Library Management System Documentation

1. User Documentation

1.1 Overview

The **Library Management System** is a Java-based application designed to manage library resources, patrons, and authors efficiently. The system allows users to add, edit, and delete library items, manage authors and patrons, and track item borrowing and returning. The application is command-line-based and demonstrates core Object-Oriented Programming principles like inheritance and encapsulation.

1.2 Application Features

- **Manage Library Items**: Add, edit, delete, and view books or periodicals in the library.
- **Manage Authors**: Add, edit, delete authors, and view their details, including a list of items they've written.
- **Manage Patrons**: Register patrons, record their borrowing and returning activities, and view their details.
- **Borrowing & Returning**: Patrons can borrow and return library items based on availability.

1.3 Class Descriptions

Library

- **Responsibilities**: Manages library items, authors, and patrons. Provides methods to add, edit, delete, and search these objects.
- **Key Methods**:
  - `addLibraryItem()`, `editLibraryItem()`, `deleteLibraryItem()`
  - `addAuthor()`, `editAuthor()`, `deleteAuthor()`
  - `addPatron()`, `borrowLibraryItem()`, `returnLibraryItem()`

LibraryItem

- **Responsibilities**: Represents an item in the library, such as a book or periodical.
- **Attributes**: `id`, `title`, `author`, `ISBN`, `publisher`, `totalCopies`, `availableCopies`, `itemType`
- **Key Methods**: `borrowItem()`, `returnItem()`

Book *(Subclass of LibraryItem)*

- **Additional Attributes**: `author, isElectronic`, `isAudio`
- **Responsibilities**: Represents a book item in the library.

- **Methods**: Inherits methods from `LibraryItem` and may include additional behavior specific to books.

Periodical *(Subclass of LibraryItem)*

- **Additional Attributes**: `issueDate`, `isElectronic`
- **Responsibilities**: Represents a periodical item in the library.
- **Methods**: Inherits methods from `LibraryItem` and may include additional behavior specific to periodicals.

Author

- **Responsibilities**: Represents an author who has written items in the library.
- **Attributes**: `name`, `dateOfBirth`, `itemsWritten`
- **Key Methods**: `addLibraryItem()`, `printDetails()`

Patron

- **Responsibilities**: Represents a library user who can borrow and return items.
- **Attributes**: `name`, `address`, `phoneNumber`, `borrowedItems`
- **Key Methods**: `borrowedItem()`, `returnItem()`, `printDetails()`

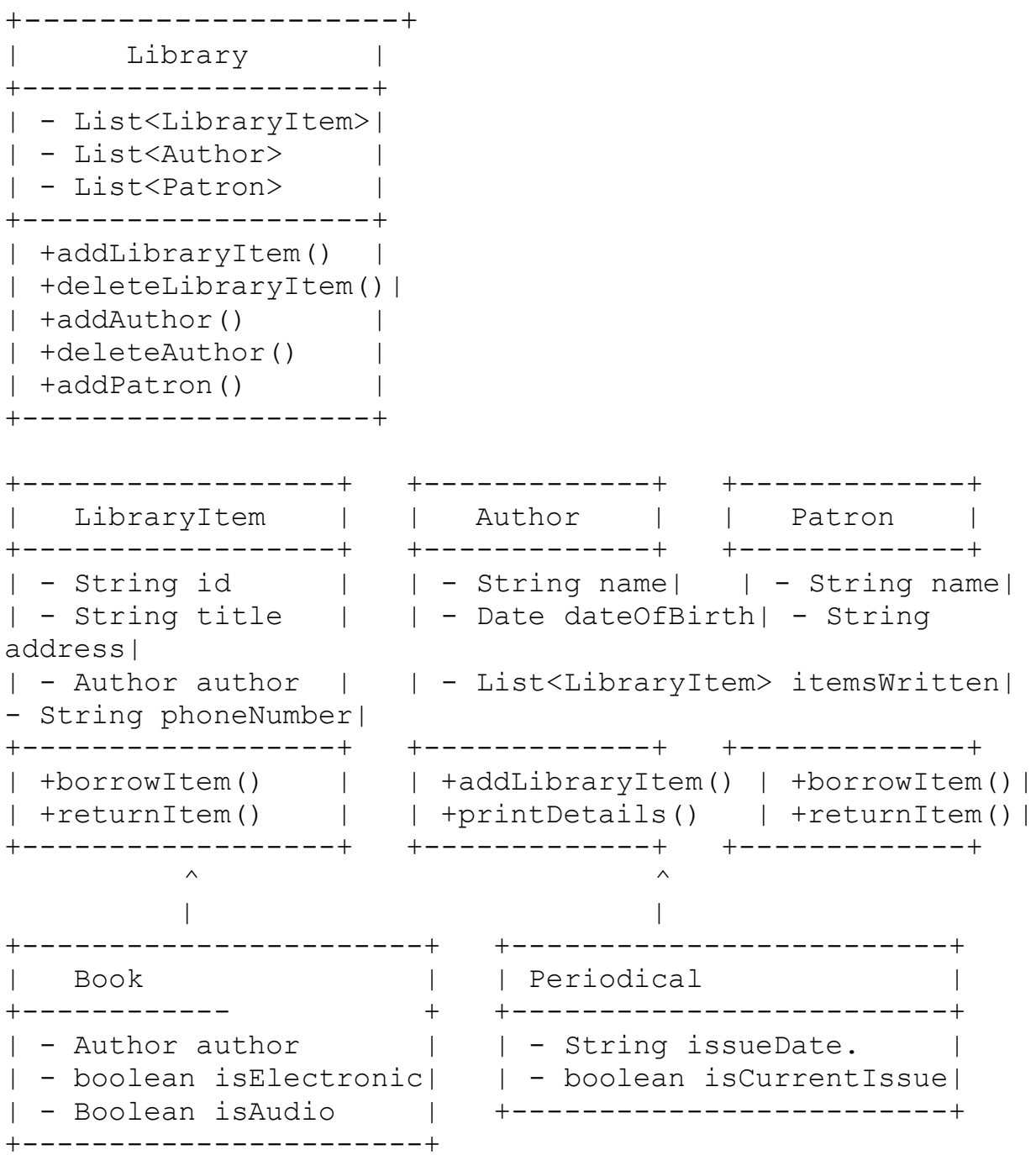
Demo

- **Responsibilities**: Main entry point for the application. Demonstrates the functionality of the library system through a command-line menu for testing purposes.
- **Key Methods**: Contains a main method with a menu-driven interface to access various features of the application.

1.4 How to Access the Application

1. **Prerequisites**: Ensure you have Java installed (JDK 11 or later).
2. **Starting the Application**:
    - Compile and run the `Demo` class.
    - The application will display a menu where you can interact with different features, such as adding authors, patrons, and library items.

1.5 Class Diagram



---

2. Development Documentation

2.1 Javadocs

- Each class is documented with Javadoc comments, including descriptions of attributes and methods. This can be generated using the `javadoc` tool.

2.2 Directory Structure

- **library**: Contains all source code files, divided into different packages if needed.
  - `Library.java`: Manages library items, authors, and patrons.
  - `LibraryItem.java`: Represents items in the library.
  - `Author.java`: Represents authors.
  - `Patron.java`: Represents patrons.
  - `Demo.java`: Main application class.
- **docs**: Contains generated Javadocs.

2.3 Build Process

- **Compiling**:

  ```
  javac Library/*.java
  ```

- **Running**:

  ```
  java Library.Demo
  ```

2.4 Compiler Dependencies

- **Java Development Kit (JDK)**: JDK 11 or later is required to compile and run the project.

2.5 Development Standards

- **Naming Conventions**: Classes are named using CamelCase, while attributes and methods use camelCase.
- **Documentation**: All classes and methods are documented with Javadoc.
- **Error Handling**: Basic error handling included for invalid operations.

2.6 Database Design (Theoretical)

The application does not implement an actual database. Below is a theoretical structure for a relational database that would support this application.

Database Tables:

1. **Authors**: Stores author details.
    - `id` (Primary Key)
    - `name`
    - `dateOfBirth`
2. **LibraryItems**: Stores library item details.
    - `id` (Primary Key)
    - `title`
    - `author_id` (Foreign Key)
    - `ISBN`
    - `publisher`
    - `totalCopies`
    - `availableCopies`
    - `itemType`
3. **Patrons**: Stores patron details.
    - `id` (Primary Key)
    - `name`
    - `address`
    - `phoneNumber`
4. **BorrowedItems**: Tracks which items patrons have borrowed.
    - `id` (Primary Key)
    - `patron_id` (Foreign Key)
    - `item_id` (Foreign Key)
    - `borrowDate`
    - `returnDate`

Entity-Relationship Diagram:

```
Authors (1) ----< (N) LibraryItems
Patrons (1) ----< (N) BorrowedItems >---- (1) LibraryItems
```

2.7 Source Code from GitHub

To clone the repository:

```
git clone https://github.com/emeritus00/Java-MidSprint/tree/main
```

---

3. Deployment Documentation

3.1 Installation Instructions

1. **Prerequisites**: Ensure JDK 11 or later is installed.
2. **Download Source Code**:
    - Clone the repository (as outlined in Section 2.7) or download the source files directly.
3. **Compile the Project**:

    ```
    javac Library/*.java
    ```

4. **Run the Application**:

    ```
    java Demo
    ```

5. **Javadocs**: Generate Javadocs if needed by running:

    ```
    cd Library

    javadoc -d docs *.java
    ```

3.2 Troubleshooting

- **Common Issues**:
    - Ensure all Java files are compiled before running.
    - Verify Java installation by running `java -version`.