Function Explanation File

This section explains the purpose and functionality of key classes and methods in the Library Management System.

Package: org.lms

Main.java

public static void main(String[] args)

- **Purpose:** Entry point of the application.
- **Functionality:**
  - Initializes Scanner for user input.
  - Sets up JPA EntityManagerFactory and EntityManager for the persistence unit "LibraryPU".
  - Instantiates LibrarianController and MembersController.
  - Enters a loop to display the main menu: Login as Librarian, Login as Member, or Exit.
  - Based on user choice, it calls the respective controller's MainMenu method.
  - Handles InputMismatchException for invalid menu choices.
  - Closes Scanner, EntityManager, and EntityManagerFactory on exit.

Package: org.lms.controllers

LibrarianController.java

public LibrarianController(Scanner sc, EntityManager em)

- **Purpose:** Constructor.
- **Functionality:** Initializes Scanner and service classes (AuthService, BookService, MemberService, TransactionService) with the provided EntityManager.

public void MainMenu()

- **Purpose:** Handles librarian login and displays the librarian's main operational menu.
- **Functionality:**
  - Prompts for admin username and password.
  - Authenticates using authService.authenticate().
  - If authentication is successful for "admin", it enters a loop displaying options: Manage Books, Manage Members, Manage Transactions, Exit.
  - Calls respective manage...() methods or exits based on choice.
  - Catches NoResultException, NullPointerException (for login issues), and

InputMismatchException.

public void manageBooks()

- **Purpose:** Provides functionalities for book management.
- **Functionality:** Loop with options:
  - **Add Book:** Takes book details, creates a Book object with Status.AVAILABLE, and saves it using bookService.save().
  - **Update Book stocks:** Lists books. Prompts for Book ID. Updates copiesAvailable and totalCopies. If status was UNAVAILABLE and copies become >0, it can be set to AVAILABLE. Uses bookService.update().
  - **Delete Book:** Lists available books. Prompts for Book ID. Checks for active transactions related to the book. If none (or if the check is bypassed by NoSuchElementException), sets the book's status to UNAVAILABLE and copies to 0 using bookService.update().
  - **List Books:** Displays all books using bookService.findAll() and bookService.printBooks().
  - **List Available Books:** Displays books with status = AVAILABLE using bookService.getAvailableBooks().
  - **Exit:** Exits this submenu.
  - Handles InputMismatchException.

public void manageMembers()

- **Purpose:** Provides functionalities for member management.
- **Functionality:** Loop with options:
  - **Add Members:** Takes member details. Checks if username exists via memberService.findByUsername().
    - If username exists and name matches, updates password, sets status to ACTIVE.
    - If username exists and name differs, suggests changing username.
    - Else, creates a new Member with Status.ACTIVE and saves using memberService.save().
  - **Delete Members:** Lists active members. Prompts for Member ID. Checks for member's active transactions. If none, sets member's status to INACTIVE using memberService.update().
  - **View All Members:** Displays all members using memberService.findAll() and memberService.printMembers().
  - **View All Active Members:** Displays members with status = ACTIVE.
  - **Find By ID:** Prompts for ID, displays member using memberService.findById().
  - **Find By Username:** Prompts for username, displays member using

memberService.findByUsername().
    - **Make User Active:** Lists inactive members. Prompts for ID. Sets status to ACTIVE using memberService.update().
    - **Exit:** Exits this submenu.
    - Handles InputMismatchException.

public void manageTransactions()

- **Purpose:** Provides functionalities for viewing transactions.
- **Functionality:** Loop with options:
    - **All Transactions:** Uses transactionService.findAllTransactions().
    - **Transactions of A Member:** Prompts for Member ID, uses transactionService.getMemberTransactions().
    - **Transactions of Type:** Prompts for TransactionType (BORROW/RETURN), uses transactionService.getTransactionsByType().
    - **Transactions by Status:** Prompts for status and type, filters results from transactionService.getTransactionsByType().
    - **Find By ID:** Prompts for Transaction ID, uses transactionService.findById().
    - **Find By Book ID:** Prompts for Book ID, uses transactionService.getbyBookId().
    - **Exit:** Exits this submenu.
    - Handles InputMismatchException.

MembersController.java

public MembersController(Scanner sc, EntityManager em)

- **Purpose:** Constructor.
- **Functionality:** Initializes Scanner and service classes (AuthService, BookService, TransactionService) with the provided EntityManager.

public void MainMenu()

- **Purpose:** Handles member login and displays the member's main operational menu.
- **Functionality:**
    - Prompts for member username and password.
    - Authenticates using authService.authenticate(). Ensures member is ACTIVE.
    - If successful, enters a loop displaying options: Borrow Books, Return Books, View My Transactions, Exit.
    - **Borrow Books:** Lists available books. Prompts for Book ID. Calls transactionService.borrowBook().
    - **Return Books:** Lists member's active BORROW transactions. Prompts for

Transaction ID. Calls transactionService.returnBook().
    - **View My Transactions:** Calls transactionService.getMemberTransactions().
    - **Exit:** Exits this submenu.
    - Handles InputMismatchException and other general exceptions.

Package: org.lms.dao (Data Access Objects)

BookDAO.java

- public BookDAO(EntityManager em): Constructor, initializes EntityManager.
- public void save(Book book): Persists a new Book entity within a transaction.
- public Book findById(int id): Finds a Book by its ID. Returns null if not found or on error.
- public List<Book> getAvailableBooks(): Queries for books where copiesAvailable > 0 and status = Status.AVAILABLE, ordered by bookId.
- public List<Book> findAll(): Queries for all books, ordered by bookId.
- public void delete(int id): Finds a book by ID and removes it from the database within a transaction. (Note: LibrarianController uses an update-status approach for "deletion" rather than this hard delete).
- public void update(Book book): Merges changes to an existing Book entity within a transaction.

MemberDAO.java

- public MemberDAO(EntityManager em): Constructor, initializes EntityManager.
- public void save(Member member): Persists a new Member entity within a transaction.
- public Member findById(int memberId): Finds a Member by ID. Returns null if not found or on error.
- public void update(Member member): Merges changes to an existing Member entity within a transaction.
- public void remove(int memberId): Finds a member by ID and removes them from the database within a transaction.
- public List<Member> findAll(): Queries for all members, ordered by memberId.
- public Member findByUsername(String username): Queries for a member by username. Returns null if not found or on error (e.g., NoResultException).

TransactionDAO.java

- public TransactionDAO(EntityManager em): Constructor, initializes EntityManager.
- public void save(Transaction transaction): Persists a new Transaction entity within a transaction.
- public Transaction findById(int id): Finds a Transaction by ID. Returns null if not

found or on error.
- public List<Transaction> getByType(TransactionType type): Queries for transactions of a specific TransactionType.
- public void update(Transaction transaction): Merges changes to an existing Transaction entity within a transaction. (Note: Parameter name member is misleading, it should be transaction).
- public List<Transaction> findAll(): Queries for all transactions, ordered by transactionId.
- public Transaction findOpenBorrow(int transactionID, int memberId): Finds a specific ACTIVE BORROW transaction for a given member and transaction ID, where actualReturnDate is null. Used for returning a book.
- public List<Transaction> getMembersTransactionsByType(int memberId, TransactionType type, Status status): Queries for transactions of a specific member, type, and status, ordered by transactionId.
- public List<Transaction> getbyBookId(int bookId): Queries for all transactions associated with a specific bookId, ordered by transactionId.
- public List<Transaction> findByMemberId(int memberId): Queries for all transactions associated with a specific memberId, ordered by transactionId.

Package: org.lms.service

AuthService.java

- public AuthService(EntityManager em): Constructor, initializes MemberDAO.

public Member authenticate(String username, String password)

- **Purpose:** Authenticates a user.
- **Functionality:**
  - Checks for hardcoded admin credentials ("admin"/"admin"). If matched, returns a temporary Member object representing the admin.
  - Otherwise, fetches a member by username using memberDAO.findByUsername().
  - If a member exists, their password matches, and their status is ACTIVE, returns the Member object.
  - Returns null if authentication fails.

BookService.java

- public BookService(EntityManager em): Constructor, initializes BookDAO.
- public void save(Book book): Calls bookDAO.save(book).
- public Book findById(int id): Calls bookDAO.findById(id).
- public void update(Book book): Calls bookDAO.update(book).

- public List<Book> getAvailableBooks(): Calls bookDAO.getAvailableBooks().
- public List<Book> findAll(): Calls bookDAO.findAll().
- public void printBooks(List<Book> books): Prints a formatted list of books to the console.
- public void printBook(Book b): Prints a formatted single book's details to the console.

MemberService.java

- public MemberService(EntityManager em): Constructor, initializes MemberDAO.
- public void save(Member member): Calls memberDAO.save(member).
- public Member findById(int id): Calls memberDAO.findById(id).
- public List<Member> findAll(): Calls memberDAO.findAll().
- public Member findByUsername(String username): Calls memberDAO.findByUsername(username).
- public void update(Member member): Calls memberDAO.update(member).
- public void printMembers(List<Member> members): Prints a formatted list of members to the console (passwords masked).
- public void printMember(Member m): Prints a formatted single member's details to the console (password masked).

TransactionService.java

- public TransactionService(EntityManager em): Constructor, initializes TransactionDAO, MemberDAO, and BookDAO.

public void borrowBook(int memberId, int bookId)

- **Purpose:** Handles the logic for a member borrowing a book.
- **Functionality:**
  - Fetches Book and Member. Validates their existence and status.
  - Checks if book copies are available (should be copiesAvailable > 0).
  - Decrements book.copiesAvailable, increments book.timesBorrowed, and updates the book via bookDAO.update().
  - Creates a new Transaction with type BORROW, Status.ACTIVE, current date, and an expectedReturnDate (7 days from borrow).
  - Saves the transaction using transactionDAO.save().

public void returnBook(int memberId, int transactionId)

- **Purpose:** Handles the logic for a member returning a book.
- **Functionality:**
  - Fetches the original BORROW transaction using transactionDAO.findOpenBorrow() (validates it's active and belongs to the

member).
    - Fetches the Book and Member. Validates.
    - Checks for late returns by comparing expectedReturnDate with the current date.
    - Increments book.copiesAvailable.
    - Creates a new Transaction with type RETURN, Status.COMPLETED, and actualReturnDate set to now. Saves this new transaction.
    - Updates the original BORROW transaction: sets its status to COMPLETED and actualReturnDate to now. Updates via transactionDAO.update(). (Note: borrow.setQuantity(borrowedQty - 1) seems to imply multiple copies per transaction, but borrowBook sets quantity to 1. If quantity is always 1, setting status to COMPLETED is sufficient).
- public List<Transaction> getMemberTransactionByType(int memberId, TransactionType type, Status status): Calls transactionDAO.getMembersTransactionsByType().
- public List<Transaction> getMemberTransactions(int memberId): Calls transactionDAO.findByMemberId().
- public List<Transaction> getTransactionsByType(TransactionType type): Calls transactionDAO.getByType().
- public List<Transaction> findAllTransactions(): Calls transactionDAO.findAll().
- public Transaction findById(int id): Calls transactionDAO.findById().
- public List<Transaction> getbyBookId(int bookid): Calls transactionDAO.getbyBookId().
- public void printTransactions(List<Transaction> transactions): Prints a formatted list of transactions.
- public void printTransactions(Transaction t): Prints a formatted single transaction's details.

Package: org.lms.model (Entity Classes)

Book.java

- Represents a book entity with properties like bookId (PK), title, author, genre, totalCopies, copiesAvailable, timesBorrowed, and status (Status enum).
- Includes constructors, getters, setters, and a toString() method for basic representation (though BookService has a more detailed print format).

Member.java

- Represents a member entity with properties like memberId (PK), name, username (unique), password, and status (Status enum).

- Includes constructors, getters, and setters.

Transaction.java

- Represents a transaction entity with properties like transactionId (PK), quantity (seems fixed at 1 for borrow/return), date, type (TransactionType enum), expectedReturnDate, actualReturnDate, and status (Status enum).
- Contains @ManyToOne relationships with Member and Book, indicating a member and a book involved in each transaction.
- Includes constructors, getters, setters, and a toString() method (though TransactionService uses a more detailed print format).

Package: org.lms.enums

Status.java

- Enum defining various statuses used in the system: ACTIVE, INACTIVE (for members, transactions), COMPLETED (for transactions), AVAILABLE, UNAVAILABLE (for books).

TransactionType.java

- Enum defining types of transactions: BORROW, RETURN.