

Library Management System — Project Report

Overview

A simple console-based Library Management System implemented in Java. It supports two user types (Student, Faculty), allows adding multiple borrowed books per person, and displays person details and their borrowed books.

Objectives

- Demonstrate object-oriented design with inheritance and polymorphism.
- Allow a user (Student or Faculty) to enter name/email and borrow multiple books.
- Provide a clean console UX and basic input validation.

Project Structure

- `src/Person.java` — abstract base class (fields: `name`, `email`; abstract `displayDetails()`).
- `src/Student.java` — extends `Person`; holds list of borrowed books and implements `displayDetails()`.
- `src/Faculty.java` — extends `Person`; holds list of borrowed books and implements `displayDetails()`.
- `src/Book.java` — represents a book (title, author, ISBN, price) and `displayInfo()`.
- `src/Library.java` — console UI, input loop to add books, manages `issuedBooks` counter and prints person details.

Design / UML (summary)

- `Person` (abstract)
 - subclasses: `Student`, `Faculty`
 - each subclass maintains a list of `Book`
- `Library` handles user interaction and uses polymorphism to print details.
- `Book` models the book entity.

Classes & Responsibilities

- `Person`: store name/email; enforce `displayDetails()` contract.
- `Student` / `Faculty`: initialize and manage their borrowed book lists; implement `displayDetails()`.
- `Book`: store and print book attributes.
- `Library`: prompt user for person type, name, email; loop to add books; print final details.

How to run

- Follow console prompts:
 - Choose 1 (Student) or 2 (Faculty)
 - Enter name and email
 - Enter books: Title / Author / ISBN / Price
 - Enter `0` as Title to finish and display details

Example interaction (condensed)

- Select user type: `1`

- Name: `Alice`
- Email: `alice@example.com`
- Title: `Java 101`
- Author: `J. Doe`
- ISBN: `12345`
- Price: `29.99`
- Title: `0` (finish)
- Output: prints name, email, and borrowed book info

Testing

- Manual tests: choose both Student and Faculty, add multiple books, enter `0` as title immediately, enter invalid price (non-number).
- Recommended: unit tests for `Book.displayInfo()`, `Student`/`Faculty` borrowing logic, and input parsing utilities.

End of report.

Screenshots

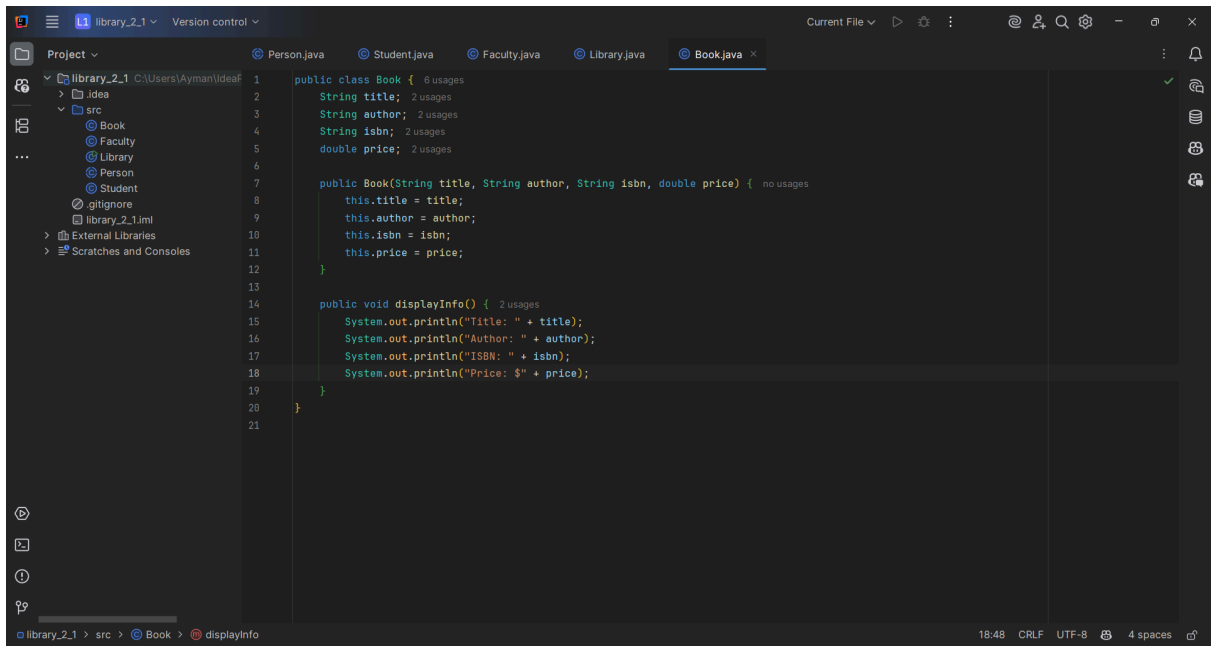
```

1  import java.util.Scanner;
2  public class Library {
3      static int pchoice=1; // 1: add book, 2: borrow book, 3: return book, 4: display info, 5: exit
4      static int issuedBooks = 0; // 1: usage
5      public static void main(String[] args) {
6          Person person = null;
7          Scanner sc = new Scanner(System.in);
8          System.out.println("Library Management System---\n You are a: \n1. Student \n2. Faculty");
9          int pchoice = sc.nextInt();
10         } while (pchoice != 1 && pchoice != 2);
11         sc.nextLine();
12
13         String name, email;
14         System.out.println("Name: ");
15         if (pchoice == 1) {
16             name = sc.nextLine();
17             System.out.println("Email: ");
18             email = sc.nextLine();
19             person = new Student(name, email);
20         } else if (pchoice == 2) {
21             name = sc.nextLine();
22             System.out.println("Email: ");
23             email = sc.nextLine();
24             person = new Faculty(name, email);
25         }
26         System.out.println("Borrow Books or enter 0 to exit system: ");
27         while(true){
28             Book book = new Book(0, null, 0, null, 0, 0);
29             System.out.println("Title:");
30             book.title = sc.nextLine();
31             if ("0".equals(book.title.trim())) {
32                 printPersonDetails(person);
33                 System.out.println("Exiting system. Goodbye!");
34                 return;
35             } else {
36                 System.out.println("Author:");

```

```
1 abstract public class Person { 3 usages 2 inheritors
2     String name; 3 usages
3     String email; 3 usages
4
5     abstract void displayDetails(); 1 usage 2 implementations
6
7     public Person(String name, String email) 2 usages
8     {
9         this.name = name;
10        this.email = email;
11    }
12 }
```

```
1 import java.util.ArrayList;
2
3 public class Student extends Person { no usages
4
5     ArrayList<Book> BorrowedBooks; 2 usages
6
7     public void displayDetails() { 1 usage
8         System.out.println("Name: " + name);
9         System.out.println("Email: " + email);
10        System.out.println("Borrowed Books: ");
11        for (Book book : BorrowedBooks) {
12            book.displayInfo();
13            System.out.println("-----");
14        }
15    }
16
17    public Student(String name, String email) no usages
18    {
19        super(name, email);
20        BorrowedBooks = new ArrayList<Book>();
21    }
22 }
```



The screenshot shows an IDE window for a project named 'library_2.1'. The 'Project' view on the left lists the file structure, including 'src' and 'Book.java'. The 'Book.java' file is open in the editor, showing the following code:

```
1 public class Book { 6 usages
2     String title; 2 usages
3     String author; 2 usages
4     String isbn; 2 usages
5     double price; 2 usages
6
7     public Book(String title, String author, String isbn, double price) { no usages
8         this.title = title;
9         this.author = author;
10        this.isbn = isbn;
11        this.price = price;
12    }
13
14    public void displayInfo() { 2 usages
15        System.out.println("Title: " + title);
16        System.out.println("Author: " + author);
17        System.out.println("ISBN: " + isbn);
18        System.out.println("Price: $" + price);
19    }
20 }
21
```

The status bar at the bottom indicates the file is 'library_2.1 > src > Book > displayInfo', the time is 18:48, and the encoding is UTF-8 with 4 spaces.