

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.

- 3 -

End of report.

Screenshots

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "library_2.1".
- File List:** The files listed are Person.java, Student.java, Faculty.java, Library.java (the current file), uml.idea.txt, and Book.java.
- Code Editor:** The code for Library.java is displayed, which contains a main method for a Library class. It uses a Scanner to get user input for choice (1 for Student, 2 for Faculty) and name/email. It then creates a Student or Faculty object and prints their details. Finally, it enters a loop where the user can borrow books or exit.
- Toolbars and Status Bar:** Standard IntelliJ IDEA toolbars and status bar are visible at the top and bottom of the interface.

library_2_1

Project library_2_1 C:\Users\Ayman\idea1

Person.java

```
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) {
8         this.name = name;
9         this.email = email;
10    }
11}
12
```

Current File Version control

Scratches and Consoles

External Libraries

Scratches and Consoles

10:6 CRLF UTF-8 4 spaces

library_2_1

Project library_2_1 C:\Users\Ayman\idea1

Student.java

```
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    public Student(String name, String email) { no usages
17        super(name, email);
18        Borrowedbooks = new ArrayList<Book>();
19    }
20}
21
```

Current File Version control

Scratches and Consoles

External Libraries

Scratches and Consoles

19:6 CRLF UTF-8 4 spaces

The screenshot shows a Java code editor interface with the following details:

- Project:** library_2_1
- File:** Book.java
- Code Content:**

```
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
```

- Toolbars and Status Bar:** Current File, Version control, Project, Person.java, Student.java, Faculty.java, Library.java, Book.java, 18:48, CRLF, UTF-8, 4 spaces.