Library Management System - User Manual
Mohammad Hossain
SD 11

Welcome to the **Library Management System**. This application provides a simple way to manage library items, authors, and patrons. It enables librarians to add, edit, and delete items and patrons, as well as manage borrowing and returning operations.

This manual includes an overview of the system's features, detailed instructions on how to use each function, and explanations of key components.

Table of Contents

1. **Application Overview**

The **Library Management System** is a console-based Java application that allows librarians to:

- Manage a catalog of books and periodicals.
- Keep track of authors and the items they have written.
- Register patrons, including students and employees, who can borrow items from the library.
- Enable patrons to borrow and return library items, keeping track of available copies.

This system is designed to provide basic library functionality with a structured, object-oriented approach.

2. **How to Start / Access the Application**

1. Open the Terminal/Command Prompt in the src directory of the project.
2. Compile all Java files by running the following command in bash
   javac models/*.java services/*.java Main.java
3. Run the application with the following command in bash:
   java Main
4. The system will load with sample data for demonstration purposes and guide you through the available features via prompts in the console.

3. **Features and Usage**

3.1 Adding Library Items

Library items can be either Books or Periodicals.

Books:

- Attributes: Title, Author, ISBN, Publisher, Number of Copies, and Format Type (Printed, Electronic, or Audio).
- Example: A printed book titled "Harry Potter" by J.K. Rowling with 5 copies available.

Periodicals:

- Attributes: Title, Author, ISBN, Publisher, Number of Copies, and Printed Status (Printed or Electronic).
- Example: A printed periodical titled "National Geographic" with 7 copies available.

Usage:

- In the console, follow the prompts to add a new item by entering its details as requested.
- The application will confirm the addition of the item.

3.2 Managing Authors

The Author class represents the writers of library items. Each author has:

- A Name
- A Date of Birth
- A List of Items they have written.

Usage:

- Add a new author by providing their name and date of birth.
- Associate authors with library items when creating or editing those items.

3.3 Managing Patrons

Patrons are users who can borrow library items. Patrons can be either Students or Employees.

Attributes for All Patrons:

- Name: The patron's name.
- Address: The patron's address.
- Phone Number: The patron's contact phone number.
- Borrowed Items: A list of items currently borrowed by the patron.

Usage:

- Register new patrons by providing their details.
- Patrons are stored in the library system and can be searched by name.

3.4 Borrowing Items

A patron can borrow a library item if copies are available.

Process:

1. Search for an Item: Locate the item by title, author, or ISBN.
2. Check Availability: Confirm that the item has available copies.
3. Borrow: When a patron borrows an item, the number of available copies decreases by one.
4. Confirmation: The system will confirm that the item has been borrowed successfully.

Example: Patron Alice searches for "Harry Potter" and borrows one copy. The system deducts one from the available copies.

3.5 Returning Items

A patron can return a previously borrowed item to the library as follows:

1. Locate Borrowed Item: Confirm the item is in the patron’s borrowed list.
2. Return Item: When the patron returns the item, the system increments the number of available copies.
3. Confirmation: The system will confirm the successful return of the item.

Example: Patron Alice returns "Harry Potter". The system updates the available copies by adding one.

4. Class Descriptions

1. LibraryItem (Abstract Class)

- Represents a general item in the library.
- Attributes: itemID, title, author, ISBN, publisher, numCopies.
- Methods:
    - borrowItem(): Decreases the number of available copies.
    - returnItem(): Increases the number of available copies.

2. Book (Subclass of LibraryItem)

- Represents a book in the library.
- Attributes: Inherits all from LibraryItem and adds formatType (Printed, Electronic, or Audio).
- Methods: Inherits borrowItem() and returnItem() from LibraryItem.

3. Periodical (Subclass of LibraryItem)

- Represents a periodical in the library (e.g., magazine).
- Attributes: Inherits all from LibraryItem and adds isPrinted.
- Methods: Inherits borrowItem() and returnItem() from LibraryItem.

4. Author

- Represents an author who has written items in the library.
- Attributes: name, dateOfBirth, itemsWritten (list of LibraryItems).
- Methods: addItem() to add an item to the author's list of written items.

5. Patron (Abstract Class)

- Represents a library patron.
- Attributes: name, address, phoneNumber, borrowedItems.
- Methods:
    - borrowItem(): Adds an item to the patron’s borrowed items.
    - returnItem(): Removes an item from the patron’s borrowed items.
    - getName(): Returns the patron’s name.

6. Student (Subclass of Patron)

- Represents a student patron.
- Inherits all attributes and methods from Patron.

7. Employee (Subclass of Patron)

- Represents an employee patron.
- Inherits all attributes and methods from Patron.

8. Library

- Manages the collection of items, authors, and patrons.
- Attributes: items (list of LibraryItems), authors (list of Authors), patrons (list of Patrons).
- Methods:
    - addItem(), addPatron(), searchItemByTitle(), getPatronByName().
    - Methods to manage borrowing and returning items.

9. Main / Demo

- The entry point of the application.
- Methods:
    - main(): Loads sample data and demonstrates the library's functionality.

5. Development Documentation

Javadocs:

1. **Authors.java**
2. **Book.java**
3. **LibraryItems.java**
4. **Patron.java**
5. **Periodical.java**
6. **Student.java**
7. **Employee.java**

6. Source Code Directory Structure: The project is organized as follows:

```
LibraryManagementSystem
├── src/
│   ├── models/       # Contains classes representing library models (e.g., Book, Patron)
│   ├── services/     # Contains service classes (e.g., Library)
│   └── Main.java     # Main entry point of the application
```

7. Build Process

1. **Navigate to the src Folder:** cd Library Java Midterm/src

2. **Compile the Code:** javac models/*.java services/*.java Main.java

3. **Run the Program:** java Main

8. Development Standards

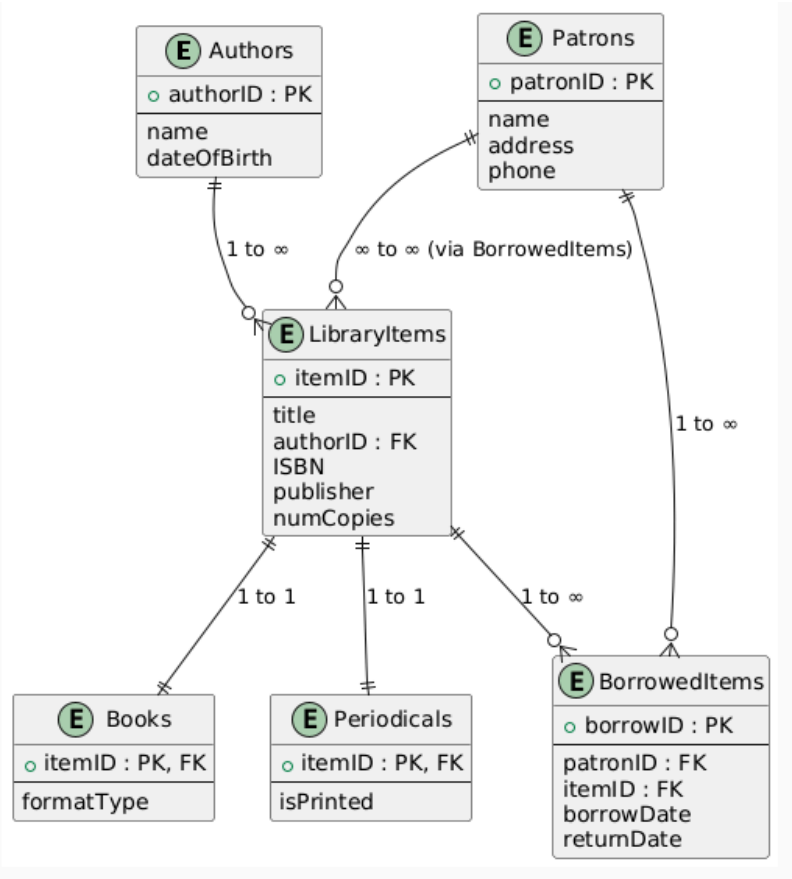
- Naming Conventions: Classes are named in PascalCase, methods in camelCase, and constants in UPPERCASE_SNAKE_CASE.
- Modular Design: Each class is responsible for a single part of the library system to ensure code modularity and reusability.
- Error Handling: Basic error handling in the Library and Patron classes when borrowing or returning items.

9. Database Design

The following tables was used:

1. Authors Table: Contains author information with relationships to library items.
2. LibraryItems Table: Stores information on books and periodicals.
3. Patrons Table: Stores information on library patrons.
4. BorrowedItems Table: Tracks which patrons have borrowed which items.

10. Entity Relationships and Diagram (ERD):



ERD Explanation

The ERD above provides a clear structure for a relational database for the Library Management System. By following this design:

- Data Integrity is maintained by using foreign keys and setting up proper one-to-many and many-to-many relationships.
- Efficiency in querying and managing data is ensured by storing related data in separate tables and linking them through relationships.
- Scalability is achieved by using a modular structure, allowing for easy addition of new features, such as new item types, without restructuring the entire database.

This ERD serves as a foundation for designing a robust, scalable, and efficient library management system database.

Relationship Summary

1. Authors to LibraryItems (1):
    - Each author can write multiple library items, but each item has only one author.
2. LibraryItems to Books and Periodicals (1:1):
    - Each library item is either a book or a periodical. Books and periodicals are specific types of library items with additional attributes.
3. Patrons to BorrowedItems (1):
    - Each patron can have multiple borrowing transactions, but each transaction is associated with one specific patron.
4. LibraryItems to BorrowedItems (1):
    - Each library item can be borrowed multiple times, with each borrowing recorded as a unique transaction in the BorrowedItems table.
5. Many-to-Many Relationship between Patrons and LibraryItems (via BorrowedItems):
    - Patrons can borrow multiple items, and each item can be borrowed by multiple patrons, facilitated by the BorrowedItems join table.

Table: ERD in a table format:

Entity	Authors	LibraryItems	Books	Periodicals	Patrons	BorrowedItems
Description	Stores information about authors who write library items.	General items in the library (books & periodicals).	Represents books in the library, a type of LibraryItem.	Represents periodicals, a type of LibraryItem.	Represents users who can borrow items.	Represents borrowing transactions.
Attributes						
Primary Key	authorID	itemID	itemID	itemID	patronID	borrowID
Attributes						
Attribute 1	authorID	itemID	formatType	isPrinted	patronID	patronID (FK)
Attribute 2	name	title	-	-	name	itemID (FK)
Attribute 3	dateOfBirth	authorID (FK)	-	-	address	borrowDate
Attribute 4	-	ISBN	-	-	phone	returnDate
Attribute 5	-	publisher	-	-	-	-
Attribute 6	-	numCopies	-	-	-	-
Relationships						
Relationship 1	-	Many-to-One with Authors	One-to-One with LibraryItems	One-to-One with LibraryItems	Many-to-Many with LibraryItems via BorrowedItems	Many-to-One with Patrons
Relationship 2	-	One-to-Many with BorrowedItems	-	-	-	Many-to-One with LibraryItems

11. Deployment Documentation

This document serves as the installation manual for the Library Management System Java application. Follow the steps below to set up, compile, and run the application.

Prerequisites

1. Java Development Kit (JDK) 8 or later
    - Download and install the JDK.

Verify the installation by running the following command in your terminal or command prompt: java -version

Ensure that javac is available: javac -version

2. Code Editor
    - Use Visual Studio Code for easy navigation and editing of the Java files.

Installation Steps

Step 1: Set Up the Project Directory Structure

Ensure that the project directory is organized as follows:

LibraryManagementSystem/Library Java Midterm

```
│
├── src/
│   ├── models/
│   │   ├── Author.java
│   │   ├── Book.java
│   │   ├── Employee.java
│   │   ├── LibraryItem.java
│   │   ├── Patron.java
│   │   ├── Periodical.java
│   │   └── Student.java
│   ├── services/
│   │   └── Library.java
│   └── Main.java
```

Step 2: Verify Package Declarations

Ensure that each Java file in the project has the correct package declaration at the top. For example:

- In models folder: The Java files start with package models;
- In services folder: The Library.java file start with package services;
- For Main.java: If it's located in the root of the src folder, it should not have any package declaration.

Step 3: Compile the Project

1. Open a Terminal in the src directory of the project. It can be done by navigating to the Library Java Midterm/src folder.
2. Compile all Java files by running the following command:javac models/*.java services/*.java Main.java
    - This command compiles all the Java files in the models and services directories, as well as Main.java.
    - If the compilation is successful, it will create .class files for each Java file in the same directories.

Step 4: Run the Application

After successful compilation, run the application by executing the following command in the src directory: java Main

This command will start the program, and the output will demonstrate the library management system's functionality, including loading sample data, borrowing, returning items, and displaying information about patrons and items.

12. Summary:

1. Ensure Prerequisites: JDK installed and verified.
2. Download project files.
3. Set Up Folder Structure: Ensure it matches the package declarations.
4. Compile: Run javac models/*.java services/*.java Main.java.
5. Run: Execute java Main to start the application.

This completes the installation and setup for the Library Management System.