User Documentation:

This is a Java-based application that is made to manage library items as well as managing patrons of the library and authors of the items available. It lets users add and remove items, while also letting them track and change the status of the library’s inventory.

Classes:

LibraryItem – Represents the items in the library, covering books and periodicals.

Attributes include:

id: An identifier for an item.

title: An item’s title.

author: The item’s author.

isbn: The ISBN number of a book.

publisher: Publisher of the item.

copiesAvailable: The number of copies of an item in stock.

itemType: Type of item (either "Book" or "Periodical").

format: Format of the item (e.g., "printed", "electronic").

status: Current status of the item (e.g., AVAILABLE, CHECKED\_OUT, OVERDUE).

Also includes getters and setters for each attribute and a method to set the item status.

Library – Manages LibraryItem, Author and Patron objects.

Attributes include:

items: The list of library items.

authors: The list of authors.

patrons: The list of patrons.

Methods include:

addItem(LibraryItem): Adds an item to the library.

removeItem(LibraryItem item): Removes an item from the library.

getItems(): Retrieves all items from the library.

addAuthor(Author): Adds an author to the library.

removeAuthor(Author): Removes an author that was added to the library.

addPatron(Patron): Adds a patron to the library.

findItemByTitle(String item): Finds an item by its title.

3. Author – The data for an author and the works they’re associated with.

Attributes include:

name: The name of the author.

dateOfBirth: The author’s birthdate.

writtenItems: A list of the titles written by the author.

Methods include:

addWrittenItem(String item): Adds an item to the titles written by an author.

Getters are included as well.

4. Patron - Represents a library patron who can borrow items.

Attributes include:

name: The name of the patron.

address: The address of the patron.

phoneNumber: The patron’s phone number.

borrowedItems: A list of items currently borrowed by the patron.

Methods include:

borrowItem(LibraryItem): Allows a patron to borrow an item.

returnItem(LibraryItem): Allows a patron to return an item.

Getters for the attributes are included as well.

5. Status (Enum) - Represents the status of a library item from the listed values.

The values are:

AVAILABLE

CHECKED\_OUT

OVERDUE

6. LibrarySystemDemo – A demo file that serves as the main method to show how the library management system works.

Functionality:

Shows off the built-in features like book returning and borrowing, adding books for the system, displaying the stock, searching for items, etc.

How to Start/Access the Application

Run the LibrarySystemDemo class to run the application. Make sure that you have the ability to run .java files and that all of your files are together in the folder.

Class Diagram:

+----------------+ +--------------+

| LibraryItem |<>-------| Author |

| (LibraryItem.java) | | (Author.java) |

+----------------+ +--------------+

| - id | | - name |

| - title | | - dateOfBirth|

| - author | | - writtenItems|

| - isbn | +--------------+

| - publisher |

| - copiesAvailable|

| - itemType |

| - format |

| - status |

+----------------+

^ ^

| |

| |

| |

+---------------------------+

|

|

+---------------+

| Library |

| (Library.java)|

+---------------+

| - items |

| - authors |

| - patrons |

+---------------+

|

|

|

+---------------+

| Patron |

| (Patron.java) |

+---------------+

| - name |

| - address |

| - phoneNumber |

| - borrowedItems|

+---------------+

+---------------+

| Status |

| (Status.java) |

+---------------+

| - AVAILABLE |

| - CHECKED\_OUT |

| - OVERDUE |

+---------------+

+------------------------+

| LibrarySystemDemo |

| (LibrarySystemDemo.java)|

+------------------------+

| - main(String[] args) |

+------------------------+

Development Documentation:

Code Javadocs:

LibraryItem:

/\*\*

\* Class representing a Library Item.

\*/

class LibraryItem {

private String id;

private String title;

private String author;

private String isbn;

private String publisher;

private int copiesAvailable;

private String itemType; // Either "Book" or "Periodical"

private String format; // "printed", "electronic", etc.

private Status status;

/\*\*

\* Constructor for creating a new LibraryItem.

\*

\* @param id the unique identifier for the item

\* @param title the title of the item

\* @param author the author of the item

\* @param isbn the ISBN of the item

\* @param publisher the publisher of the item

\* @param copiesAvailable the number of copies available

\* @param itemType the type of item (Book or Periodical)

\* @param format the format of the item (printed, electronic, etc.)

\*/

public LibraryItem(String id, String title, String author, String isbn, String publisher, int copiesAvailable, String itemType, String format) {

// constructor implementation

}

// Getters, and also setters

/\*\*

\* Gets the unique identifier of the item.

\*

\* @return the item ID

\*/

public String getId() {

// method implementation

}

/\*\*

\* Gets the title of the item.

\*

\* @return the title of the item

\*/

public String getTitle() {

// method implementation

}

/\*\*

\* Gets the author of the item.

\*

\* @return the author of the item

\*/

public String getAuthor() {

// method implementation

}

/\*\*

\* Gets the ISBN of the item.

\*

\* @return the ISBN of the item

\*/

public String getIsbn() {

// method implementation

}

/\*\*

\* Gets the publisher of the item.

\*

\* @return the publisher of the item

\*/

public String getPublisher() {

// method implementation

}

/\*\*

\* Gets the number of copies available for the item.

\*

\* @return the number of copies available

\*/

public int getCopiesAvailable() {

// method implementation

}

/\*\*

\* Sets the number of copies available for the item.

\*

\* @param copiesAvailable the new number of copies available

\*/

public void setCopiesAvailable(int copiesAvailable) {

// method implementation

}

/\*\*

\* Gets the type of the item.

\*

\* @return the item type

\*/

public String getItemType() {

// method implementation

}

/\*\*

\* Gets the format of the item.

\*

\* @return the item format

\*/

public String getFormat() {

// method implementation

}

/\*\*

\* Gets the current status of the item.

\*

\* @return the status of the item

\*/

public Status getStatus() {

// method implementation

}

/\*\*

\* Sets the status of the item.

\*

\* @param status the new status of the item

\*/

public void setStatus(Status status) {

// method implementation

}

}

Library:

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Class for the Library.

\*/

class Library {

private List<LibraryItem> items;

private List<Author> authors;

private List<Patron> patrons;

/\*\*

\* The constructor initializes the library with empty lists for items, authors, and patrons.

\*/

public Library() {

// constructor implementation

}

/\*\*

\* Method for adding a library item.

\*

\* @param item the LibraryItem to add

\*/

public void addItem(LibraryItem item) {

// method implementation

}

/\*\*

\* The method to remove a library item.

\*

\* @param item the LibraryItem to remove

\*/

public void removeItem(LibraryItem item) {

// method implementation

}

/\*\*

\* The method to get all library items.

\*

\* @return a list of all LibraryItems

\*/

public List<LibraryItem> getItems() {

// method implementation

}

/\*\*

\* The method for adding an author.

\*

\* @param author the Author to add

\*/

public void addAuthor(Author author) {

// method implementation

}

/\*\*

\* For removing an author.

\*

\* @param author the Author to remove

\*/

public void removeAuthor(Author author) {

// method implementation

}

/\*\*

\* Method to add a patron.

\*

\* @param patron the Patron to add

\*/

public void addPatron(Patron patron) {

// method implementation

}

/\*\*

\* Method to find an item by title.

\*

\* @param title the title of the item to find

\* @return the LibraryItem found, or null if not found

\*/

public LibraryItem findItemByTitle(String title) {

// method implementation

}

}

Author:

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Class for an Author.

\*/

class Author {

private String name;

private String dateOfBirth;

private List<String> writtenItems;

/\*\*

\* Constructor for creating a new Author.

\*

\* @param name the name of the author

\* @param dateOfBirth the date of birth of the author

\*/

public Author(String name, String dateOfBirth) {

// constructor implementation

}

/\*\*

\* The method for adding a written item.

\*

\* @param item the title of the item written by the author

\*/

public void addWrittenItem(String item) {

// method implementation

}

/\*\*

\* Gets the name of the author.

\*

\* @return the name of the author

\*/

public String getName() {

// method implementation

}

/\*\*

\* Gets the date of birth of the author.

\*

\* @return the date of birth of the author

\*/

public String getDateOfBirth() {

// method implementation

}

/\*\*

\* Gets the list of items written by the author.

\*

\* @return a list of written item titles

\*/

public List<String> getWrittenItems() {

// method implementation

}

}

Patron:

import java.util.ArrayList;

import java.util.List;

/\*\*

\* A patron of the library.

\*/

class Patron {

private String name;

private String address;

private String phoneNumber;

private List<LibraryItem> borrowedItems;

/\*\*

\* The constructor for creating a new Patron.

\*

\* @param name the name of the patron

\* @param address the address of the patron

\* @param phoneNumber the phone number of the patron

\*/

public Patron(String name, String address, String phoneNumber) {

// constructor implementation

}

/\*\*

\* Method for borrowing an item.

\*

\* @param item the LibraryItem to borrow

\*/

public void borrowItem(LibraryItem item) {

// method implementation

}

/\*\*

\* Method to return an item.

\*

\* @param item the LibraryItem to return

\*/

public void returnItem(LibraryItem item) {

// method implementation

}

/\*\*

\* Gets the name of the patron.

\*

\* @return the name of the patron

\*/

public String getName() {

// method implementation

}

/\*\*

\* Gets the address of the patron.

\*

\* @return the address of the patron

\*/

public String getAddress() {

// method implementation

}

/\*\*

\* Gets the phone number of the patron.

\*

\* @return the phone number of the patron

\*/

public String getPhoneNumber() {

// method implementation

}

/\*\*

\* Gets the list of borrowed items by the patron.

\*

\* @return a list of LibraryItems borrowed by the patron

\*/

public List<LibraryItem> getBorrowedItems() {

// method implementation

}

}

Status:

/\*\*

\* The statuses of a book.

\*/

public enum Status {

AVAILABLE,

CHECKED\_OUT,

OVERDUE

}

LibrarySystemDemo:

import java.util.Scanner;

/\*\*

\* Main class for demonstrating the functionality of the Library System.

\* This class simulates a simple library management system where

\* authors, library items, and patrons can be added, and various

\* operations such as borrowing and returning items can be performed.

\*/

public class LibrarySystemDemo {

/\*\*

\* The entry point of the Library System demo application.

\*

\* @param args command line arguments (not used)

\*/

public static void main(String[] args) {

Library library = new Library(); // Create a new library instance

Scanner scanner = new Scanner(System.in); // Create a scanner for user input

// Adding Authors

Author author1 = new Author("J.K. Rowling", "1965-07-31");

library.addAuthor(author1); // Add first author

Author author2 = new Author("George Orwell", "1903-06-25");

library.addAuthor(author2); // Add second author

// Adding example books for the system

LibraryItem book1 = new LibraryItem("1", "Harry Potter and the Sorcerer's Stone", "J.K. Rowling", "978-0439708180", "Scholastic", 5, "Book", "printed");

library.addItem(book1); // Add first book to the library

author1.addWrittenItem(book1.getTitle()); // Link the book to its author

LibraryItem book2 = new LibraryItem("2", "1984", "George Orwell", "978-0451524935", "Harcourt Brace", 3, "Book", "electronic");

library.addItem(book2); // Add second book to the library

author2.addWrittenItem(book2.getTitle()); // Link the book to its author

LibraryItem periodical1 = new LibraryItem("3", "National Geographic", "Various", "0027-9358", "National Geographic Partners", 10, "Periodical", "printed");

library.addItem(periodical1); // Add a periodical to the library

// Adding a patron for the example

Patron patron1 = new Patron("Alice", "10 Example Lane", "333-2210");

library.addPatron(patron1); // Add patron to the library

// Display the stock of items

System.out.println("Initial Status of Items:");

for (LibraryItem item : library.getItems()) {

System.out.println(item.getTitle() + " - Status: " + item.getStatus());

}

// Search for an item

System.out.println("Searching for '1984':");

LibraryItem searchedItem = library.findItemByTitle("1984");

if (searchedItem != null) {

System.out.println("Found: " + searchedItem.getTitle() + " - Copies Available: " + searchedItem.getCopiesAvailable());

} else {

System.out.println("Item not found.");

}

// Borrowing a book

patron1.borrowItem(book2); // In this case, 1984

System.out.println("After borrowing:");

System.out.println(book2.getTitle() + " - Status: " + book2.getStatus());

// Returning a book

patron1.returnItem(book2); // Returning 1984

System.out.println("After returning:");

System.out.println(book2.getTitle() + " - Status: " + book2.getStatus());

// Editing the information of an item

book1.setCopiesAvailable(10); // Editing number of copies for book 1

System.out.println("Updated Copies of " + book1.getTitle() + ": " + book1.getCopiesAvailable());

// Removing an author

library.removeAuthor(author2);

System.out.println("Author " + author2.getName() + " removed.");

// Removing an item

library.removeItem(book1);

System.out.println("Item " + book1.getTitle() + " removed from the library.");

scanner.close(); // Close the scanner to free resources

}

}

Source Code Structure:

/midtermsprint3

│

├── src

│ ├── Library.java

│ ├── LibraryItem.java

│ ├── Author.java

│ ├── Patron.java

│ ├── Status.java

│ └── LibrarySystemDemo.java

│

└── Document

└── readme

The Build Process:

Compile the project by going to the src directory and running the command:

Javac \*.java

This will compile all of the Java files inside of the directory. From here, you can run the demo, LibrarySystemDemo.

Compiler Dependencies:

You need Java Development Kit support in some form to compile the code. Use a version with support for Java SE 8 or higher.

Development Standards:

Code should be easy to understand, and comments should be added to important code that states what it does. Features that are added should be tested to ensure that the code does not break.

A hypothetical database for the application would be something like the following:

Authors

author\_id (Primary Key)

name

date\_of\_birth

LibraryItems

item\_id (Primary Key)

title

author\_id (Foreign Key)

isbn

publisher

copies\_available

item\_type (ENUM: 'Book', 'Periodical')

format (ENUM: 'printed', 'electronic')

status (ENUM: 'AVAILABLE', 'CHECKED\_OUT', 'OVERDUE')

Patrons

patron\_id (Primary Key)

name

address

phone\_number

Exchanges

exchange\_id (Primary Key)

patron\_id (Foreign Key)

item\_id (Foreign Key)

borrow\_date

return\_date

Entity Relationships:

Authors can have multiple items in LibraryItems (One-to-Many).

Items in LibraryItems can be borrowed by multiple Patrons through Exchanges (Many-to-Many).

Patrons can borrow multiple items in LibraryItems (One-to-Many through Exchanges).

How to get the source code from Github:

Clone the repository by using the command:

git clone <https://github.com/Zach-Reid/SprintThreeLibrary.git>

Afterwards, go to the project directory with the cd command and follow the steps listed above to compile it (see The Build Process).

Deployment:

Prerequisites:

* Java Development Kit (JDK). Download JDK 8 or higher if it is not already installed on your machine. It can be obtained from sites such as the Oracle website.
* A text editor. Something that is optimized for coding is preferred for working with the code provided, such as Visual Studio Code.
* Git. While not needed normally, you will need Git if you want to clone the repository from GitHub.

Installation:

* Clone the repository with Git by using “git clone <https://github.com/Zach-Reid/SprintThreeLibrary.git>”. Alternatively, download the code as a ZIP file by clicking on the Code button and selecting “Download ZIP”, then extracting the file to your location of choice.
* Navigate to the directory where the project files are through a terminal or a command prompt. Then, compile the Java files by using “javac \*.java”. This is to compile the files.

After this, the code is set up to run by running it through the LibrarySystemDemo file.