**Лабораторна робота №5**

**Введення-виведення(I/O). Файли. Виключення**

**Тема**: Розробка та реалізація програм із використанням класів для зберігання даних

та файлового введення-виведення

**Мета роботи**: отримати навички створення та реалізації програм, що реалізують

операції введення-виведення із файлами.

**Завдання**

1. На основі завдання з лабораторної роботи №3 (частина1), створити абстрактний

клас, в якому визначити абстрактні поля, що характеризують об’єкт, наприклад,

одиниці вимірювання, кількість в упаковці та інше.

2. Визначити метод, що створює порожній масив об’єктів, з максимальною

кількістю 100 елементів.

2. Реалізувати зберігання даних у текстовий файл та зчитування з текстового

файлу. Враховувати можливість виникнення виключень, та обробляти їх, виводячи

відповідні повідомлення для користувача.

3. Створити інтерактивне меню, за допомогою якого надати можливість

користувачеві виконувати додавання нових та вилучення існуючих даних, файлові

операції введення-виведення та запити відповідно варіанту завдання.

**Короткі теоретичні відомості**

Сохранение объектов в текстовом формате

Рассмотрим пример программы, сохраняющей массив записей типа Employee в

текстовом файле. Каждая запись сохраняется в отдельной строке. Поля экземпляра

отделяются друг от друга разделителем. В качестве разделителя в данном примере

используется знак вертикальной черты (|). Другим распространенным разделителем

является знак двоеточия (:).

Мы пока что не будем касаться того, что может произойти, если символ |

встретится непосредственно в одной из сохраняемых символьных строк. Ниже

приведен образец сохраняемых записей:

Harry Hacker|35500|1989|10|1

Carl Cracker|75000|1987|12|15

Tony Tester|38000|1990|3|15

Процесс записи происходит очень просто. Для этой цели применяется класс

PrintWriter, поскольку запись выполняется в текстовый файл. Все поля просто

записываются в файл, завершаясь символом |, а если это последнее поле, то

комбинацией символов \n. Весь процесс записи совершается в теле приведенного

ниже метода writeData(), который описан в классе Employee.

public void writeData(PrintWriter out) throws IOException

{

GregorianCalendar calendar = new GregorianCalendar();

calendar.setTime(hireDay);

out.println(name + "|"

+ salary + "|"

+ calendar.get(Calendar.YEAR) + "|"

+ (calendar.get(Calendar.MONTH) + 1) + "|"

+ calendar.get(Calendar.DAY\_OF\_MONTH));

}

Файлы

Приведенный выше пример целесообразно применять к текстовыми файлами умеренной длины. Если же файл крупный или двоичный, то для обращения с ним можно воспользоваться потоками ввода-вывода или чтения и записи

данных, как показано ниже. Их методы избавляют от необходимости обращаться непосредственно к классам FilelnputStream, FileOutputStream, BufferedReader или

BufferedWriter.

*Розв’язання*

1. **Текст програми**

// = = = ELib.java = = =

public class ELib extends Application {

private IndexWindow indexWindow;

@Override

public void start(Stage primaryStage) throws IOException {

indexWindow = new IndexWindow(primaryStage);

indexWindow.getWindow().show();

}

public static void main(String[] args) {

launch(args);

}

}

// = = = IndexWindow = = =

public class IndexWindow extends BaseWindow<Index> {

// Путь к файлу в котором хранятся книги

static final private String DB\_TEXT\_URL = "src/com/labs/lab5/ELib/configs/books-db.txt";

static final private String DB\_BIN\_URL = "src/com/labs/lab5/ELib/configs/books-db";

// Хранилице книг - содежит все книги

private IStorage<Book> storage;

private Alerts alerts = new Alerts();

private IndexWindow() throws IOException { }

public IndexWindow(Stage window) throws IOException {

super(window);

}

protected void load() throws IOException {

var loader = new FXMLLoader(getClass().getResource("../views/index.fxml"));

loader.setControllerFactory(this::controllerFactory);

Parent root = loader.load();

Scene scene = new Scene(root);

controller = loader.getController();

window.setScene(scene);

window.setTitle("ELib - your world of books");

Image icon = new Image(getClass().getResource("../img/icon.png").toString());

window.getIcons().add(icon);

window.setMinWidth(700);

window.setMinHeight(400);

window.setMaximized(true);

}

private Index controllerFactory(Class<?> type) {

initStorage();

Index index = new Index(storage);

index.setOnExit(this::onExit);

return index;

}

private void initStorage() {

if (storage != null) return;

try {

storage = new BinaryStorage<>(DB\_BIN\_URL, Book.class);

//storage = new TextStorage<>(DB\_TEXT\_URL, Book::toString, Book::parse, Book.class);

} catch (IOException err) {

err.printStackTrace();

}

}

}

// = = = Index.java (main controller) = = =

public class Index implements Initializable {

@FXML private JFXTextField fxFilterName;

@FXML private JFXTextField fxFilterAuthor;

@FXML private JFXTextField fxFilterPublisher;

@FXML private JFXSlider fxFilterPriceFrom;

@FXML private JFXSlider fxFilterPriceTo;

@FXML private JFXSlider fxFilterPagesFrom;

@FXML private JFXSlider fxFilterPagesTo;

@FXML private JFXDatePicker fxFilterDateFrom;

@FXML private JFXDatePicker fxFilterDateTo;

@FXML private TableView<Book> fxBooksTable;

@FXML private TableColumn<Book, String> fxBooksTableColumnName;

@FXML private TableColumn<Book, String> fxBooksTableColumnAuthor;

@FXML private TableColumn<Book, String> fxBooksTableColumnPublisher;

@FXML private TableColumn<Book, Double> fxBooksTableColumnPrice;

@FXML private TableColumn<Book, Package> fxBooksTableColumnPages;

@FXML private TableColumn<Book, LocalDate> fxBooksTableColumnDate;

// Menu Events

@FXML private void fxOnMenuAbout() { onAbout(); }

@FXML private void fxOnMenuAddBook() { onAddBook(); }

@FXML private void fxOnMenuAppExit() { onExit(); }

@FXML private void fxOnMenuEditBook() { onEditBook(); }

@FXML private void fxOnMenuHelp() { onHelp(); }

@FXML private void fxOnMenuRemoveBooks() { onRemoveBook(); }

@FXML private void fxOnMenuResetFilters() { onResetFilters(); }

// Tools Events

@FXML private void fxOnToolAdd() { onAddBook(); }

@FXML private void fxOnToolEdit() { onEditBook(); }

@FXML private void fxOnToolRemove() { onRemoveBook(); }

// Filters Events

@FXML private void fxOnRunFilter() { onRunFilter(); }

@FXML private void fxOnResetFilters() { onResetFilters(); }

// Хранилице книг - содежит все книги

private IStorage<Book> storage;

// Массив книг удовлетворяющих фильтр (привязан к содержимому таблици)

private ObservableList<Book> filteredBooks = FXCollections.observableArrayList();

// Фильтры книг

private BookFilters filters = new BookFilters();

// Другие окна

private WindowCreateBook windowAddBook;

private WindowCreateBook windowEditBook;

private Alerts alerts = new Alerts();

// Граничные значение параметров книг

// Привязываються к минимальны/максимальным значения фильтров (fxml-элементов)

private final DoubleProperty minPrice = new SimpleDoubleProperty(0);

private final DoubleProperty maxPrice = new SimpleDoubleProperty(100);

private final IntegerProperty minPages = new SimpleIntegerProperty(0);

private final IntegerProperty maxPages = new SimpleIntegerProperty(100);

// Книга что редактируется в данный момент (проверка на null обязательна)

private Book editingBook;

public Index() { }

public Index(IStorage<Book> storage) {

this.storage = storage;

}

@Override

public void initialize(URL location, ResourceBundle resources) {

if (storage == null) {

alerts.show(alerts.getAlertErr(), "Can't load books");

onExit();

return;

}

updateFilterLimits();

resetFilters();

runFilter();

initTable();

}

private void runFilter() {

updateFilterLimits();

updateFilters();

updateFilteredBooks();

}

private void updateFilters() {

filters.setNameFilter(fxFilterName.getText());

filters.setAuthorFilter(fxFilterAuthor.getText());

filters.setPublisherFilter(fxFilterPublisher.getText());

filters.setPriceFromFilter(fxFilterPriceFrom.getValue());

filters.setPriceToFilter(fxFilterPriceTo.getValue());

filters.setPagesFromFilter((int)fxFilterPagesFrom.getValue());

filters.setPagesToFilter((int)fxFilterPagesTo.getValue());

filters.setDateFromFilter(fxFilterDateFrom.getValue());

filters.setDateToFilter(fxFilterDateTo.getValue());

}

private void updateFilteredBooks() {

filteredBooks.clear();

filteredBooks.addAll(storage.getArrOfData(book -> filters.check(book)));

}

private void updateFilterLimits() {

var books = storage.getArrOfData();

if (books.length == 0) return;

double \_minPrice = Arrays.stream(books).min(Comparator.comparingDouble(Book::getPrice)).get().getPrice() - 10;

double \_maxPrice = Arrays.stream(books).max(Comparator.comparingDouble(Book::getPrice)).get().getPrice() + 10;

int \_minPages = Arrays.stream(books).min(Comparator.comparingInt(Book::getPages)).get().getPages() - 10;

int \_maxPages = Arrays.stream(books).max(Comparator.comparingInt(Book::getPages)).get().getPages() + 10;

if (\_minPrice < 0) \_minPrice = 0;

if (\_minPages < 0) \_minPages = 0;

minPrice.set(\_minPrice);

minPages.set(\_minPages);

maxPrice.set(\_maxPrice);

maxPages.set(\_maxPages);

}

/\*\*

\* Сбрасывает фильтры (значения полей fxml)

\* Первый вызов должен быть после первого вызова метода initBinds()

\*/

private void resetFilters() {

filters.reset();

fxFilterName.clear();

fxFilterAuthor.clear();

fxFilterPublisher.clear();

fxFilterPriceFrom.setValue(minPrice.getValue());

fxFilterPriceTo.setValue(maxPrice.getValue());

fxFilterPagesFrom.setValue(minPages.getValue());

fxFilterPagesTo.setValue(maxPages.getValue());

fxFilterDateFrom.setValue(null);

fxFilterDateTo.setValue(null);

}

private void showWindowAddBook() {

initWindowAddBook();

windowAddBook.getController().setDateField(LocalDate.now());

windowAddBook.getWindow().show();

}

private void showWindowEditBook() {

// Редактируется ли уже какая-то книга

Book selected = fxBooksTable.getSelectionModel().getSelectedItem();

boolean alreadyEditing = false;

if (editingBook != null && editingBook == selected) {

alreadyEditing = true;

} else {

editingBook = selected;

}

if (editingBook == null) {

alerts.show(alerts.getAlertInfo(), "No selected books");

return;

}

initWindowEditBook();

if (!alreadyEditing) {

windowEditBook.getController().setValuesBy(editingBook);

}

windowEditBook.getWindow().show();

}

private void initWindowAddBook() {

if (windowAddBook != null) return;

try {

windowAddBook = new WindowCreateBook("Add New Book");

windowAddBook.getController().setOnSaveHandler(this::addNewBook);

} catch (IOException err) {

alerts.show(alerts.getAlertErr(), "Can't open the window");

}

}

private void initWindowEditBook() {

if (windowEditBook != null) return;

try {

windowEditBook = new WindowCreateBook("Edit The Book");

windowEditBook.getController().setOnSaveHandler(this::editBook);

} catch (IOException err) {

alerts.show(alerts.getAlertErr(), "Can't open the window");

System.out.println(err.toString());

}

}

/\*\*

\* Добавляет новую книгу в storage из windowAddBook

\*/

private void addNewBook() {

Book newBook = windowAddBook.getController().create();

if (newBook == null) {

alerts.show(alerts.getAlertErr(), "Incorrect data");

}

try {

storage.add(newBook);

} catch ( IOException err) {

alerts.show(alerts.getAlertErr(), "Something's wrong. Can't save the book");

return;

}

windowAddBook.getController().reset();

windowAddBook.getWindow().close();

runFilter();

fxBooksTable.getSelectionModel().select(newBook);

fxBooksTable.scrollTo(newBook);

}

/\*\*

\* Редактирует текущюю вибраную книгу

\* (на деле заменяет текущую выбранную книну на книгу созаную пользователем в windowEditBook)

\* Для правильной работы нужно убедиться, что за время, пока пользователь

\* Вводил данные книги выбранная книга в таблице не поменялась

\* (Например, окно windowEditBook может быть APPLICATION\_MODAL)

\*/

private void editBook() {

int selectedIndex = fxBooksTable.getSelectionModel().getSelectedIndex();

if (editingBook == null) {

alerts.show(alerts.getAlertErr(), "Something's wrong. Can't save the changes");

return;

}

Book edited = windowEditBook.getController().create();

if (edited == null) {

alerts.show(alerts.getAlertErr(), "Incorrect data");

return;

}

try {

storage.replace(editingBook, edited);

} catch (IOException err) {

alerts.show(alerts.getAlertErr(), "Something's wrong. Can't save the changes");

}

windowEditBook.getController().reset();

windowEditBook.getWindow().close();

editingBook = null;

runFilter();

fxBooksTable.getSelectionModel().select(selectedIndex);

}

private void removeSelectedBook() {

int selectedIndex = fxBooksTable.getSelectionModel().getSelectedIndex();

Book selected = fxBooksTable.getSelectionModel().getSelectedItem();

if (selected == null) {

alerts.show(alerts.getAlertInfo(), "No selected books");

return;

}

String mes = "Are you really want to remove '"+ selected.getName() +"'?";

var res = (alerts.show(alerts.getAlertConfirm(), mes));

if (!alerts.getAnswer(res)) {

return;

}

try {

storage.remove(selected);

} catch (IOException err) {

alerts.show(alerts.getAlertErr(), "Something's wrong. Can't remove the book");

}

runFilter();

// Восстановить выделение книги в таблице (обязательно после runFilter)

fxBooksTable.getSelectionModel().select(selectedIndex);

}

private void initTable() {

fxBooksTableColumnName.setCellValueFactory(new PropertyValueFactory<>("name"));

fxBooksTableColumnAuthor.setCellValueFactory(new PropertyValueFactory<>("author"));

fxBooksTableColumnPublisher.setCellValueFactory(new PropertyValueFactory<>("publisher"));

fxBooksTableColumnPrice.setCellValueFactory(new PropertyValueFactory<>("price"));

fxBooksTableColumnPages.setCellValueFactory(new PropertyValueFactory<>("pages"));

fxBooksTableColumnDate.setCellValueFactory(new PropertyValueFactory<>("date"));

fxBooksTable.setItems(filteredBooks);

}

private void onAddBook() {

showWindowAddBook();

}

private void onEditBook() {

showWindowEditBook();

}

private void onRunFilter() {

runFilter();

}

private void onRemoveBook() {

removeSelectedBook();

}

private void onHelp() {

alerts.show(alerts.getAlertInfo(), "Only You Can Help yourself!");

}

private void onAbout() {

alerts.show(alerts.getAlertInfo(), "ELib - Your World Of Books!");

}

private void onExit() {

onExitHandler.call();

}

private void onResetFilters() {

resetFilters();

runFilter();

}

}

// = = = CreateBook = = =

public class CreateBook implements Initializable {

@FXML private JFXTextField fxName;

@FXML private JFXTextField fxAuthor;

@FXML private JFXTextField fxPublisher;

@FXML private JFXTextField fxPrice;

@FXML private JFXTextField fxPages;

@FXML private JFXDatePicker fxDate;

@FXML public void fxOnSave() { onSave(); }

@FXML public void fxOnCancel() { onCancel(); }

private HandlerFunction onCancelHandler;

private HandlerFunction onSaveHandler;

private SimpleStringProperty title = new SimpleStringProperty("Create New Book");

private Alerts alerts = new Alerts();

@Override

public void initialize(URL location, ResourceBundle resources) {

setValidators();

initValidators();

}

/\*\*

\* Создает и возващает книгу на основании введенных пользователем данных

\* @return книга созданная на основании введенних данных

\*/

public Book create() {

if (isInvalid()) {

return null;

}

String name = fxName.getText();

String author = fxAuthor.getText();

String publisher = fxPublisher.getText();

double price = Double.parseDouble(fxPrice.getText());

int pages = Integer.parseInt(fxPages.getText());

LocalDate date = fxDate.getValue();

return new Book(name, author, publisher, date, pages, price);

}

/\*\*

\* Устанавливает в поля формы данних переданной книги

\* @param book книга данные которой будут установлены в поля формы

\*/

public void setValuesBy(Book book) {

fxName.setText(book.getName());

fxAuthor.setText(book.getAuthor());

fxPublisher.setText(book.getPublisher());

fxPrice.setText(Double.toString(book.getPrice()));

fxPages.setText(Integer.toString(book.getPages()));

fxDate.setValue(book.getDate());

}

public void reset() {

fxName.clear();

fxAuthor.clear();

fxPublisher.clear();

fxPrice.clear();

fxPages.clear();

fxDate.setValue(null);

}

/\*\*

\* @return можно ли ли формировать новую книгу из введенный в форме данных

\*/

public boolean isInvalid() {

// Все значения вычисляются до проверки, так как нужно что бы отработали все validate()

var name = fxName.validate();

var author = fxAuthor.validate();

var publisher = fxPublisher.validate();

var price = fxPrice.validate();

var pages = fxPages.validate();

var date = fxDate.validate();

return !(name && author && publisher

&& price && pages && date);

}

private void setValidators() {

var required = new RequiredFieldValidator("Required field");

fxName.getValidators().add(required);

fxAuthor.getValidators().add(required);

fxPublisher.getValidators().add(required);

fxPrice.getValidators().add(required);

fxPages.getValidators().add(required);

fxDate.getValidators().add(required);

fxPrice.getValidators().add(new DoubleValidator("Not a number"));

fxPages.getValidators().add(new IntegerValidator("Not an integer"));

}

private void initValidators() {

initValidators(fxName);

initValidators(fxAuthor);

initValidators(fxPublisher);

initValidators(fxPrice);

initValidators(fxPages);

initValidators(fxDate);

}

private void initValidators(JFXTextField node) {

node.textProperty().addListener((o, oldVal, newVal) -> node.validate());

node.focusedProperty().addListener((o, oldVal, newVal) -> {

// Обход ошибки jfoenix, при которой валидатор не правильно ратает

// Когда текст в поле выравнен не по левому краю

node.setAlignment(Pos.TOP\_LEFT);

if (!newVal) node.validate();

node.setAlignment(Pos.CENTER);

});

}

private void initValidators(JFXDatePicker node) {

node.valueProperty().addListener((o, oldVal, newVal) -> node.validate());

node.focusedProperty().addListener((o, oldVal, newVal) -> {

if (!newVal) node.validate();

});

}

}

// = = = index.fxml = = =

<BorderPane styleClass="wrapper" stylesheets="@../styles/index.css" xmlns="http://javafx.com/javafx/10.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.labs.lab5.ELib.controllers.Index">

<top>

<VBox BorderPane.alignment="CENTER">

<children>

<!-- Menu -->

<MenuBar styleClass="menu">

<menus>

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

<items>

<MenuItem fx:id="fxMenuAddBook" mnemonicParsing="false" onAction="#fxOnMenuAddBook" text="Add new book" />

<MenuItem fx:id="fxMenuResetFilters" mnemonicParsing="false" onAction="#fxOnMenuResetFilters" text="Reset filters" />

<SeparatorMenuItem mnemonicParsing="false" />

<MenuItem fx:id="fxMenuAppExit" mnemonicParsing="false" onAction="#fxOnMenuAppExit" text="Exit" />

</items>

</Menu>

<Menu mnemonicParsing="false" text="Edit">

<items>

<MenuItem fx:id="fxMenuEditBook" mnemonicParsing="false" onAction="#fxOnMenuEditBook" text="Edit the book" />

<MenuItem fx:id="fxMenuRemoveBooks" mnemonicParsing="false" onAction="#fxOnMenuRemoveBooks" text="Remove the book" />

</items>

</Menu>

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

<items>

<MenuItem fx:id="fxMenuHelp" mnemonicParsing="false" onAction="#fxOnMenuHelp" text="Help" />

<MenuItem fx:id="fxMenuAbout" mnemonicParsing="false" onAction="#fxOnMenuAbout" text="About" />

</items>

</Menu>

</menus>

</MenuBar>

<!-- Toolbar -->

<ToolBar nodeOrientation="RIGHT\_TO\_LEFT" styleClass="toolbar">

<items>

<JFXButton fx:id="fxToolRemove" onAction="#fxOnToolRemove" styleClass="toolbar\_\_tool" text="Remove" />

<JFXButton fx:id="fxToolEdit" onAction="#fxOnToolEdit" styleClass="toolbar\_\_tool" text="Edit" />

<JFXButton fx:id="fxToolAdd" onAction="#fxOnToolAdd" styleClass="toolbar\_\_tool" text="Add" />

</items>

</ToolBar>

</children></VBox>

</top>

<center>

<SplitPane dividerPositions="0.1" prefHeight="200.0" prefWidth="200.0" BorderPane.alignment="CENTER">

<items>

<ScrollPane fitToHeight="true" fitToWidth="true" focusTraversable="true" nodeOrientation="LEFT\_TO\_RIGHT" styleClass="filters">

<content>

<!-- Filters -->

<VBox spacing="15.0">

<children>

<VBox prefHeight="200.0" prefWidth="100.0">

<children>

<Label styleClass="filters\_\_title" text="Filters" />

</children>

</VBox>

<VBox styleClass="filterGroup">

<Label styleClass="filterGroup\_\_title" text="Name:" />

<JFXTextField fx:id="fxFilterName" focusColor="#757575" onAction="#fxOnRunFilter" onKeyTyped="#fxOnRunFilter" promptText="The name" styleClass="filterGroup\_\_field" />

<VBox.margin>

<Insets />

</VBox.margin>

</VBox>

<VBox styleClass="filterGroup">

<Label styleClass="filterGroup\_\_title" text="Author:" />

<JFXTextField fx:id="fxFilterAuthor" focusColor="#757575" onAction="#fxOnRunFilter" onKeyTyped="#fxOnRunFilter" promptText="The author" styleClass="filterGroup\_\_field" />

<VBox.margin>

<Insets />

</VBox.margin>

</VBox>

<VBox styleClass="filterGroup">

<Label styleClass="filterGroup\_\_title" text="Publisher:" />

<JFXTextField fx:id="fxFilterPublisher" focusColor="#757575" onAction="#fxOnRunFilter" onKeyTyped="#fxOnRunFilter" promptText="The publisher" styleClass="filterGroup\_\_field" />

</VBox>

<VBox styleClass="filterGroup">

<Label styleClass="filterGroup\_\_title" text="Price:" />

<Label styleClass="filterGroup\_\_subtitle" text="From:" />

<JFXSlider fx:id="fxFilterPriceFrom" min="${controller.minPrice}" max="${controller.maxPrice}" majorTickUnit="100.0" onKeyReleased="#fxOnRunFilter" onMouseClicked="#fxOnRunFilter" showTickLabels="true" showTickMarks="true" value="0" />

<Label styleClass="filterGroup\_\_subtitle" text="To:" />

<JFXSlider fx:id="fxFilterPriceTo" min="${controller.minPrice}" max="${controller.maxPrice}" majorTickUnit="100.0" onKeyReleased="#fxOnRunFilter" onMouseClicked="#fxOnRunFilter" showTickLabels="true" showTickMarks="true" value="0" />

</VBox>

<!-- min="${controller.minPrice}" max="${controller.maxPrice}"-->

<VBox styleClass="filterGroup">

<Label styleClass="filterGroup\_\_title" text="Pages:" />

<Label styleClass="filterGroup\_\_subtitle" text="From:" />

<JFXSlider fx:id="fxFilterPagesFrom" min="${controller.minPages}" max="${controller.maxPages}" majorTickUnit="100.0" onKeyReleased="#fxOnRunFilter" onMouseClicked="#fxOnRunFilter" showTickLabels="true" showTickMarks="true" value="0" />

<Label styleClass="filterGroup\_\_subtitle" text="To:" />

<JFXSlider fx:id="fxFilterPagesTo" majorTickUnit="100.0" onKeyReleased="#fxOnRunFilter" onMouseClicked="#fxOnRunFilter" showTickLabels="true" showTickMarks="true" value="0" />

</VBox>

<VBox styleClass="filterGroup">

<Label styleClass="filterGroup\_\_title" text="Date:" />

<Label styleClass="filterGroup\_\_subtitle" text="From:" />

<JFXDatePicker fx:id="fxFilterDateFrom" defaultColor="rgba(117,117,117,1)" onAction="#fxOnRunFilter" onKeyTyped="#fxOnRunFilter">

<VBox.margin>

<Insets bottom="10.0" />

</VBox.margin>

</JFXDatePicker>

<Label styleClass="filterGroup\_\_subtitle" text="To:" />

<JFXDatePicker fx:id="fxFilterDateTo" defaultColor="rgba(117,117,117,1)" onAction="#fxOnRunFilter" onKeyTyped="#fxOnRunFilter">

<padding>

<Insets bottom="10.0" />

</padding>

</JFXDatePicker>

</VBox>

<VBox nodeOrientation="RIGHT\_TO\_LEFT">

<children>

<JFXButton fx:id="fxResetFilters" buttonType="RAISED" onAction="#fxOnResetFilters" styleClass="filters\_\_reset" text="Reset filters" />

</children>

</VBox>

</children>

<padding>

<Insets bottom="15.0" left="15.0" right="15.0" top="15.0" />

</padding>

</VBox>

</content>

</ScrollPane>

<AnchorPane prefHeight="200.0" prefWidth="200.0">

<children>

<VBox AnchorPane.leftAnchor="10.0">

<children>

<Label styleClass="books\_\_title" text="Books" />

</children>

</VBox>

<TableView fx:id="fxBooksTable" AnchorPane.bottomAnchor="10.0" AnchorPane.leftAnchor="10.0" AnchorPane.rightAnchor="10.0" AnchorPane.topAnchor="40.0">

<columns>

<TableColumn fx:id="fxBooksTableColumnName" text="Name" minWidth="150" />

<TableColumn fx:id="fxBooksTableColumnAuthor" text="Author" minWidth="150" />

<TableColumn fx:id="fxBooksTableColumnPublisher" text="Publisher" minWidth="150" />

<TableColumn fx:id="fxBooksTableColumnPrice" text="Price" minWidth="90" />

<TableColumn fx:id="fxBooksTableColumnPages" text="Pages" minWidth="60" />

<TableColumn fx:id="fxBooksTableColumnDate" text="Date" minWidth="100" />

</columns>

<columnResizePolicy>

<TableView fx:constant="CONSTRAINED\_RESIZE\_POLICY" />

</columnResizePolicy>

</TableView>

</children>

</AnchorPane>

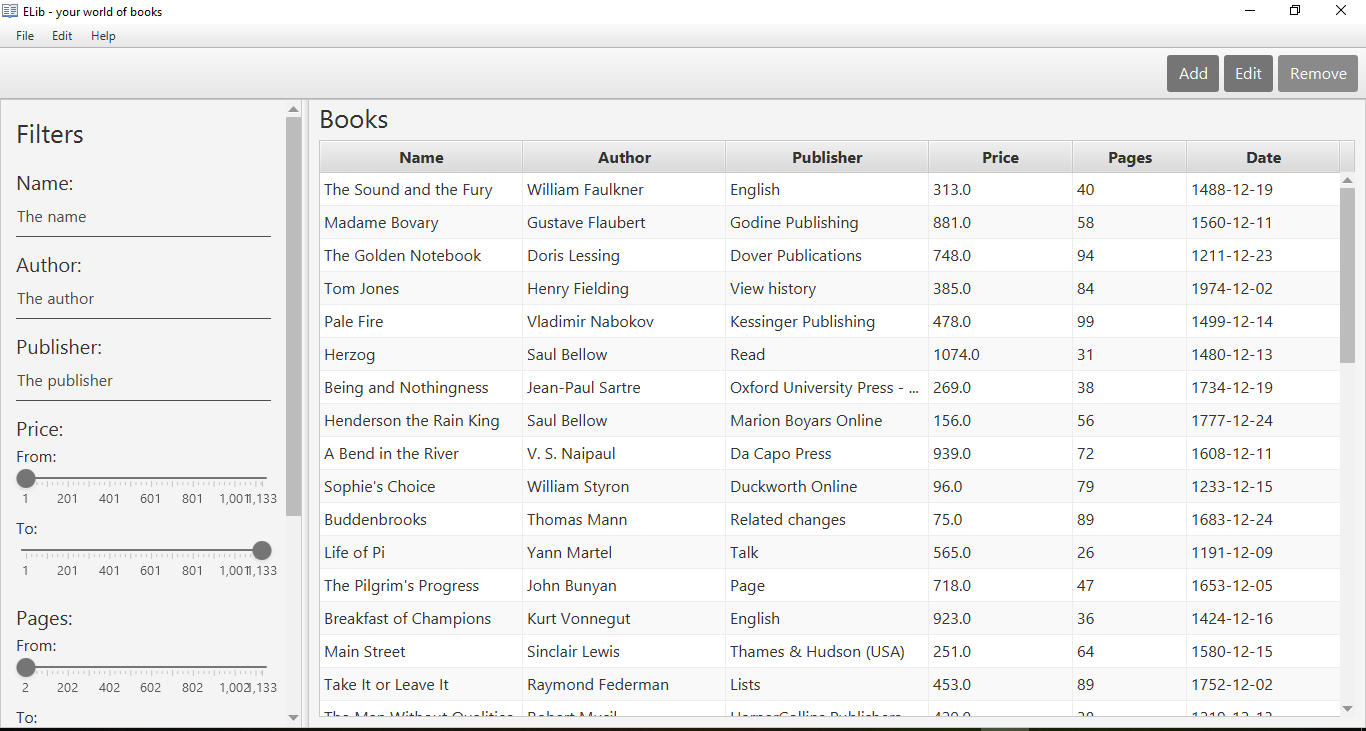
</items>

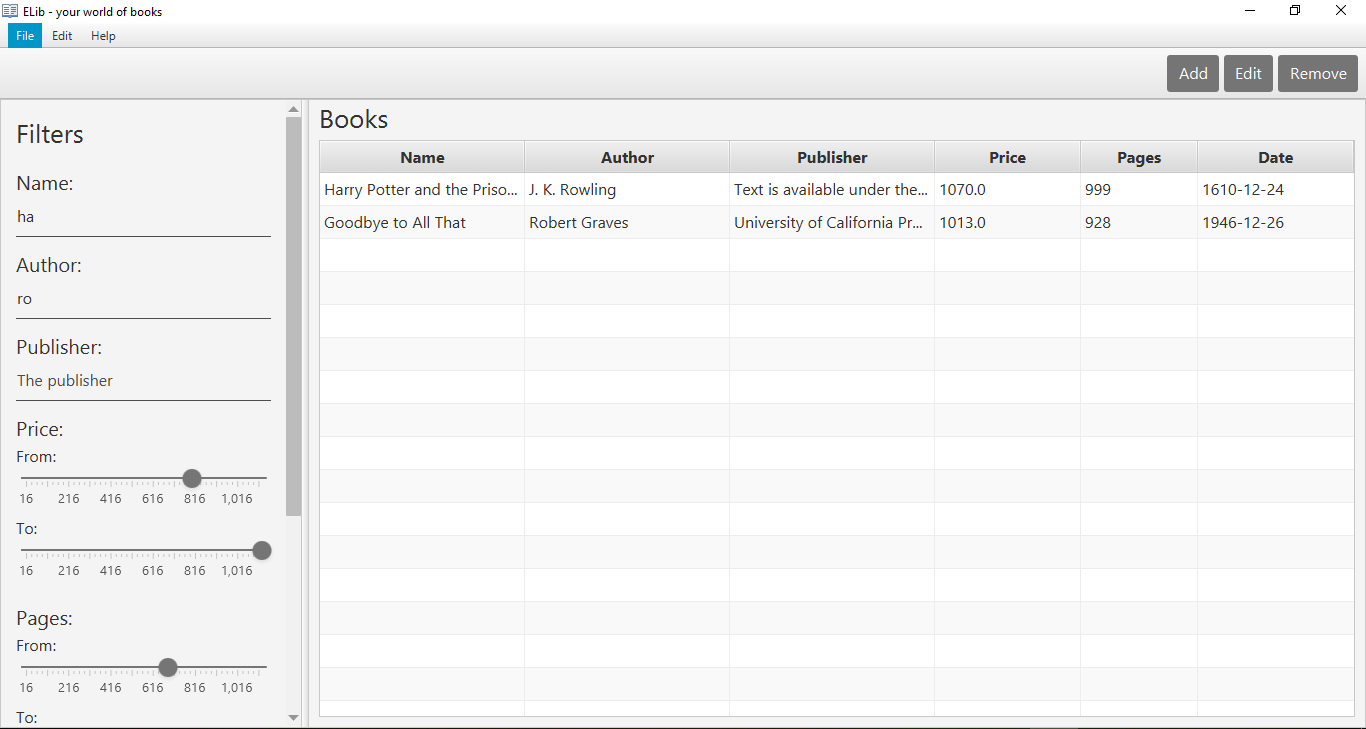
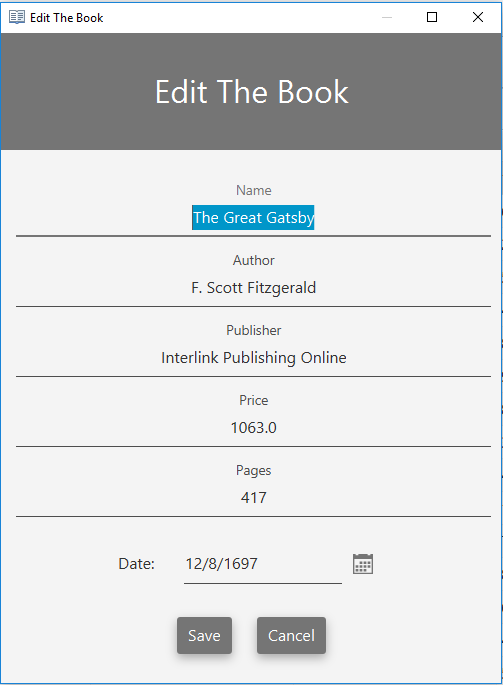
</SplitPane>

</center>

</BorderPane>

1. Результат виконання програми:



**Висновок:** на цій лабораторній роботі я отримав навички створення та реалізації програм, що реалізують операції введення-виведення із файлами.