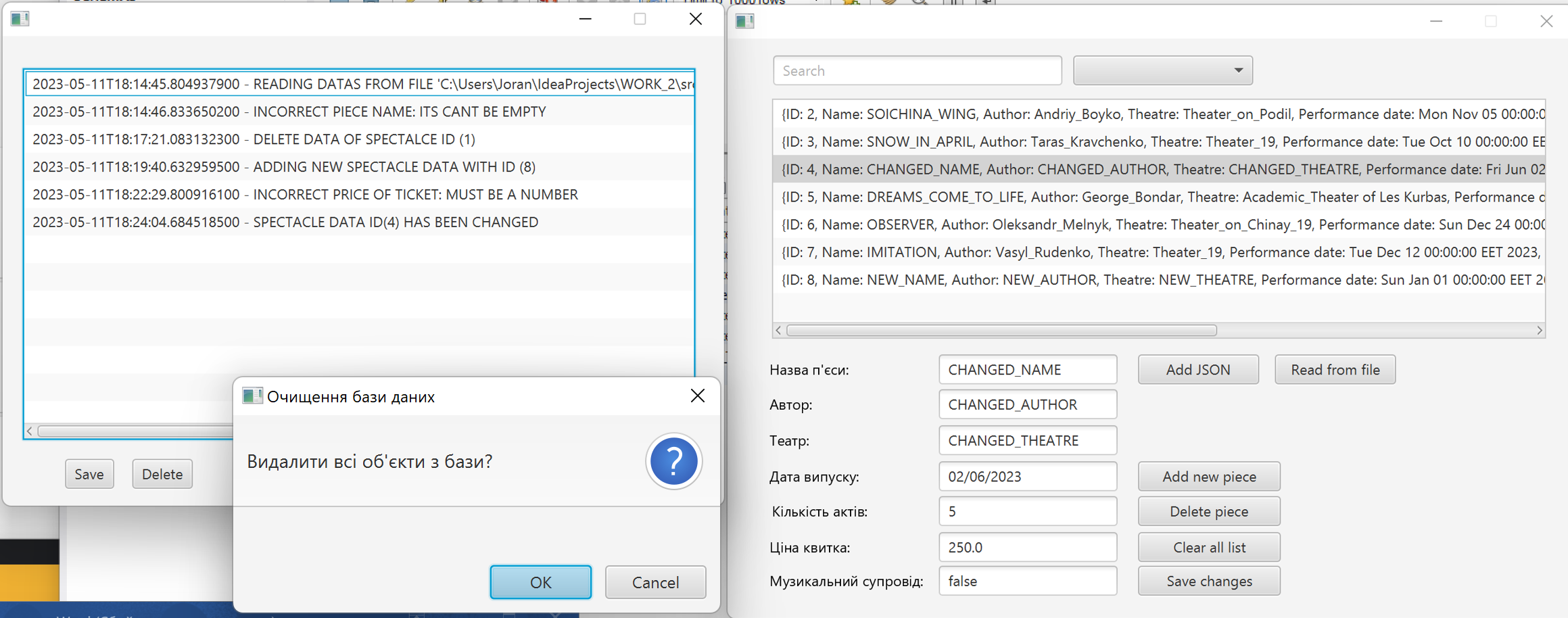
****

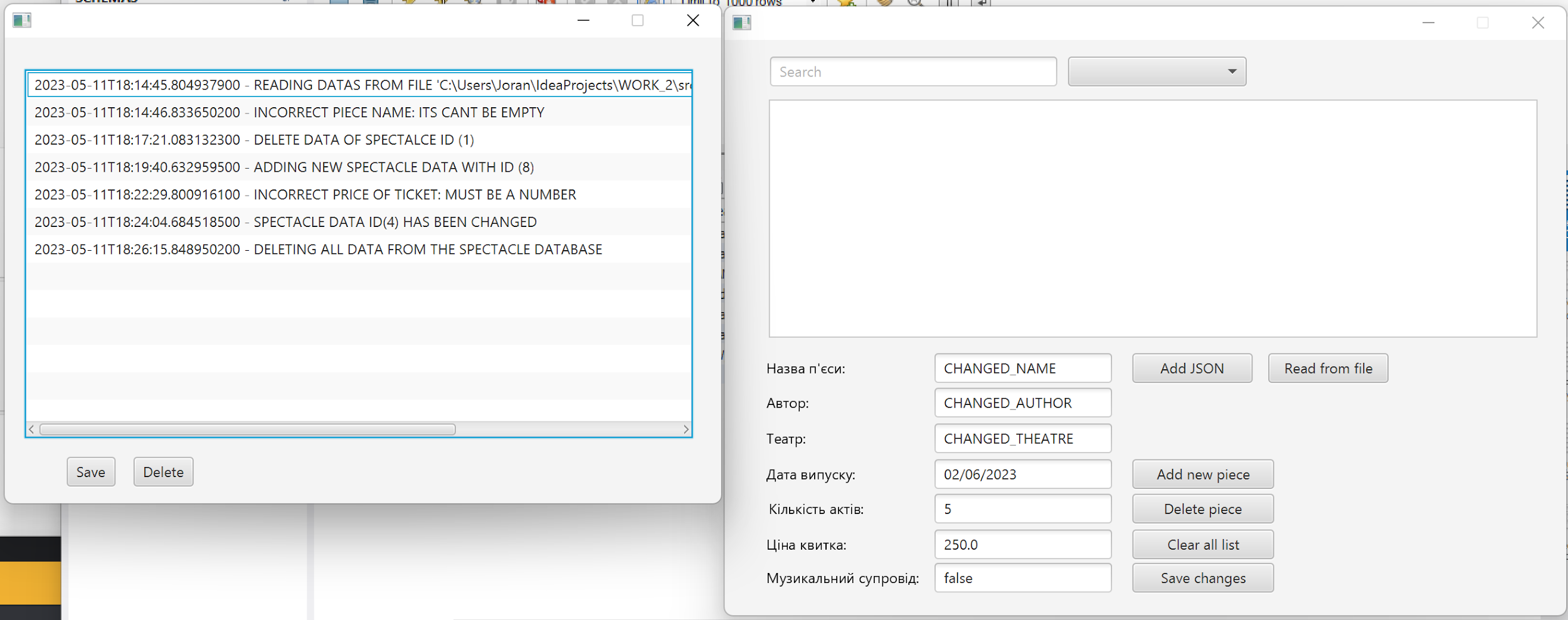
* **База даних:**

**** ****

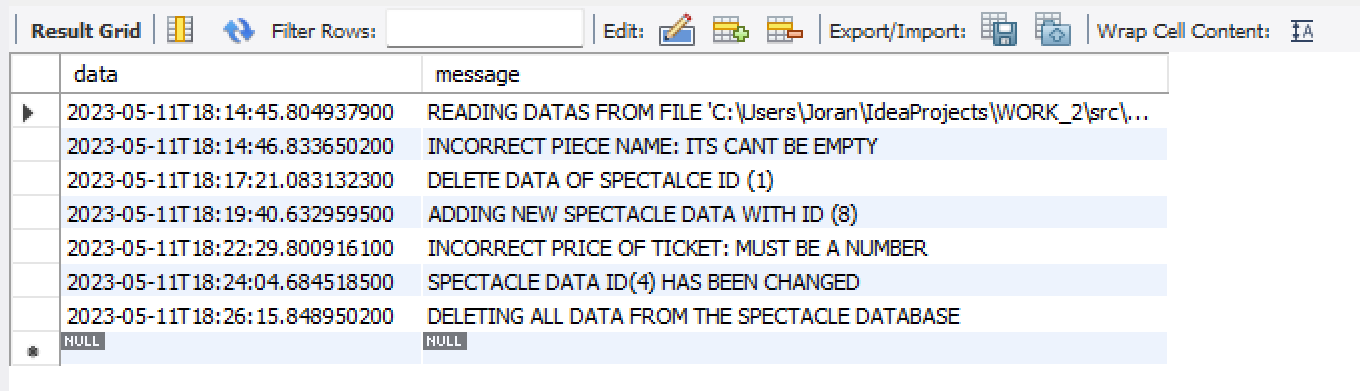
**Видалення всіх даних вистав:**

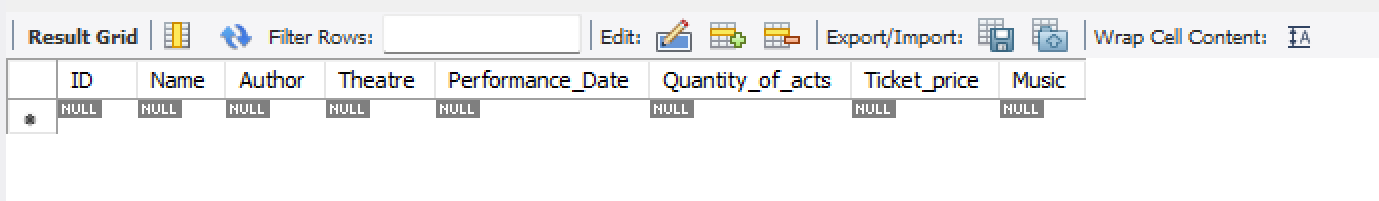
* **Лістинг програми:**

****

****

* **База даних:**

****

****

**Висновок:**

Ми ознайомилися з принципами роботи з базами даних із програмного коду; усвідомили технології підключення програми до бази даних на прикладі її конкретної реалізації; сформували вміння написання коду для виконання запитів до бази даних з використанням мови SQL; придбали навики використання параметризованих операторів; ознайомилися з можливостями пакетування операторів запитів SQL; організували зберігання об'єктів, що розроблені у попередніх роботах, у базі даних та отримування їх з неї за критеріями користувача.

## **Контрольні запитання:**

1. **Надайте характеристику компоненту для вбудовування веб сторінок в додаток JavaFX.**

Для вбудовування веб-сторінок в додаток JavaFX можна використовувати характеристику компоненту WebView. Це вбудований браузер, який дає можливість відображати веб-сторінки в JavaFX додатку.

Основні характеристики компонента WebView включають:

Відображення веб-сторінок: WebView дозволяє завантажувати та відображати веб-сторінки в JavaFX додатках. Він підтримує HTML5, CSS, JavaScript та інші веб-стандарти та має підтримку SSL і підтримує використання проксі-серверів.

1. **Надайте порівняльну характеристику аплетів, сервлетів та мідлетів Java.**

Аплети, сервлети та мідлети є різними технологіями для розробки Java- веб-додатків.

Аплети

* Аплети є Java-додатками, які можна вбудовувати в веб-сторінки.
* Вони виконуються в контексті веб-браузера і відображаються в спеціальному вікні або області сторінки.
* Вони можуть використовувати графічний інтерфейс користувача, виконувати обробку подій та комунікувати з сервер
* Аплети потребують встановлення Java Runtime Environment (JRE) на клієнтському комп'ютері.

Сервлети:

* Сервлети є Java-класами, які використовуються для розробки веб-додатків.
* Вони виконуються на сервері в спеціальному контейнері сервлетів (наприклад, Apache Tomcat) і взаємодіють з клієнтами через протокол HTTP.
* Сервлети мають повну функціональність Java та доступ до всіх ресурсів на сервері.
* Вони можуть обробляти запити клієнтів, генерувати відповіді у вигляді HTML-сторінок або інших форматів, зберігати та отримувати дані з бази даних, керувати станом сесії тощо.

Мідлети (Midlets):

* Мідлети є Java-додатками, які розробляються для мобільних пристроїв, зокрема для платформи Java ME (Java Micro Edition).
* Мають обмежену функціональність порівняно з повноцінними десктопними або веб-додатками.
* Мідлети не потребують встановлення Java Runtime Environment (JRE) на пристрої користувача.
* Мають доступ до обмеженого набору бібліотек та ресурсів на мобільному пристрої.

1. **Опишіть сутність JDBC та назвіть три основні види активності, які він надає.**

JDBC – платформо-незалежний стандарт взаємодії Java-додатків з різними СУБД. JDBC реалізований у вигляді пакета java.sql, що входить до складу Java SE

Основні три види активності:

1. Встановлення з'єднання з базою даних

JDBC надає можливість встановити з'єднання з реляційною базою даних. Цей процес включає в себе завантаження драйвера JDBC, вказівку URL-адреси бази даних, ім'я користувача та пароля.

1. Виконання SQL-запитів

JDBC дозволяє виконувати SQL-запити до бази даних. Це можуть бути SELECT-запити для вибірки даних, або UPDATE, INSERT, DELETE-запити для оновлення даних. За допомогою JDBC можна передавати SQL-запити до бази даних і отримувати результати запиту.

1. Отримання та обробка результатів з бази даних

JDBC надає можливість отримувати результати SQL-запитів та обробляти їх в Java-додатку, наприклад використовується об'єкт ResultSet та механізми роботи з метаданими.

1. **Яке місце в структурі JDBC посідає JDBC-драйвер та які функції він виконує?**

Драйвер JDBC - програмний компонент, що дозволяє Java взаємодіяти з базою даних. Для зв’язку з окремими базами даних JDBC вимагає драйвер для кожної бази даних. Драйвер JDBC видає підключення до бази даних і реалізує протокол передачі запиту та результату між клієнтом та базою даних.

1. **Перелічіть основні інтерфейси JDBC та опишіть їх.**

* Java.sql.DriverManager - забезпечує завантаження драйверів і створення нових з'єднань (connection) з базою даних; це основний інтерфейс JDBC, який визначає коректний вибір і ініціалізацію драйвера для даної СУБД в даних умовах;
* Java.sql.Connection - визначає характеристики і стан з'єднання з БД; крім того, він надає кошти для контролю транзакцій і рівня їх ізольованості;
* Java.sql.Statement - виконує функції контейнера по відношенню до SQL-виразу; при цьому під виразом розуміється не тільки сам текст запиту, а й такі характеристики, як параметри і стан вираження;
* Java.sql.ResultSet - надає доступ до набору рядків, отриманого в результаті виконання даного SQL-вирази.

1. **З яких компонентів складаються JDBC та JDBC Api?**

Компоненти JDBC:

* The JDBC API
* JDBC Driver Manager
* JDBC Test Suite
* JDBC-ODBC Bridge

Компоненти JDBC API

* Драйвер (Driver);
* З'єднання (Connection);
* Вираз (Statement);
* Менеджер драйверів (Driver Manager);
* Результат (ResultSet);
* Винятки (SQL Exception).

1. **Для чого використовуються підготовані оператори (prepared statements) SQL та як вони виконуються?**

Підготовлені оператори SQL використовуються для покращення продуктивності та забезпечення безпеки при виконанні SQL запитів до бази даних. Замість того, щоб виконувати SQL запит напряму, ми можемо використовувати підготовлений оператор SQL, який складається з SQL запиту, який містить місця підстановки (placeholder), та параметрів, які ми хочемо передати у ці місця підстановки.

Підготовлені оператори SQL мають наступні переваги:

* Захист від SQL ін'єкцій - параметри підготовлених операторів використовуються як значення, а не частини запиту, що дозволяє уникнути SQL ін'єкцій та інших безпекових проблем.
* Підвищення продуктивності - при використанні підготовлених операторів база даних може кешувати скомпільовані версії запитів та використовувати їх при подальших запитах, що дозволяє підвищити продуктивність виконання запитів.
* Повторне використання - використовуючи підготовлені оператори SQL, мають можливість використовуватися повторно з іншими параметрами. Через це зменшується час виконання та кількість запитів до бази даних.

1. **Надайте характеристику трьом основним типам ResultSet.**

У ResultSet є 3 типи:

* TYPE\_FORWARD\_ONLY: Це варіант за замовчуванням, коли курсор рухається від початку до кінця, тобто у напрямку вперед.
* TYPE\_SCROLL\_INSENSITIVE: У цьому типі курсор рухатиметься як вперед, так і назад. Якщо ми вносимо будь-які зміни в дані під час ітерації збережених даних, вони не оновлюватимуться в наборі даних, якщо хтось змінить дані в БД. Оскільки в наборі даних є дані з моменту, коли запит SQL повертає дані.
* TYPE\_SCROLL\_SENSITIVE: Це схоже на TYPE\_SCROLL\_INSENSITIVE, різниця полягає в тому, що якщо хтось оновлює дані після того, як SQL-запит повернув дані, в той час як ітерація відображатиме зміни в наборі даних.

1. **Перелічить методи для переміщення курсору по даних.**

Для переміщення курсору по даних в JDBC API можна використовувати наступні методи:

* next() - переміщає курсор на наступний рядок в результаті запиту до бази даних.
* previous() - переміщає курсор на попередній рядок в результаті запиту до бази даних. Цей метод може не підтримуватися деякими типами результатів запитів.
* first() - переміщає курсор на перший рядок в результаті запиту до бази даних.
* last() - переміщає курсор на останній рядок в результаті запиту до бази даних.
* absolute(int row) - переміщає курсор до вказаного рядка у результаті запиту до бази даних. При цьому, параметр row вказує номер рядка, куди потрібно перемістити курсор.
* relative(int rows) - переміщує курсор на певну кількість рядків у результаті запиту до бази даних. При цьому, параметр rows вказує на кількість рядків, які потрібно пропустити або пройти назад.
* beforeFirst() - переміщає курсор перед першим рядком у результаті запиту до бази даних.
* afterLast() - переміщає курсор за останнім рядком у результаті запиту до бази даних.
* isFirst() - перевіряє, чи знаходиться курсор на першому рядку у результаті запиту до бази даних.
* isLast() - перевіряє, чи знаходиться курсор на останньому рядку у результаті запиту до бази даних.

1. **Яку інформацію містить екземпляр SQLException?**

Екземпляр SQLException містить інформацію про виникнення помилки при виконанні запиту до бази даних. Ця інформація може включати в себе наступне:

Кожен SQLException надає кілька видів інформації:

Рядок, що описує помилку. Використовується як повідомлення виключення Java, доступне за допомогою методу getMessage;

Рядок «SQLstate». Значення рядка SQLState описані у відповідній специфікації. Метод getSQLStateType з DatabaseMetaData може використовуватися для визначення типу помилки та її подальшої обробки.