تقرير مشروع

نظام إدارة المكتبة باستخدام JavaFX

١. فكرة المشروع

اخترت أن أبني نظاماً صغيراً لإدارة محتويات مكتبة، يسمح للمستخدمة بإضافة كتاب جديد إلى المجموعة، وعرض كل الكتب المضافة داخل جدول، وإمكانية حذف أي كتاب عند الحاجة. التطبيق مكتبي بواجهة مبنية بـ JavaFX، وكل كتاب يتم تمثيله ككائن مستقل يخزَّن في قائمة بالذاكرة.

المكتبة لا تتعامل مع نوع واحد من الكتب فقط، فالكتب اليوم إما ورقية تحفظ على رفوف فعلية، أو إلكترونية تخزَّن كملفات بصيغ مختلفة. ولأن النوعين يشتركان في معلومات أساسية كالعنوان والمؤلفة وسنة النشر، ويختلفان في طريقة التخزين والوصف، استخدمت كلاساً مجرداً يجمع الصفات المشتركة، وكلاسين يرثان منه: واحد للكتاب الورقي والآخر للكتاب الإلكتروني، مع إجبار كل منهما على تنفيذ ميثوديه الخاصتين بطريقته.

الهدف التعليمي من المشروع هو التطبيق العملي للوراثة والكلاسات المجردة وإلزامية تنفيذ الميثودات في الكلاسات الوارثة، مع ربط كل ذلك بواجهة رسومية تتعامل مع المستخدمة بشكل مباشر.

١,١ المتطلبات التي تم تنفيذها

- بناء واجهة JavaFX تحتوي على حقول إدخال وأزرار وقائمة منسدلة وجدول عرض.
- إنشاء كلاس مجرد يحوي الصفات المشتركة بين الكتب وميثودين abstract.
- إنشاء كلاسين يرثان من الكلاس المجرد، كل منهما يضيف حقولاً خاصة به.

- إنشاء الكائن من النوع المناسب وقت الإدخال حسب اختيار المستخدمة من القائمة.
- تخزين الكائنات في LinkedList واستعراضها داخل الجدول.
- إمكانية حذف عنصر محدد من القائمة والجدول معاً.

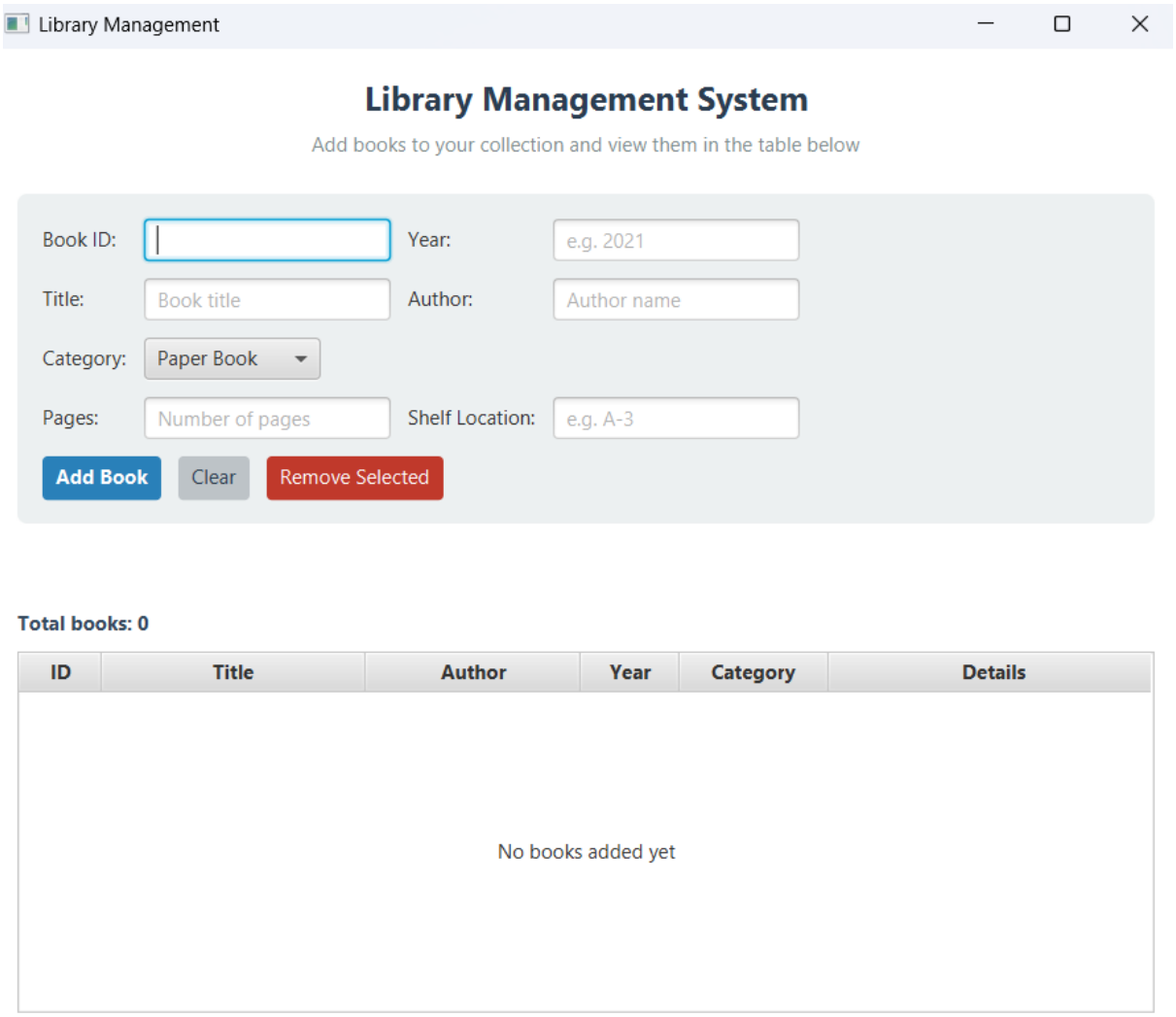
١,٢ الأدوات المستخدمة

- لغة Java بإصدار JDK ١٧.
- مكتبة JavaFX بإصدار ٢١,٠,١.
- أداة Maven لإدارة المكتبات.
- بيئة Apache NetBeans للتطوير.

٢. صور الواجهات

٢,١ الواجهة عند فتح التطبيق

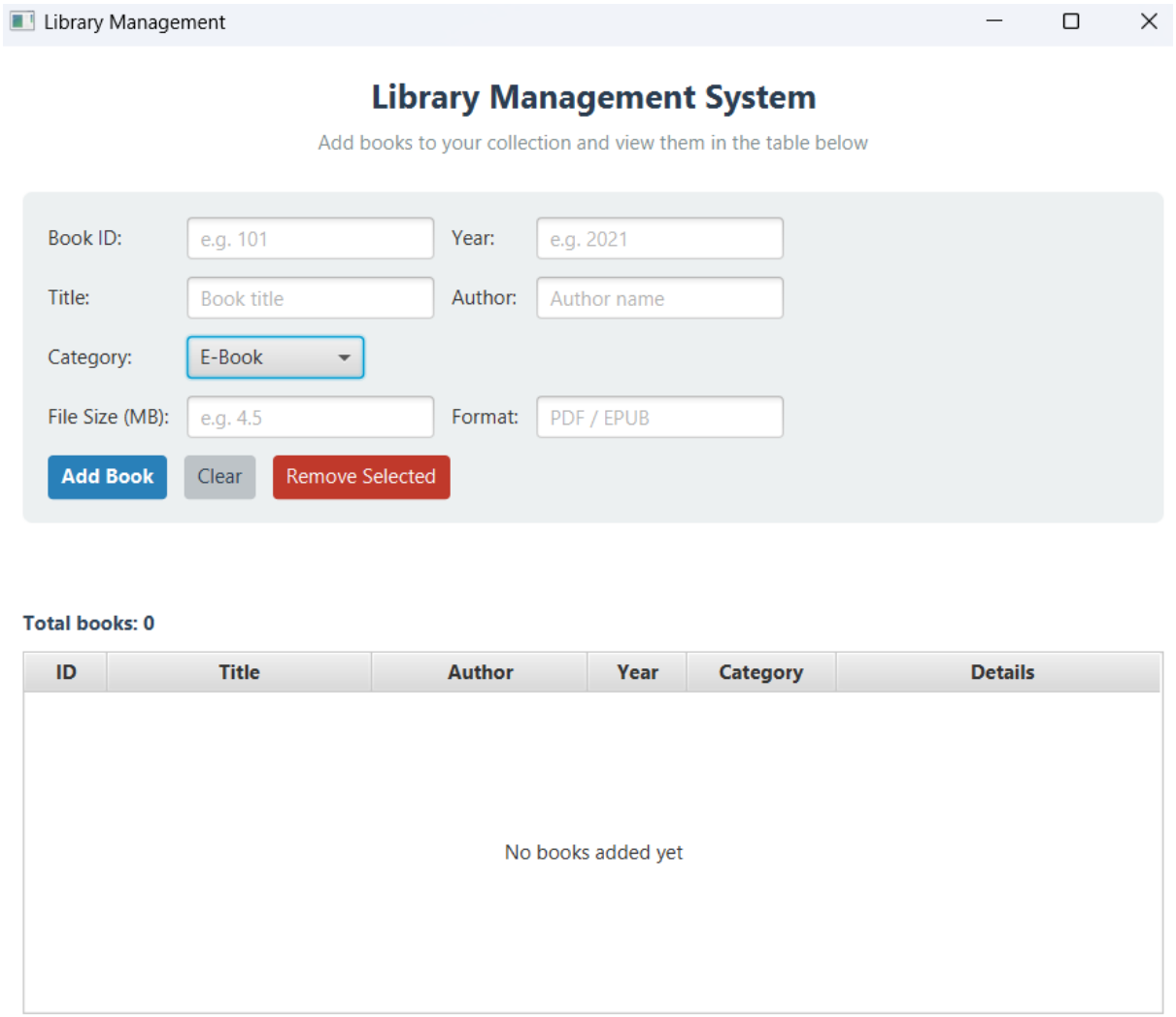
بمجرد تشغيل البرنامج تظهر الواجهة الرئيسية: شريط عنوان في الأعلى، ثم نموذج الإدخال على خلفية رمادية فاتحة يحوي حقول الـ ID والعنوان والمؤلفة وسنة النشر، وقائمة منسدلة لنوع الكتاب مضبوطة افتراضياً على Paper Book، ويظهر تحتها حقلان إضافيان لعدد الصفحات وموقع الرف. أسفل النموذج ثلاثة أزرار: Add Book و Clear و Remove Selected.



شكل ١: الواجهة الرئيسية عند تشغيل التطبيق

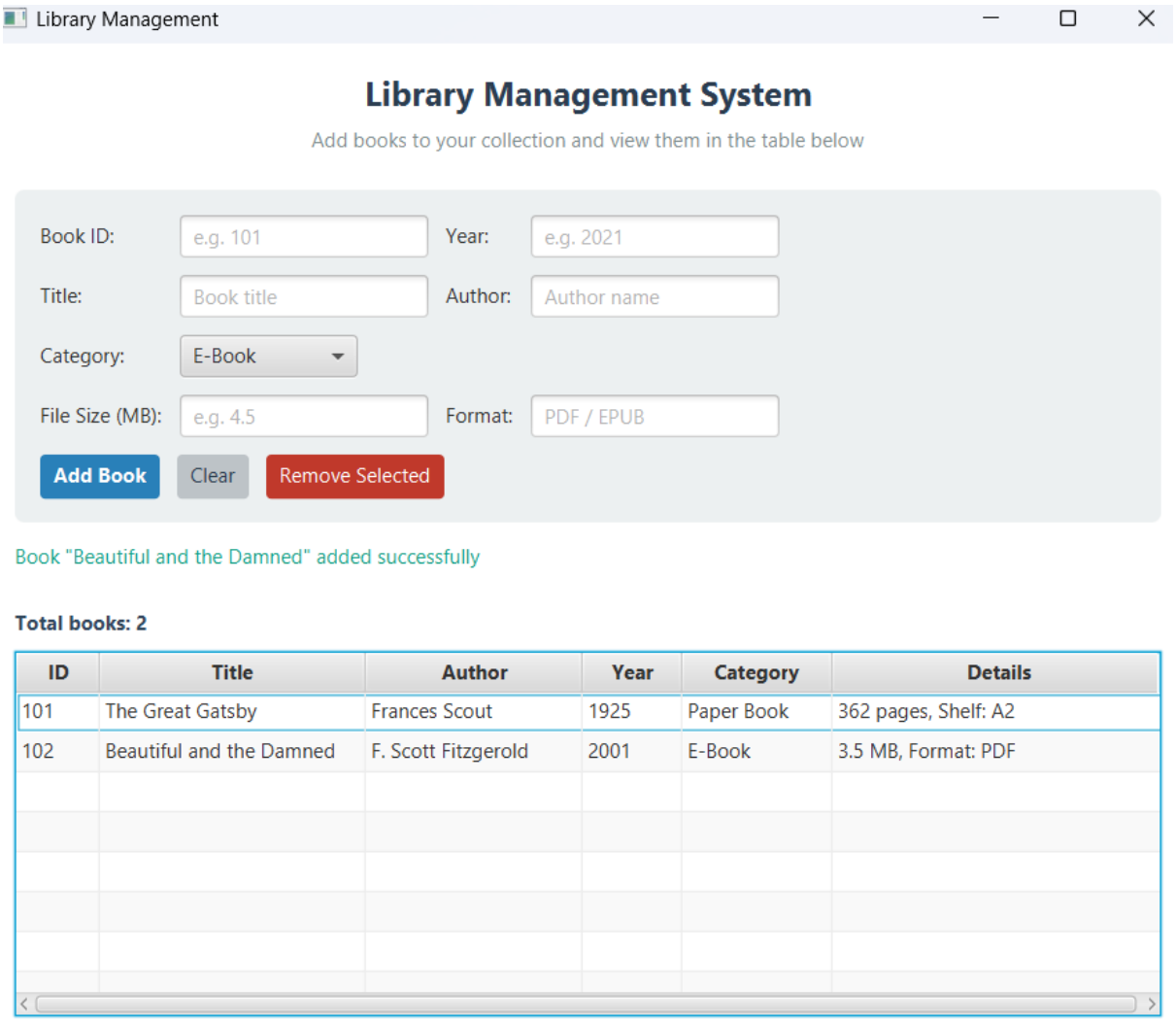
٢,٢ الواجهة عند تغيير نوع الكتاب إلى Electronic

عند فتح القائمة المنسدلة واختيار E-Book، يتغير اسم الحقل الإضافي الأول من Pages إلى File Size (MB)، ويتغير الحقل الثاني من Shelf Location إلى Format. هذا التحويل يحدث تلقائياً بفضل ربط حدث setOnAction بالقائمة، وهو ما يجعل الواجهة الواحدة تكفي لإدخال النوعين دون الحاجة إلى نافذتين منفصلتين.



شكل ٢: تغيُّر الحقول الإضافية عند اختيار الكتاب الإلكتروني

٢,٣ الواجهة بعد إضافة عدة كتب



شكل ٣: الجدول بعد إضافة كتب من النوعين

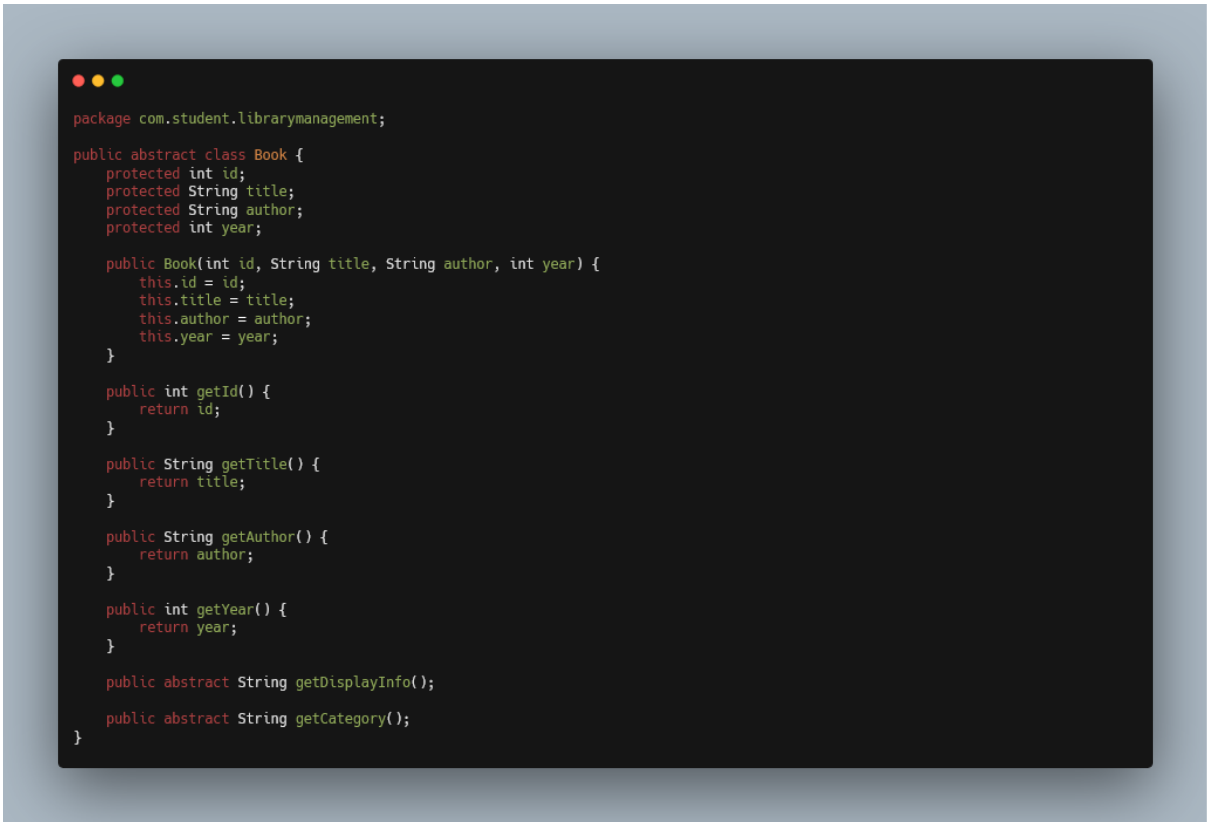
٣. الكود مع الشرح

المشروع موزَّع على أربعة ملفات داخل الحزمة com.student.librarymanagement. كل ملف منها مسؤول عن جزء واحد من المنظومة، وهذا التقسيم يسهِّل القراءة والتعديل لاحقاً.

٣,١ الكلاس المجرد Book

هذا هو الكلاس الأب لكل أنواع الكتب. تم تعريفه بكلمة abstract لأن مفهوم كتاب بشكل عام لا معنى له بدون نوع محدد، وبالتالي لا يجب أن تُنشَأ منه كائنات مباشرة. يحوي الحقول المشتركة

بين كل الكتب: id والعنوان واسم المؤلفة وسنة النشر، مع الـ getters المعتادة. يحوي أيضاً ميثودين abstract هما getDisplayInfo التي تعيد وصف التخزين الخاص بكل نوع، و getCategory التي تعيد اسم الفئة. هذا الإلزام يضمن أن أي كلاس وارث جديد سيكون مجبراً على تعريف هاتين الميثودين بأسلوبه الخاص.



```java
package com.student.librarymanagement;

public abstract class Book {
    protected int id;
    protected String title;
    protected String author;
    protected int year;

    public Book(int id, String title, String author, int year) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.year = year;
    }

    public int getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public int getYear() {
        return year;
    }

    public abstract String getDisplayInfo();

    public abstract String getCategory();
}
```

٣,٢ الكلاس PaperBook

يرث من Book ويضيف حقلين خاصين بالكتاب الورقي: عدد الصفحات pageCount وموقع الرف shelfLocation. الميثود getDisplayInfo تعيد عبارة تجمع عدد الصفحات ورمز الرف، أما getCategory فتعيد النص Paper Book. الفائدة من فصلِها في كلاس مستقل أن أي تعديل مستقبلي يخص الكتب الورقية فقط (مثلاً إضافة حقل حالة الغلاف أو عدد النسخ المتاحة) سيتم في هذا الملف وحده، دون المساس بالكتب الإلكترونية.

```java
package com.student.librarymanagement;

public class Paperbook extends Book {
    private int pageCount;
    private String shelfLocation;

    public Paperbook(int id, String title, String author, int year,
                     int pageCount, String shelfLocation) {
        super(id, title, author, year);
        this.pageCount = pageCount;
        this.shelfLocation = shelfLocation;
    }

    public int getPageCount() {
        return pageCount;
    }

    public String getShelfLocation() {
        return shelfLocation;
    }

    @Override
    public String getDisplayInfo() {
        return pageCount + " pages, Shelf: " + shelfLocation;
    }

    @Override
    public String getCategory() {
        return "Paper Book";
    }
}
```

٣,٣ الكلاس EBook

النظير الإلكتروني للكتاب الورقي. يضيف حقلين: حجم الملف بالميجابايت fileSizeMB وصيغة الملف fileFormat (مثل PDF أو EPUB). يلاحَظ أن نوع البيانات هنا double وليس int، وهذا فرق منطقي لأن أحجام الملفات نادراً ما تكون أعداداً صحيحة. ميثود getDisplayInfo هنا تعيد نصاً مختلفاً تماماً عن نظيرتها في PaperBook رغم أنهما تحملان الاسم نفسه، وهذا هو جوهر مفهوم Polymorphism.

```java
package com.student.librarymanagement;

public class Ebook extends Book {
    private double fileSizeMB;
    private String fileFormat;

    public Ebook(int id, String title, String author, int year,
                 double fileSizeMB, String fileFormat) {
        super(id, title, author, year);
        this.fileSizeMB = fileSizeMB;
        this.fileFormat = fileFormat;
    }

    public double getFileSizeMB() {
        return fileSizeMB;
    }

    public String getFileFormat() {
        return fileFormat;
    }

    @Override
    public String getDisplayInfo() {
        return fileSizeMB + " MB, Format: " + fileFormat;
    }

    @Override
    public String getCategory() {
        return "E-Book";
    }
}
```

٣,٤ الكلاس Main (الواجهة والتحكم)

هو الكلاس الذي يرث من Application ويحتوي على ميثود start المسؤولة عن بناء كل عناصر الواجهة. اخترت هنا استخدام LinkedList بدلاً من ArrayList لتخزين الكتب، لأن عمليات الإضافة المتكررة في نهاية القائمة وعمليات الحذف من المنتصف (عند الضغط على Remove Selected) تكون أكثر كفاءة في LinkedList. وإلى جانب القائمة الأصلية يوجد ObservableList مرتبطة بالجدول حتى يتحدَّث الجدول تلقائياً مع كل إضافة أو حذف.

```java
package com.student.librarymanagement;


import java.util.LinkedList;

public class Main extends Application {

    private LinkedList<Book> library = new LinkedList<>();
    private ObservableList<Book> tableData = FXCollections.observableArrayList();

    @Override
    public void start(Stage stage) {
        Label headerLabel = new Label("Library Management System");
        headerLabel.setStyle("-fx-font-size: 20px; -fx-font-weight: bold; -fx-text-fill: #2C3E50;");

        Label subHeader = new Label("Add books to your collection and view them in the table below");
        subHeader.setStyle("-fx-font-size: 12px; -fx-text-fill: #7F8C8D;");

        VBox headerBox = new VBox(4, headerLabel, subHeader);
        headerBox.setAlignment(Pos.CENTER);
        headerBox.setPadding(new Insets(15, 10, 10, 10));

        Label idLabel = new Label("Book ID:");
        TextField idField = new TextField();
        idField.setPromptText("e.g. 101");

        Label titleLabel = new Label("Title:");
        TextField titleField = new TextField();
        titleField.setPromptText("Book title");

        Label authorLabel = new Label("Author:");
        TextField authorField = new TextField();
        authorField.setPromptText("Author name");

        Label yearLabel = new Label("Year:");
        TextField yearField = new TextField();
        yearField.setPromptText("e.g. 2021");

        Label categoryLabel = new Label("Category:");
        ComboBox<String> categoryBox = new ComboBox<>();
        categoryBox.getItems().addAll("Paper Book", "E-Book");
        categoryBox.setValue("Paper Book");

        Label firstExtraLabel = new Label("Pages:");
        TextField firstExtraField = new TextField();
        firstExtraField.setPromptText("Number of pages");

        Label secondExtraLabel = new Label("Shelf Location:");
        TextField secondExtraField = new TextField();
        secondExtraField.setPromptText("e.g. A-3");

    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

٤. الخلاصة

نُفِّذت كل بنود المشروع المطلوبة: واجهة JavaFX تحتوي على الحقول والأزرار والقائمة المنسدلة، وكلاس مجرد واحد بميثودين abstract، وكلاسان وارثان كل منهما له حقوله الخاصة وتنفيذه الخاص للميثودين. الكائنات تُنشَأ من النوع المناسب وقت الإدخال، وتُخزَّن في LinkedList، وتُعرَض داخل الجدول مع إمكانية الحذف.

أكثر ما استفدت منه أثناء العمل هو الفرق العملي بين التشابه الكامل و التشابه الجزئي بين الكلاسات. الكتاب الورقي والإلكتروني يشتركان في الكثير، لكن طريقة وصف كل منهما مختلفة. الكلاس المجرد فرض عليّ أن أعترف بهذا الاختلاف صراحةً وأنفِّذ ميثوداً مستقلة لكل نوع، بدلاً من تجميع كل الحالات في كلاس واحد ضخم بجمل if. كما لاحظت أن التصميم بهذا الشكل يجعل إضافة نوع ثالث مستقبلاً (مثل كتاب صوتي AudioBook بحقول إضافية كمدة التسجيل والصيغة الصوتية) أمراً سهلاً جداً ولا يتطلب تعديل الواجهة سوى بإضافة خيار جديد للقائمة المنسدلة.