МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Практична робота №2**

з дисципліни «Розробка крос-платформного програмного забезпечення»

тема: Розділення поведінки, вигляду та управління; шаблон MVC. Графічний інтерфейс користувача

**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 розділенням структури задачи на поведінку, вигляд та управління; сформувати вміння відокремленного написання коду для цих компонентів та його повторного використання; ознайомитися з бібліотеками та платформами для створення графічного інтерфейсу користувача на мові Java; організувати перегляд об'єктів в колекції, що розроблені в попередній роботі, за допомогою GUI (Swing, JavaFX тощо).

**Завдання:**

1. Створити вікно програми з елементами управління для керування колекцією об'єктів, розробленою у попередній роботі.

2. Надати можливість перегляду колекції за допомогою списку об'єктів по Id та ключевій властивості та набору полів для інших властивостей вибраного в списку об'єкта, (або, як альтернатива, сітки (grid) для одночасного відображення об'єктів колекції).

3. Створити окремий фрейм для відображення виконаних дій та виявлених помилок; записи у фреймі мають починатися з дати та часу виконання дії або виникнення помилки. Помилки при виконанні дій мають бути відображені у фреймі повідомлень та у діалоговому вікні.

4. Надати можливість модифікації полів об'єкту; при зміні даних має відбуватися перевірка значень на припустимість за допомогою відповідних сетерів класу сутності.

5. Організувати послідовний пошук та фільтр об'єктів в списку по одній з ключових властивостей.

6. Використати стандартні діалогові вікна вибору файлів для:

* читання вхідних даних в форматі CSV/TSV
* збереження вихідних даних в форматі JSON
* збереження журналу виконаних дій та виявлених помилок.

## Варіанти:

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

## **Лістинг програми:**

**Container\_Spectacle.java:**

import java.util.Collections;  
import java.util.ArrayList;  
import java.util.Scanner;  
import java.io.IOException;  
import java.io.File;  
import java.io.FileWriter;  
  
public class Container\_Spectacle {  
 public ArrayList<Spectacle> Constain = new ArrayList<>();  
  
 public Container\_Spectacle(){}  
  
 public void Output(String name) throws Exception {  
 try{  
 File file = new File (name);  
 Scanner scan\_file = new Scanner(file);  
 String[] parametrs = scan\_file.nextLine().split(",");  
 int counter = 1;  
 while(scan\_file.hasNextLine())  
 {  
 counter+=1;  
 try{  
 String data = scan\_file.nextLine();  
 Spectacle spectacle = new ID\_Spectacle(data, parametrs);  
 Constain.add(spectacle);  
  
 }  
  
 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.work\_2;  
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, String[] value) throws Exception {  
 super(parametrs,value);  
 this.id = ++*id\_static*;  
 }  
 public int GET\_id() {  
 return id;  
 }  
  
  
 @Override  
 public String toString() {  
  
 return "{ID: " + GET\_id() +", "+ super.toString()+"}\n";  
 }  
  
 @Override  
 public String toJson(){  
 return ("\"ID\": " + id + ", " + super.toJson());  
 }  
  
  
  
  
  
}

**Spectacle.java:**

package com.example.work\_2;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import javafx.scene.control.Alert;  
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");  
 private static final String[] *spectacle\_field* = {"Name","Author","Theatre","Performance\_Date","Quantity\_of\_acts","Ticket\_price","Music"};  
 private int[] spectacle\_order = new int[*spectacle\_field*.length];  
  
  
 public Spectacle(){  
  
 }  
  
  
 public Spectacle(String parameters,String[] value) throws Exception  
 {  
 if(parameters == null)  
 throw new WorkException("EMPTY PARAMETERS");  
  
 String[] parameter = parameters.split(",");  
  
 if(value.length != 7 || parameter.length != 7)  
 {  
 throw new WorkException("INCORRECT QUANTITY OF DATA");  
  
 }  
  
 SET\_spectacle\_data(value);  
  
 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]]);  
  
  
 }  
  
  
 void SET\_spectacle\_data(String[] value) throws Exception{  
 for(int i = 0; i < *spectacle\_field*.length; ++i) {  
 for(int j = 0; j < spectacle\_order.length; ++j) {  
 if(*spectacle\_field*[i].equals(value[j])) {  
 spectacle\_order[i] = j;  
 }  
  
 }  
 }  
 }  
  
  
 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);  
  
}  
}

**WorkException.java:**

package com.example.work\_2;  
import java.io.FileWriter;  
import java.io.\*;  
import java.io.IOException;  
  
  
import java.time.LocalDateTime;  
  
public class WorkException extends Exception  
{  
  
 private static String *error\_fileName*= "Errors\_message.txt";;  
 private String info;  
  
 */\*\*  
 \*  
 \** ***@param*** *info contains error messages  
 \*  
 \*/* public WorkException(String info)  
 {  
 super(info);  
 *error\_fileName*= "Errors\_message.txt";  
 }  
  
 */\*\*  
 \* A method of writing an error message while the program is running  
 \** ***@param*** *error\_info contains the error message  
 \*/* public void FileError(String error\_info)  
 {  
 try  
 {  
  
 System.*out*.println(LocalDateTime.*now*() + ",TYPE OF ERROR: "+error\_info);  
 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());  
 }  
  
  
  
 }  
}

**WorkException:**

package com.example.work\_2;  
import java.io.FileWriter;  
import java.io.\*;  
import java.io.IOException;  
  
  
import java.time.LocalDateTime;  
  
public class WorkException extends Exception  
{  
  
 private static String *error\_fileName*= "Errors\_message.txt";;  
 private String info;  
  
 */\*\*  
 \*  
 \** ***@param*** *info contains error messages  
 \*  
 \*/* public WorkException(String info)  
 {  
 super(info);  
 *error\_fileName*= "Errors\_message.txt";  
 }  
  
 */\*\*  
 \* A method of writing an error message while the program is running  
 \** ***@param*** *error\_info contains the error message  
 \*/* public void FileError(String error\_info)  
 {  
 try  
 {  
  
 System.*out*.println(LocalDateTime.*now*() + ",TYPE OF ERROR: "+error\_info);  
 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.work\_2;  
  
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.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) {  
 *launch*();  
 }  
}

**Main\_Controller:**

package com.example.work\_2;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.io.File;  
import javafx.scene.Scene;  
import javafx.collections.FXCollections;  
import javafx.fxml.FXML;  
import java.util.stream.Collectors;  
import javafx.fxml.FXMLLoader;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
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;  
  
  
  
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 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 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 + "'");  
 System.*out*.println("READING DATAS FROM FILE '" + path + "'");  
 *container\_spect*.Output(path);  
  
 } catch (Exception ex) {  
  
 *error\_log*.addListView\_error(ex);  
 System.*out*.println(ex);  
 }  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
  
 }  
  
 private void Error\_log() throws IOException{  
 try {  
 FXMLLoader fxml = new FXMLLoader(getClass().getResource("view-error.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 + "'");  
 System.*out*.println("SAVE DATA IN JSON FILE '"+ path + "'");  
 }  
 catch(Exception ex)  
 {  
 *error\_log*.addListView\_error(ex);  
 System.*out*.println(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());  
  
 }  
 private void Spect\_select()  
 {  
  
 *spect* = view\_data.getSelectionModel().getSelectedItem();  
 Fill\_spect();  
  
 }  
 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("CHANGE ITEM FROM {"+s+ "}"+" TO {"+*spect* + "}");  
 System.*out*.println("CHANGE ITEM FROM {"+s+ "}"+" TO {"+*spect* + "}");  
 }  
 catch (Exception ex)  
 {  
 *error\_log*.addListView\_error(ex);  
 System.*out*.println(ex);  
 }  
 view\_data.setItems(FXCollections.*observableList*(*container\_spect*.GET\_arraylist()));  
 }  
 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'");  
 System.*out*.println("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'");  
 System.*out*.println("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'");  
 System.*out*.println("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'");  
 System.*out*.println("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'");  
 System.*out*.println("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'");  
 System.*out*.println("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'");  
 System.*out*.println("SEARCH FOR '"+ search\_input.getText() + "' IN CASE 'Quantity\_of\_acts'");  
 break;  
 }  
 view\_data.setItems(FXCollections.*observableList*(arr));  
 }  
 @FXML  
 void initialize() throws IOException {  
 Error\_log();  
 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());  
 }  
  
}

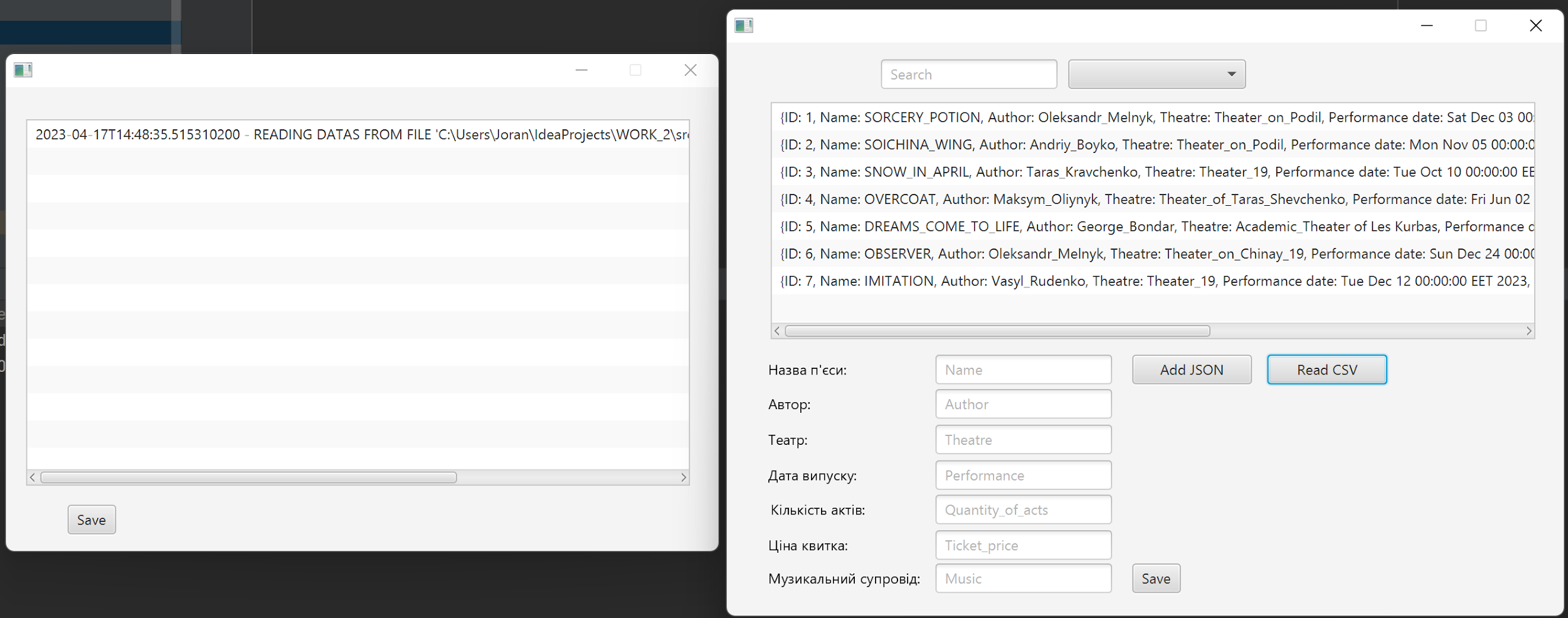
**view-error.fxml:**

<?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.work\_2.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" />  
 </children>  
</AnchorPane>

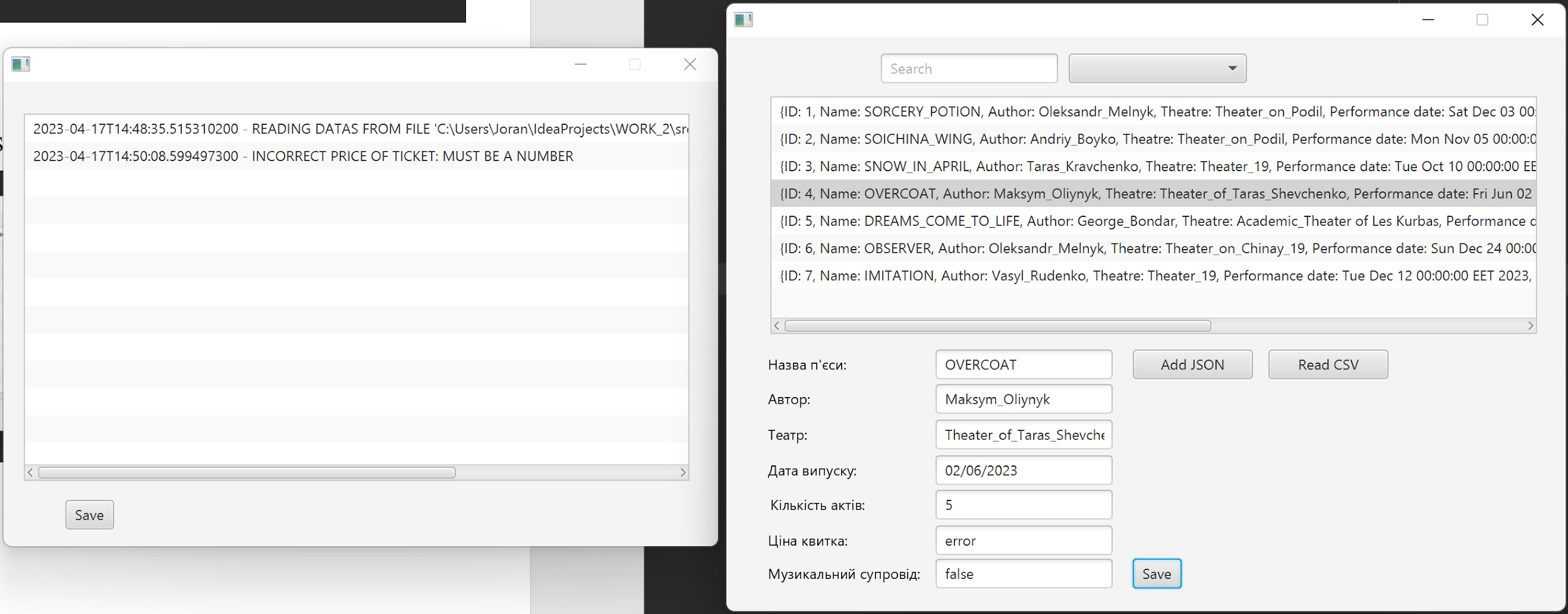
**view-main.fxml:**

<?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="478.0" prefWidth="708.0" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.work\_2.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 CSV" />  
 <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="Music\_input" layoutX="176.0" layoutY="439.0" promptText="Music" />  
 <TextField fx:id="search\_input" layoutX="130.0" layoutY="14.0" 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" text="Save" />  
 <ComboBox fx:id="box\_search" layoutX="288.0" layoutY="14.0" prefWidth="150.0" />  
 </children>  
</AnchorPane>

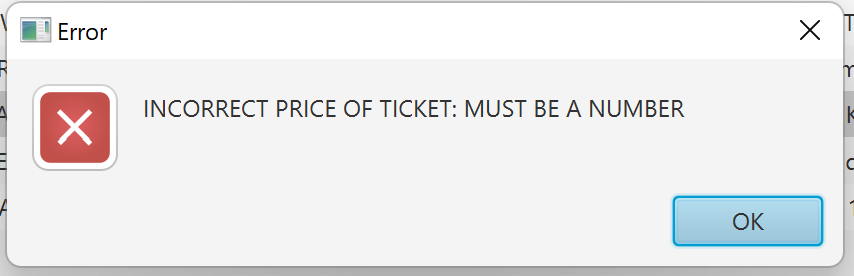
**Результати:**Зчитування даних з файлу СSV:



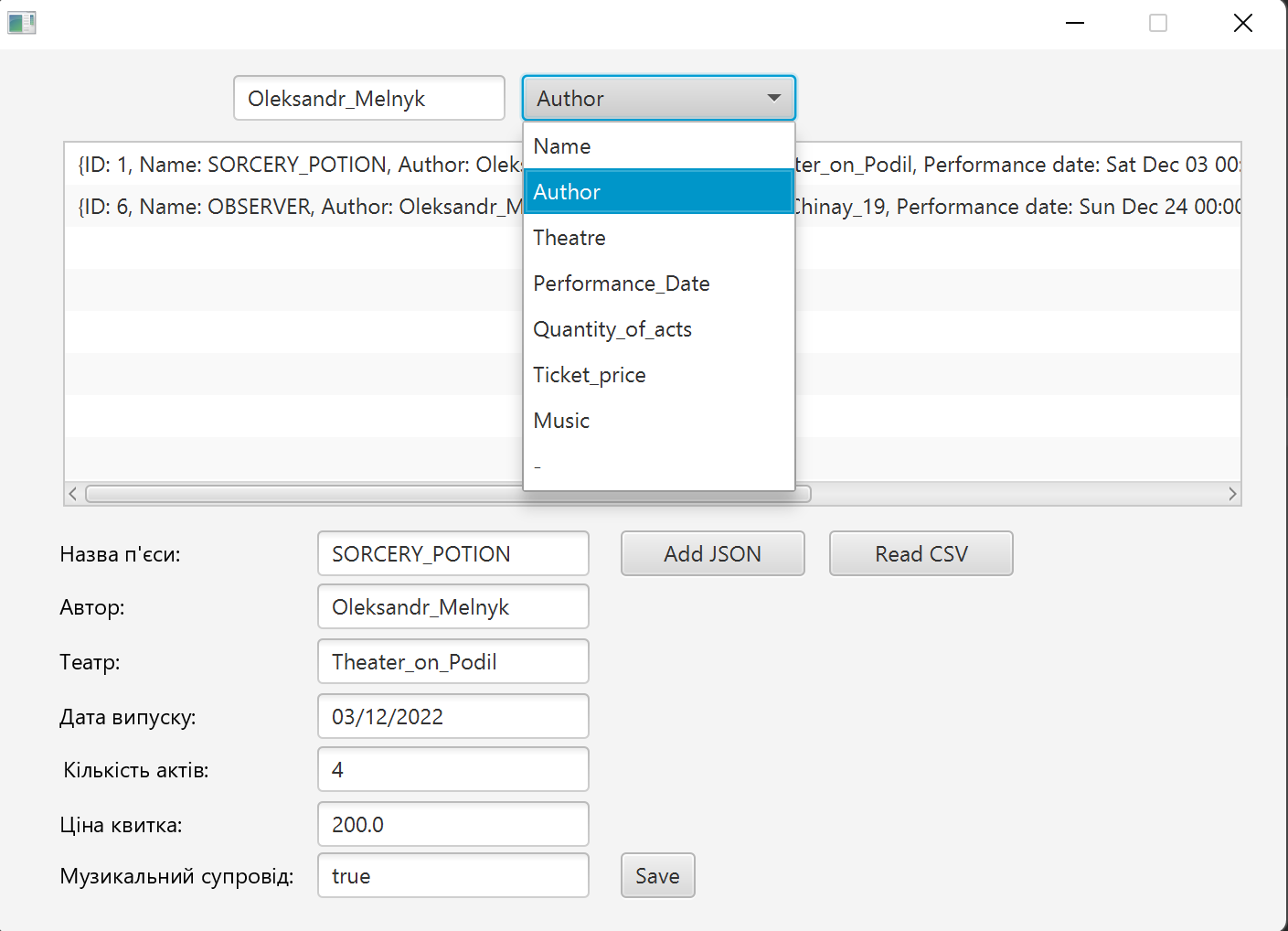
Некоректна зміна даних:

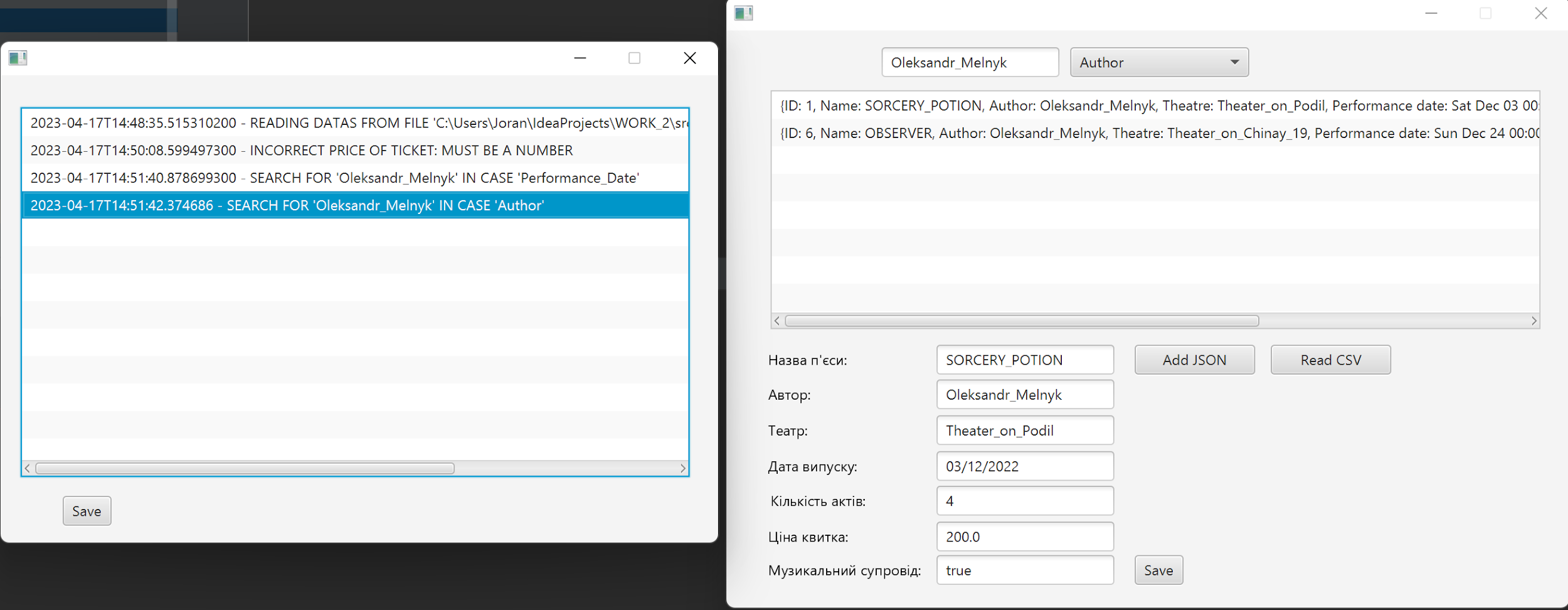


Попередження про некоректну зміну даних:

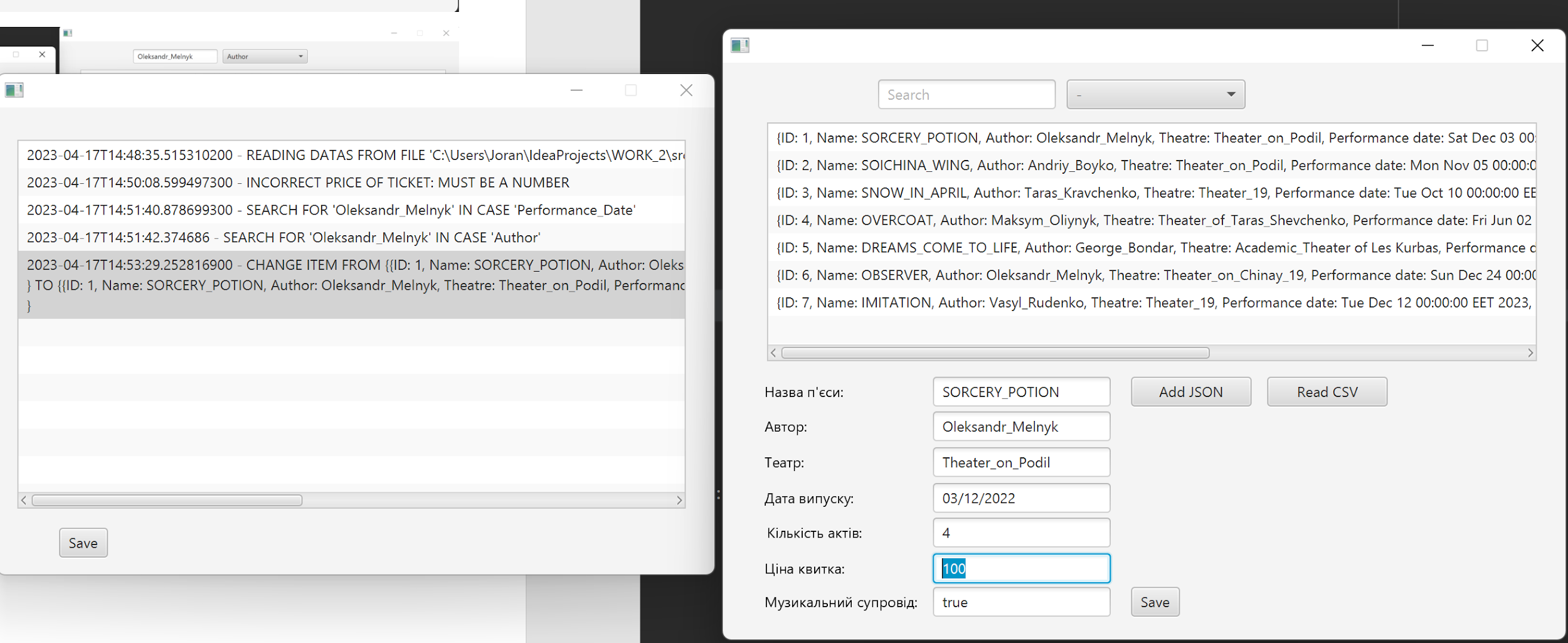


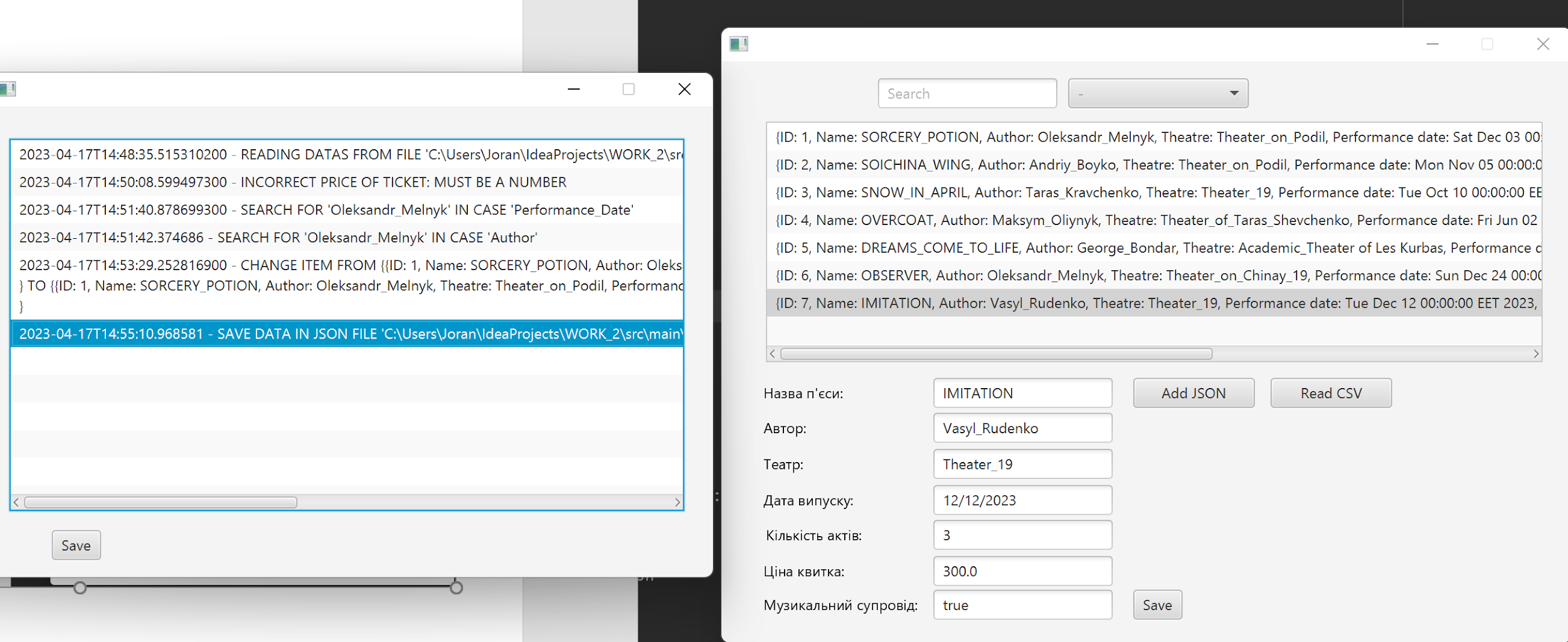
Пошук схожих об’єктів в базі через параметр:



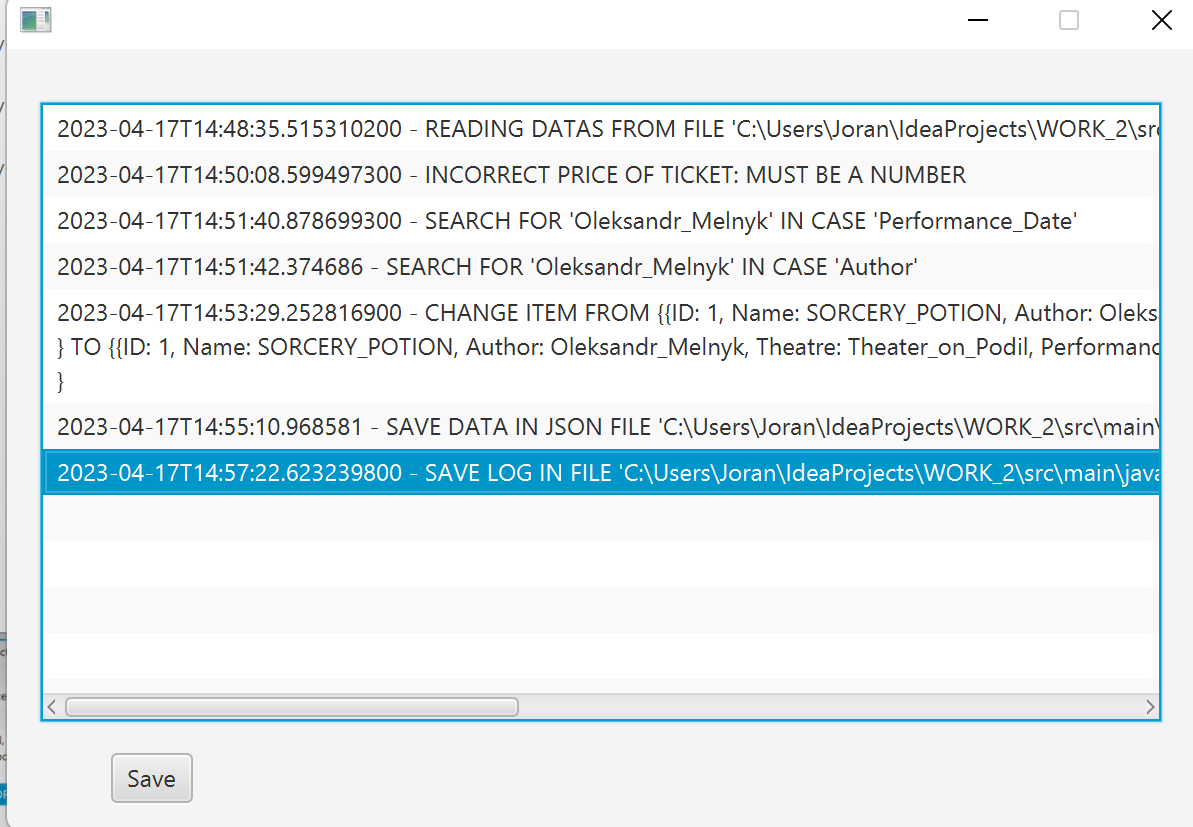


Зміна параметру:

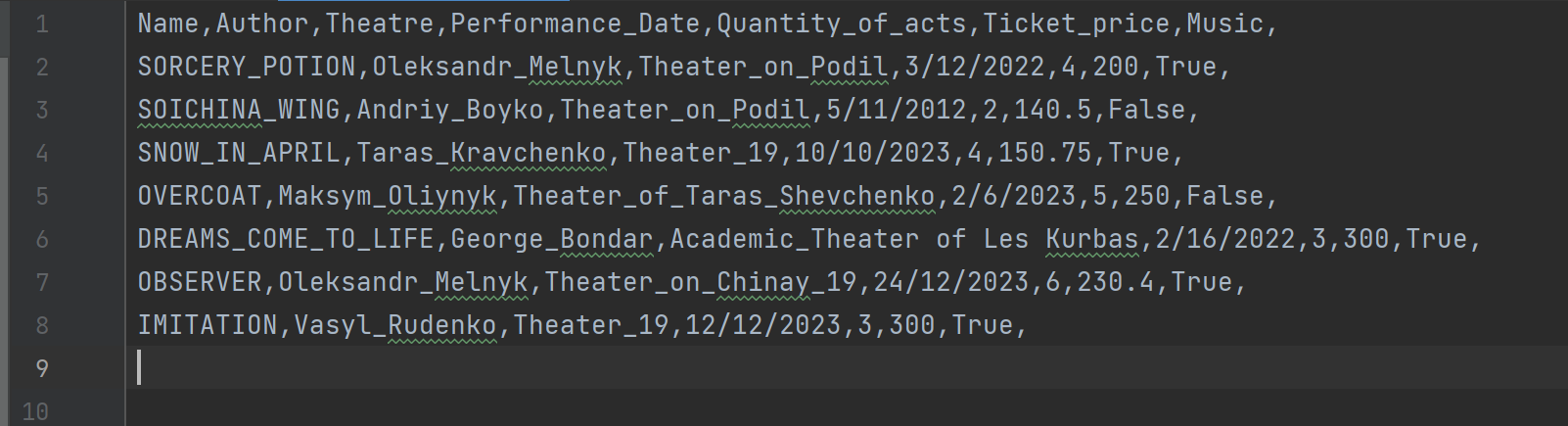
  
Збереження файлу в JSON форматі:



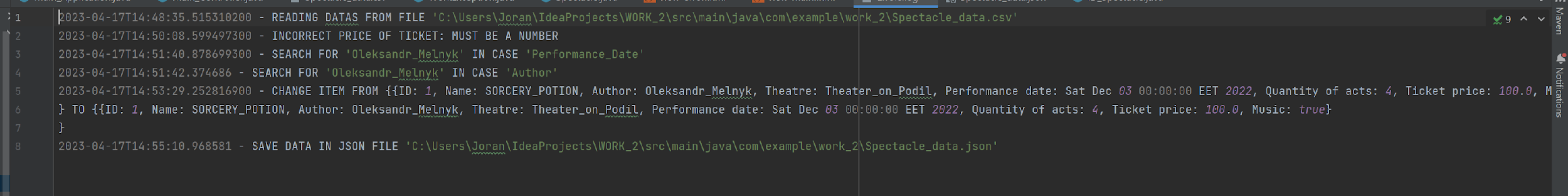
Збереження журналу виконаних дій та виявлених помилок

****

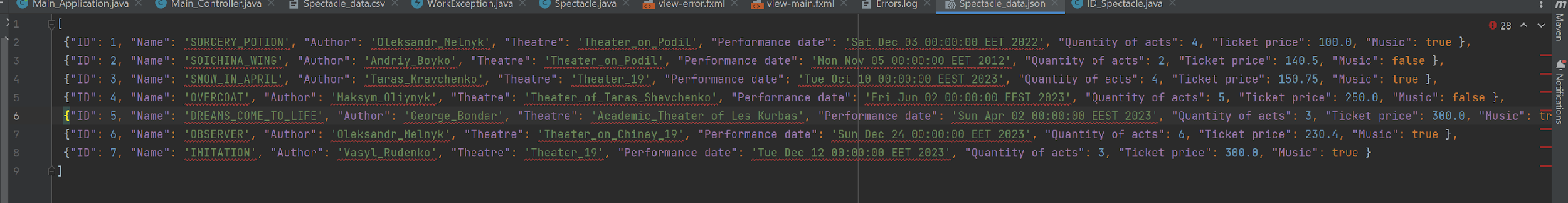
Spectacle\_data.csv:



Errors.log:



Spectacle\_data.json:

****

**Висновок:** Ми ознайомилися з розділенням структури задачи на поведінку, вигляд та управління; сформували вміння відокремленного написання коду для цих компонентів та його повторного використання; ознайомитися з бібліотеками та платформами для створення графічного інтерфейсу користувача на мові Java; організували перегляд об'єктів в колекції, що розроблені в попередній роботі, за допомогою GUI (Swing, JavaFX тощо).

## **Контрольні запитання:**

1. **Опишіть, в чому користь шаблону MVC та розкрийте його сутність. Вкажіть, які частини та які класи в протоколі цієї роботи відносяться до кожного з компонентів шаблону.**

Модель–вигляд–контролер (MVC) – це паттерн проектування, який передбачає виділення компонентів програми, що відповідають за вирішення певних окремих задач та розділяє програму на модель, представлення та контролер.

* Модель: Container\_Spectacle, ID\_Spectacle, Spectacle, WorkException
* Представелення: view-main.fxml, view-error.fxml:
* Контролер: Main\_Controller, Error\_Controller

1. **Назвіть відомі технології Java для розробки графічного користувацького інтерфейсу. Обґрунтуйте вибір бібліотеки для реалізації інтерфейсу при виконанні цієї роботи.**

Для графічного інтерфейсу в java можна використовувати - JavaFX, SwingX, Apache Pivot, SWT.

JavaFX має велику кількість готових елементів для розробки інтерфейсу користувача, що дозволяє розробникам створювати складні додатки зі зручним та інтуїтивним інтерфейсом користувача. Це дозволяє розробникам зосередитись на основній функціональності додатка, а не на розробці власного інтерфейсу.

JavaFX також має вбудовану підтримку графіки та мультимедіа, що дозволяє розробникам легко і швидко створювати багатофункціональні додатки з різноманітними ефектами та анімацією.

JavaFX пропонує простий та інтуїтивний інтерфейс для розробки додатків, що дозволяє розробникам швидко створювати складні додатки зі зручним інтерфейсом користувача.

1. **Поясніть, для чого використовуються та як працюють слухачі подій (event listeners).**

Слухачі подій використовуються для обробки подій. Коли деякі дії «слухаються» і після виконання, програма реагує прописаним кодом в слухачеві цієї події. Слухач генерує якусь подію і коли вона відбувається, то викликається код.

1. **За рахунок яких технологій можна відокремити налаштування зовнішнього вигляду додатку JavaFX від його бізнес-логіки?**

Завдяки використанню окремого файлу з розміткою можна відокремити налаштування зовнішнього вигляду додатку JavaFX від його бізнес-логіки, який має розширення .fxml

1. **Перелічіть поширені середовища розробки програм на Java та порівняйте їх можливості.**

* IntelliJ IDEA – поширюється в двох версіях, одна з яких абсолютно безкоштовна. Випускається компанією JetBrains та має регулярні оновлення та підтримку нових технологій
* NetBeans – найпотужніша середовище розробки з відкритим вихідним кодом, орієнтована на веб, мобільні і десктопні додатки. Працює з Linux, Windows, MacOS та Oracle Solaris.
* Eclipse – Має багато версій, актуальна з яких на даний момент називається Neon. У ній було реалізовано багато нових функцій, наприклад, відладчик приймає додаткові аргументи налагодження, а також перегляд ресурсів операційної системи, який дозволяє використовувати в налагодженні інформацію про процеси.

1. **Надайте порівняльну характеристику узагальнення в Java та шаблону в C++.**

Узагальнення (Generics) в Java та шаблони (Templates) в C++ - це концептуально схожі механізми, які дозволяють писати загальні алгоритми та контейнери, які можуть працювати з різними типами даних. Однак, є кілька відмінностей між ними:

* + Синтаксис: Синтаксис узагальнень в Java та шаблонів в C++ різний. В Java, узагальнення оголошується за допомогою символу "<>" після імені класу, наприклад ArrayList<String>. У C++, шаблони оголошуються за допомогою ключового слова "template", наприклад vector<int>.
  + Перевірка типів: У Java, перевірка типів відбувається під час компіляції, що дозволяє виявляти помилки на ранніх етапах розробки. У C++, перевірка типів здійснюється під час компіляції, але також може бути відкладена до моменту виконання програми, що може призвести до помилок під час виконання.
  + Обмеження типів: У Java можна використовувати обмеження типів, щоб обмежити типи даних, які можуть використовуватися в узагальненні. Наприклад, ArrayList<T extends Number> дозволяє використовувати узагальнення тільки з типами, які наслідують клас Number. У C++, шаблони також можуть мати обмеження типів за допомогою ключового слова "typename", наприклад template<typename T> void printVector(vector<T> v).

1. **Опишіть шаблони проектування “Адаптер” та “Фабричний метод”.**

Фабричний метод - породжуючий патерн (шаблон) проектування, що надає підклас інтерфейс для створення екземплярів деякого класу. У момент створення спадкоємці можуть визначити, який клас створювати. Іншими словами, Фабрика делегує створення об'єктів спадкоємцям батьківського класу. Це дозволяє використовувати в коді програми не специфічні класи, а маніпулювати абстрактними об'єктами на найвищому рівні. Також відомий під назвою віртуальний конструктор

Адаптер – шаблон проектування, який дозволяє об’єктам з несумісними інтерфейсами працювати разом. Адаптер представляє собою деякий об’єкт-перекладач, який трансформує інтерфейс або дані одного об’єкту у зрозумілий вигляд для іншого об’єкту.

1. **Для чого та де саме використовується IDL?**

IDL - це мова специфікації, яка використовується для опису інтерфейсу прикладного програмування програмного компонента. За допомогою IDL описують інтерфейс незалежно від мови, забезпечуючи зв'язок між програмними компонентами, які не мають спільної мови, наприклад, між компонентами написаними на C ++ та написаними на Java. IDL зазвичай використовується в програмному забезпеченні віддаленого виклику процедур. У цих випадках машини на різних кінцях можуть використовувати різні операційні системи та комп’ютерні мови. IDL пропонує «міст» між двома різними системами.

1. **Опишіть призначення блоку finally в Java.**

finally створює блок коду, який буде завершено після завершення блоку try/catch, але перед кодом, запропонованим за ним. Блокування буде виконане, незалежно від того, передано чи ні. Оператор finally не обов'язковий, проте кожен оператор try вимагає наявності або catch, або finally.

1. **Що таке “виключення, що перевіряється” в Java та в чому його відмінність від інших типів виключень?**

виключення, що перевіряється – використовується під час компіляції коду, бо компілятор вказує найчастіші з них та попереджає про них перед виконанням коду