Benjamin Osemene, Shukrullah Zahiri, Andrew Ohwoka.

Keyin college  St. John’s, NL

Java document

Table of Contents

[1.0 User Documentation 2](#_Toc170704622)

[1.1 What is the application about? 2](#_Toc170704623)

[1.2 Explanation of the classes and their work 2](#_Toc170704624)

[1.3 How to start the management system 3](#_Toc170704625)

[1.4 Class diagram and association. 3](#_Toc170704626)

[2.0 Development Documentation 4](#_Toc170704627)

[2.1 Source code Directory Structure 4](#_Toc170704628)

[2.2 Build Process 4](#_Toc170704629)

[2.3 Compiler Time Dependencies 4](#_Toc170704630)

[2.4 Development Standards 5](#_Toc170704631)

[2.5 Database structure 5](#_Toc170704632)

[2.5.1 LibraryMenu.java 5](#_Toc170704633)

[2.5.2 Library.java 5](#_Toc170704634)

[2.5.3 Author.java 5](#_Toc170704635)

[2.5.4 Patron.java 5](#_Toc170704636)

[2.5.5 LibraryItem.java 6](#_Toc170704637)

[2.5.6 Book.java 6](#_Toc170704638)

[2.5.7 Borrow.java 6](#_Toc170704639)

[2.5.8 Periodical.java 6](#_Toc170704640)

[2.5.9 Status.java 6](#_Toc170704641)

[2.5.10 Student.java 6](#_Toc170704642)

[2.5.11 Entity Relationship Diagram (ERD) 7](#_Toc170704643)

[2.5.12 Entity Relationships 7](#_Toc170704644)

[2.6 Getting Source Code 7](#_Toc170704645)

[3.0 Deployment Documentation 8](#_Toc170704646)

[3.1 Installation process 8](#_Toc170704647)

# User Documentation

This part of the document describes the user document of the library management system

## 1.1 What is the application about?

This Java application is designed to serve as a robust and efficient library management system, streamlining various administrative and operational tasks associated with managing a library. The system provides comprehensive functionality for handling library items, patrons, and authors, ensuring a smooth and organized management process.

Key features of the application include the ability to add, edit, and delete library items such as books, magazines, and other media. Users can also manage patron information, allowing for the addition, modification, and removal of library members. This ensures that all records are up-to-date and accurate.

The borrowing and returning of library items are efficiently handled by the system, allowing patrons to check out and return items with ease. This feature is essential for maintaining the circulation of library resources and ensuring that items are available to patrons in a timely manner.

Additionally, the system offers functionality for managing author information, enabling users to add new authors, update existing author details, and remove authors as needed. This helps in maintaining a comprehensive database of all authors associated with the library's collection.

The application boasts an interactive, menu-driven interface that guides users through various operations seamlessly. This user-friendly interface is designed to enhance the overall user experience, making it easy for library staff and patrons to navigate and perform tasks efficiently.

## 1.2 Explanation of the classes and their work

The function of the Management system for library is listed below.

* LibraryMenu.java: The main class that displays the menu to the user and handles user input to perform various library management tasks.
* Library.java: Manages the collection of library items, authors, and patrons. It contains methods to add, edit, delete, borrow, and return items, and to add, edit, and delete authors and patrons.
* Author.java: Represents an author with properties such as name, date of birth, and a list of written items.
* Patron.java: Represents a library patron with properties like name, patron ID, and the list of borrowed items.
* LibraryItem.java: The LibraryItem class is a base class for all items in the library. This class can be extended by specific item types such as books and periodicals.
* Book.java: The Book class represents a book item within the library. This class typically extends from a base class like Library Item.
* Borrow.java: The Borrow class manages the borrowing transactions within the library. It records the details of items borrowed by patrons.
* Employee.java: The Employee class represents library employees who manage various operations within the library.
* Periodical.java: The Periodical class represents periodicals such as magazines and journals within the library. This class extends LibraryItem and includes additional attributes relevant to periodicals.
* Status.java: The Status class is an enumeration that defines the various statuses a library item can have.
* Student.java: The Student class represents a student who is a patron of the library. This class typically includes attributes and methods specific to student patrons.

## 1.3 How to start the management system

To start the application, run the [LibraryMenu.java](vscode-file://vscode-app/Applications/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) class. This will display the interactive menu in the console, where you can choose different operations by entering the corresponding number.

## 1.4 Class diagram and association.

A diagram of a library

Description automatically generated

# 2.0 Development Documentation

The javadocs are essiential for providing clear and structured documentation for the java classes and methods. All classes, methods and attributes documented using Javadoc. This can be seen in the java codes.

## 2.1 Source code Directory Structure

The source code directory structure for the java project is shown below.

* **src/**: Contains all the Java source files (.java).
  + **main/java/com/library**: Contains the application source code.
    - **LibraryMenu.java**
    - **Library.java**
    - **Author.java**
    - **Patron.java**
    - **LibraryItem.java**
    - **Book.java**
    - **Borrow.java**
    - **Periodical.java**
    - **Status.java**
    - **Student.java**
  + **test/java/com/library**: Contains the unit tests for the application.
  + **build/**
  + **lib/**
  + **docs/**
  + **REAME.md**

## 2.2 Build Process

To compile the project, navigate to the root directory of the project in your terminal and run:

javac src/main/java/\*.java -d out/

This compiles all .java files in the src/main/java directory and outputs the class files to the out/ directory.

## 2.3 Compiler Time Dependencies

In order to compile and run the library management system java application, there is need to have the JDK version 8 or a later version installed on the user system. Here is what need to be done to have JDK in the user system:

* **Check JDK Installation**: Ensure that the JDK is installed on your system. User can check this by running the command javac -version in the terminal or command prompt. It should return the version of the JDK installed.
* **Install JDK**: If the JDK is not installed or user have a version lower than 8, user need to download and install the JDK version 8 or above from the [Oracle JDK Downloads](https://www.oracle.com/java/technologies/javase-downloads.html) page or [OpenJDK](https://openjdk.java.net/).
* **Set Up Environment Variables**: After installing the JDK, user might need to set up the environment variables JAVA\_HOME and add the bin directory of the JDK to your system's PATH variable to make the javac and java commands available in your command line.

## 2.4 Development Standards

* Coding Standards: Follow Java naming conventions, use meaningful variable names, and ensure code readability.
* Version Control: Use Git for version control and maintain a clean commit history.
* Testing: Write unit tests for all classes and methods using a testing framework like JUnit.
* Use Javadoc comments for documentation.

## 2.5 Database structure

Here is a theoretical database design including tables for the various classes in your library management system:

### 2.5.1 LibraryMenu.java

The LibraryMenu class is primarily for the user interface and doesn't directly translate to a database table.

### 2.5.2 Library.java

The Library class itself doesn't directly map to a table but is used to manage the collections of other tables.

### 2.5.3 Author.java

**Table: Authors**

* author\_id (Primary Key): A unique identifier for each author.
* name: The name of the author.
* date\_of\_birth: Date of birth of the author
* written\_items: Written items

### 2.5.4 Patron.java

**Table: Patrons**

* patron\_id (Primary Key): A unique identifier for each patron.
* name: The name of the patron.
* address: The address of patron
* phone\_number: The phone number of patron
* borrow\_items: The list of borrow items

### 2.5.5 LibraryItem.java

**Table: LibraryItems**

* item\_id (Primary Key): A unique identifier for each library item.
* title: The title of the item.
* author: The author
* isbn: The ISBN for the item
* publisher: The publisher of the item
* number\_of\_copies: The number of copies for the items
* status: The current status of the item (e.g., available, checked\_out, overdue).

### 2.5.6 Book.java

**Table: Books**

* book\_id (Primary Key): A unique identifier for each book.
* library\_item\_id (Foreign Key): References LibraryItems.item\_id.
* author\_id (Foreign Key): References Authors.author\_id.

### 2.5.7 Borrow.java

**Table: Borrowings**

* borrow\_id (Primary Key): A unique identifier for each borrowing transaction.
* Library\_item\_id (Foreign Key): References LibraryItems.item\_id.
* patron\_id (Foreign Key): References Patrons.patron\_id.
* borrow\_status: The status of the item was borrowed.

### 2.5.8 Periodical.java

**Table: Periodicals**

* periodical\_id (Primary Key): A unique identifier for each periodical.
* library\_item\_id (Foreign Key): References LibraryItems.item\_id.
* Periodical\_type: The periodical type (Printed, electronic)

### 2.5.9 Status.java

**Table: Statuses**

* status\_id (Primary Key): A unique identifier for each status.
* status\_name: The name of the status (e.g., available, checked\_out, overdue).

### 2.5.10 Student.java

**Table: Students**

* student\_id (Primary Key): A unique identifier for each student.
* patron\_id (Foreign Key): References Patrons.patron\_id.
* name: The name of student.
* address: The student address.
* Phone\_number: The student phone number

### 2.5.11 Entity Relationship Diagram (ERD)

This ERD illustrates the relationships between the tables:

* LibraryItems is a central table that is referenced by Books and Periodicals.
* Authors is related to Books through author\_id.
* Patrons is referenced by both Borrowings and Students.
* Statuses is linked to LibraryItems to track the status of each item.
* Borrowings links Patrons and LibraryItems to manage borrowing transactions.

### 2.5.12 Entity Relationships

* One Author can author many Books.
* One Book or Periodical belongs to one LibraryItem.
* One Patron can borrow many LibraryItems.
* Each LibraryItem can have one Status.
* One Patron can be associated with one Student.

## 2.6 Getting Source Code

By going to this link the user can obtain the source code.

https://github.com/AndrewOhwoka/SprintOneJava

# 3.0 Deployment Documentation

This document serves as the installation manual for the Java library management system application, detailing the steps necessary to install and run the application.

## 3.1 Installation process

This is the installation processs.

* Ensure Java Development Kit (JDK) is installed on your system.
* Download the source code from the GitHub repository.
* Compile the project using the instructions in the Development Documentation.
* Run the application by running the library Menu.