System Documentation

For

Library Management System

Document Version: [1.0.0]

Date: [27/02/2024]

Contents

1 Documentation Management

Introduction:

The Library Management System is a Java application designed to manage books, authors, and patrons in a library setting. It allows users to perform operations such as adding and deleting books, authors, and patrons, borrowing and returning books, searching for books by title, authors, or ISBN, and generating a list of overdue books.

1.1 Contributors

Role	Unit	Name
Software Developer		Corina Jewer
Systems Analyst Designer		Corina Jewer
Other document contributors		

1.2 Version Control

Date	Version	Author	Section	Amendment
02/26/2024	1.0.0	Corina Jewer		

2 User Documentation

The Library Management System is a Java application designed to manage books, authors, and patrons in a library setting. It allows users to perform operations such as adding and deleting books, authors, and patrons, borrowing and returning books, searching for books by title, author, or ISBN, and generating a list of overdue books

2.1 Classes and Functionality:

Author: Represents an author who writes books. It stores information such as the author's name, date of birth, and a list of written books.

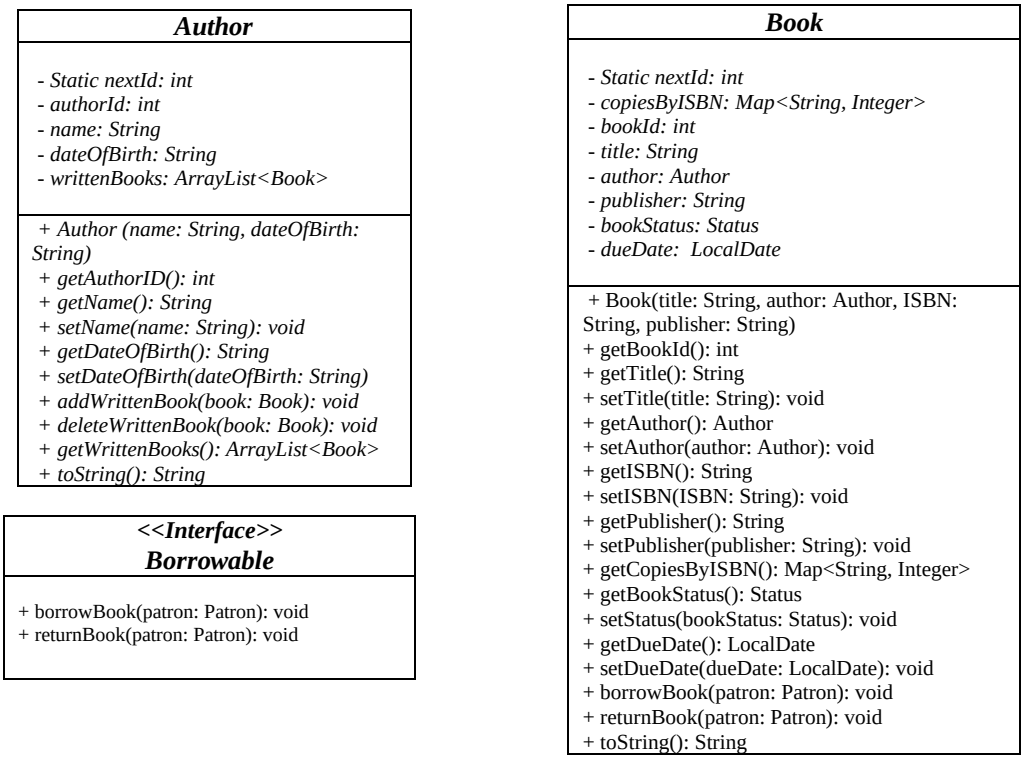
Book: Represents a book in the library. It contains information such as the title, author, ISBN, publisher, status, and due date.

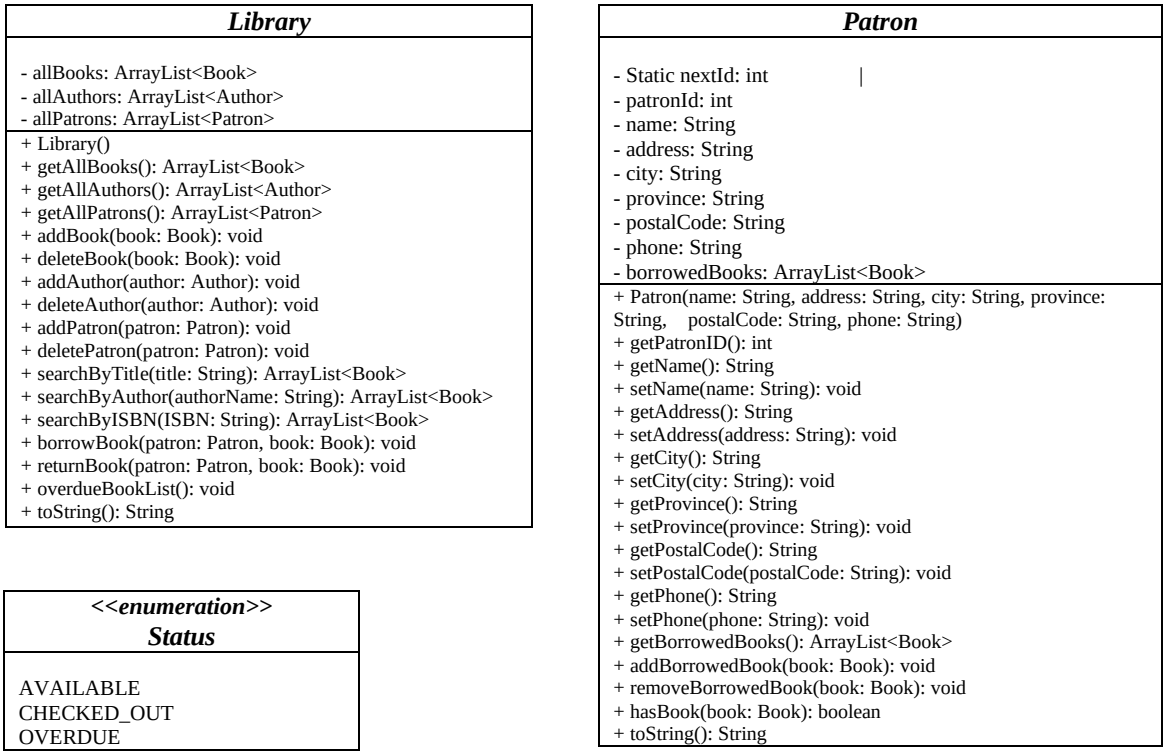
Patron: Represents a library patron. It stores information such as the patron's name, address, city, province, postal code, phone number, and a list of borrowed books.

Library: Manages books, authors, and patrons in the library. It provides methods to add, delete, and search for books, authors, and patrons, as well as to borrow and return books and generate a list of overdue books.

TestLibrary: The command-line interface provides functionality testing for the library class. Users interact with the library system by borrowing and returning books, searching for books by various criteria, as well as adding and deleting patrons, books, and authors.

2.2 UML Diagrams





2.3 How to Start/Access the Application

- Ensure you have Java Development Kit (JDK) Installed. (version "17.0.9" 2023-10-17 or later)
- Install Java Extension Pack: Make sure you have the Java Extension Pack installed in your VS Code. This extension pack includes essential tools for Java development, such as language support, debugging, and code snippets.
- Download Java files to your local disk or clone url link from GitHub repository in GitHub desktop to open in VS code.
- After downloading or cloning the repository, open the Java project in VS Code. You can do this by navigating to the project directory and using the "File" > "Open Folder" option in VS Code or by opening the project directly from GitHub desktop.
- Compile and Run: To compile your Java code, you can use the VS Code integrated terminal. Open the terminal within VS Code by selecting "Terminal" > "New Terminal" from the menu. Then, navigate to the directory containing your Java files and use the javac command to compile them. You can also use the built-in "Run" and "Debug" features of VS Code to compile and execute your Java code menu.
- Run the TestLibrary Class to start the application.

3 Development Documentation

3.1 Javadoc's:

- Javadoc comments are provided for all classes and methods to document their functionality and usage.

3.2 Source Code Structure:

- All Java source files are organized in a single directory.
  Each class is defined in its own .java file.

3.3 Build Process:

- Compile the project using a Java compiler (e.g., javac *.java). You can also use the built-in "Run" and "Debug" features of VS Code to compile and execute the main method of the TestLibrary class .

3.4 Compiler Time Dependencies:

- There are no external dependencies required at compile time.

3.5 Development Standards:

- Followed standard Java coding conventions and best practices for naming, formatting, and structuring code.

3.6 Setting up a Database for Development:

- This application does not utilize a database for development currently. It employs a function named **loadData** to initialize data. The **loadData** function is responsible for populating the application with initial data, such as authors, books, and patrons. This data-loading mechanism is invoked within the **main** method, allowing the application to operate with pre-defined data without the need for a database infrastructure during development.

3.7 Getting the Source Code from the Repository:

- Clone the repository containing the source code using Git (https://github.com/CorinaJewer/Sprint1-Java)

4 Deployment Documentation

Installation Manual:
Ensure Java Runtime Environment (JRE) is installed on the target system.
Copy the compiled .class files to the deployment directory.
Run the TestLibrary class to start the application.

To Demonstrate the Project:
Compile the Java files using a Java compiler.
Run the TestLibrary class to access the Library Management System and perform operations such as borrowing and returning books.