Group: Evan O’Bryan, Maryam Shaheen, Jack Coady

**Library Documentation:**

**User Documentation:**

This application is a Java representation for a Library interface, capable of adding, editing, and deleting data on the item in the library (i.e. the books), along with the author as well as the patrons who the books are being lent out to.

Here is an explanation of each class:

1. LibraryItem.java, or the LibraryItem class is used to set as well as get all the relevant information on the items of the library, the books. This includes the item ID, title of the book, author, ISBN (or the International Standard Book Number), the publisher of the book, as well as the number of copies of the book the library has.
2. Author.java, or the Author class is used to set and get the name of the author, their date of birth, as well as a list of items they have written. It also allows for the librarians to add new authors or to edit or delete existing authors.
3. Patron.java, or the Patron class allows librarians to add new patrons, or to edit or delete existing patrons from the library database. It also gets and sets the name, address, phone number, and a list of library items they have borrowed. The patron can also be either a librarian or a student.
4. Library.java, or the Library class allows patrons to borrow from the library or return items they have already borrowed. With this, they are also able to search for items by the title, name of the author, or by the ISBN and can borrow a specific number of copies, provided that there is a sufficient number of that item to be borrowed. If not, the system displays an error message stating that the item is currently being borrowed. This can also be said for the items they have borrowed, where they are able to search by the same information and are able to return a specific number of them.
5. main.java or the main class is where we start off, allowing the user to initiate the menu to be able to navigate the system. It displays a menu with 13 options. To add, edit, or delete a library item, add, edit or delete an author, add, edit or delete a patron, to borrow, return or to search an item, as well as an option to end the system, exiting the menu.

We start by running the main.java file acquired from the GitHub repository, and then follow the menu as well as its instructions to navigate to what you need.

**Class Diagram:**

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|                    Library                         |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| - items: List<LibraryItem>                         |

| - authors: List<Author>                             |

| - patrons: List<Patron>                             |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| + addItem(item: LibraryItem): void                 |

| + removeItem(item: LibraryItem): void               |

| + findItemById(itemId: String): LibraryItem         |

| + searchItemsByTitle(title: String): List<LibraryItem> |

| + searchItemsByAuthor(authorName: String): List<LibraryItem> |

| + searchItemByIsbn(isbn: String): LibraryItem       |

| + addAuthor(author: Author): void                   |

| + removeAuthor(author: Author): void               |

| + findAuthorByName(name: String): Author           |

| + addPatron(patron: Patron): void                   |

| + removePatron(patron: Patron): void               |

| + findPatronByName(name: String): Patron           |

| + borrowItem(patronName: String, itemId: String): void |

| + returnItem(patronName: String, itemId: String): void |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|       LibraryItem         |       |      Author       |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|     |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| - itemId: String         |     | - name: String     |

| - title: String           |     | - dateOfBirth: String |

| - author: Author         |     | - itemsAuthored: List<LibraryItem> |

| - isbn: String           |     |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

| - publisher: String     |

| - numCopies: int         |        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|       |       Patron       |

| + getters/setters       |       |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|       | - name: String     |

                              | - address: String |

                              | - phoneNumber: String |

                              |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

**Associations:**

* Library to LibraryItem. Composition (1 to many). Library manages multiple instances of LibraryItem.
* Library to Author. Composition (1 to many). Same as above but for Author.
* Library to Patron. Composition (1 to many). Same as above but for Patron.
* LibraryItem to Author. Aggregation (1 to 1). Each LibraryItem is associated with one Author.
* Author to LibraryItem. Aggregation (1 to many). Each Author can be associated with multiple LibraryItem instances.
* Patron to LibraryItem. Aggregation (1 to many). Each Patron can borrow multiple LibraryItem instances.

**Development Documentation:**

Javadocs included with the code for easy understanding.

**Source code directory structure:**

library-management-system/

|-- src/ (Contains all the source code for the application)

|   |-- main/ (The main directory for the source code)

|   |   |-- java/ (The directory containing Java source files)

|   |   |   |-- LibraryItem.java (Class representing a library item)

|   |   |   |-- Author.java (Class representing an author)

|   |   |   |-- Patron.java (Class representing a patron)

|   |   |   |-- Library.java (Class representing the library, which manages library items, authors and patrons)

|   |   |   |-- main.java (The main class containing the entry point of the application and the menu system)

|-- build/ (The directory where compiled classes and build artifacts will be stored)

|   |-- classes/ (Directory for compiled ‘.class’ files)

|   |-- libs/ (Directory for any external libraries)

|-- lib/ (Directory for external libraries if applicable)

|-- docs/ (Directory for documentation)

|-- README.md (A markdown file with info about the project)

|-- build.sh (Shell script to compile the project)

|-- run.sh (Shell script to run the project)

To get the source code from a GitHub repository, you can follow these steps:

Step-by-Step Guide to Clone a GitHub Repository

1. Install Git

Make sure you have Git installed on your machine. You can download it from [git-scm.com](https://git-scm.com/downloads) and follow the installation instructions for your operating system.

2. Clone the Repository

Open your terminal (or command prompt) and navigate to the directory where you want to clone the repository. Use the `git clone` command followed by the URL of the repository. Here’s an example command:

```sh

git clone https://github.com/your-username/library-management-system.git

```

Replace `https://github.com/your-username/library-management-system.git` with the actual URL of the GitHub repository you want to clone.

3. Navigate to the Project Directory

Once the repository is cloned, navigate into the project directory:

```sh

cd library-management-system

```

**Build Process:**

1. Compile the source code.  
   Use the javac command. Provided is a build.sh to automate this.

#!/bin/bash

# Create the build directory if it doesn't exist

mkdir -p build/classes

# Compile the Java source files

javac -d build/classes src/main/java/\*.java

echo "Build complete."

Then run ./build.sh in the terminal to compile the project.

1. Run the application.

#!/bin/bash

# Run the Main class

java -cp build/classes Main

echo "Execution complete."

Then execute the following in the terminal: ./run.sh

**Compiler Time Dependencies:**

For compiler time dependencies, this project relies on only having JDK, Java Development Kit.

**Development Standards:**

Used meaningful variable and method names, properly commented the code to ensure easy understanding. Provided Javadocs and provides error messages to allow users to understand errors.

**Theoretical Database:**

Here is a theoretical design for the database of the Library Management System. This design includes entities and their relationships, represented in an Entity-Relationship diagram.

**Entities and Attributes:**

1. LibraryItem

   - `item\_id` (Primary Key)

   - `title`

   - `isbn`

   - `publisher`

   - `num\_copies`

   - `item\_type` (e.g., Book, Periodical)

   - `format` (e.g., Printed, Electronic, Audio)

2. Author

   - `author\_id` (Primary Key)

   - `name`

   - `date\_of\_birth`

3. Patron

   - `patron\_id` (Primary Key)

   - `name`

   - `address`

   - `phone\_number`

   - `patron\_type` (e.g., Student, Employee)

4. BorrowedItem

   - `borrow\_id` (Primary Key)

   - `patron\_id` (Foreign Key referencing Patron)

   - `item\_id` (Foreign Key referencing LibraryItem)

   - `borrow\_date`

   - `return\_date`

5. WrittenBy

   - `author\_id` (Foreign Key referencing Author)

   - `item\_id` (Foreign Key referencing LibraryItem)

**Entity-Relationship Diagram:**

Here’s an overview of the relationships between the entities:

1. LibraryItem - Author (Many-to-Many)

   - A library item can be written by multiple authors.

   - An author can write multiple library items.

   - This relationship is represented by the `WrittenBy` entity.

2. Patron - LibraryItem (Many-to-Many)

   - A patron can borrow multiple library items.

   - A library item can be borrowed by multiple patrons.

   - This relationship is represented by the `BorrowedItem` entity.

**ER Diagram Representation:**

```

Author

--------------

| author\_id    |---\

| name         |     \

| date\_of\_birth|     |

----------------     \

                      |

                     WrittenBy

Patron               ----------

--------------       | author\_id  |

| patron\_id   |     | item\_id    |

| name        |       -------------

| address     |

| phone\_number|

| patron\_type |

--------------

LibraryItem

--------------

| item\_id     |---/

| title       |

| isbn        |

| publisher   |

| num\_copies  |

| item\_type   |

| format      |

--------------

BorrowedItem

----------------

| borrow\_id  |

| patron\_id  |

| item\_id    |

| borrow\_date|

| return\_date|

----------------

```

**Table Descriptions:**

1. LibraryItem Table

   - Stores information about all library items.

   - `item\_id` is the primary key.

2. Author Table

   - Stores information about authors.

   - `author\_id` is the primary key.

3. Patron Table

   - Stores information about patrons.

   - `patron\_id` is the primary key.

4. BorrowedItem Table

   - Stores information about the borrowing activity of patrons.

   - `borrow\_id` is the primary key.

   - `patron\_id` references the `patron\_id` in the Patron table.

   - `item\_id` references the `item\_id` in the LibraryItem table.

5. WrittenBy Table

   - Stores information about which authors wrote which library items.

   - `author\_id` references the `author\_id` in the Author table.

   - `item\_id` references the `item\_id` in the LibraryItem table.

**Deployment Documentation:**

**Installation Manual for Library Management System**

**Prerequisites:**

* Java Development Kit (JDK) installed on your machine (version 8 or higher).

Steps to Install and Run the Application

1. Download the Source Code:

   - Clone the repository or download the source code as a ZIP file and extract it to a directory of your choice.

   ```sh

   git clone https://github.com/your-username/library-management-system.git

   cd library-management-system

   ```

2. Set Up the Directory Structure:

   - Ensure the directory structure is as follows:

   ```

   library-management-system/

   |-- src/

   |   |-- main/

   |   |   |-- java/

   |   |   |   |-- LibraryItem.java

   |   |   |   |-- Author.java

   |   |   |   |-- Patron.java

   |   |   |   |-- Library.java

   |   |   |   |-- Main.java

   |-- build/

   |   |-- classes/

   |   |-- libs/

   |-- lib/

   |-- docs/

   |-- README.md

   |-- build.sh

   |-- run.sh

   ```

3. Build the Project:

   - Ensure you have the `build.sh` script in the root directory. This script will compile the Java source files.

   ```sh

   ./build.sh

   ```

   - The script will create the `build/classes` directory and place the compiled `.class` files there.

4. Run the Application:

   - Ensure you have the `run.sh` script in the root directory. This script will run the main application.

   ```sh

   ./run.sh

   ```

5. Using the Application:

   - After running the application, you will see a menu system in the console that allows you to perform various library management operations such as adding, editing, deleting items, authors, and patrons, as well as borrowing and returning items.

**Troubleshooting:**

- Java Not Recognized: Ensure that Java is installed and properly added to your system's PATH.

- Permissions: If you encounter permission issues while running the scripts, you may need to make them executable:

   ```sh

   chmod +x build.sh

   chmod +x run.sh

   ```

- Compilation Errors: Ensure that all Java files are in the correct directory (`src/main/java/`) and that you are running the `build.sh` script from the root directory of the project.

**Additional Information:**

- Javadocs: Generated documentation for the classes is available in the `docs/` directory.

- Source Code: Source code for the project is located in the `src/main/java/` directory.

By following these steps, you should be able to successfully install and run the Library Management System on your machine. If you encounter any issues or need further assistance, please refer to the project's documentation or contact the project maintainer.