### **Library Management System**

The **Library Management System** will now include additional features such as a reservation system for books, management of overdue books, and notifications for members. This expansion will allow for more complex interactions and provide a deeper dive into Java programming concepts.

#### **Key Components of the Expanded System:**

1. **Book Class:**
   * **Attributes:**
     + id: A unique identifier for each book.
     + title: The title of the book.
     + author: The author of the book.
     + genre: The genre of the book (e.g., Fiction, Non-Fiction).
     + isAvailable: A boolean indicating if the book is currently available for borrowing.
     + reservationList: A list of member IDs who have reserved the book.
   * **Methods:**
     + Constructors to initialize book objects.
     + Getters and setters for each attribute.
     + addReservation(String memberId): Adds a member to the reservation list.
     + removeReservation(String memberId): Removes a member from the reservation list.
     + toString(): Displays book details.
2. **Member Class:**
   * **Attributes:**
     + id: A unique identifier for each member.
     + name: The name of the member.
     + membershipType: The type of membership (e.g., Regular, Premium).
     + borrowedBooks: A list of book IDs currently borrowed by the member.
     + notifications: A list of notifications for the member (e.g., overdue alerts, book availability).
   * **Methods:**
     + Constructors to initialize member objects.
     + Getters and setters for each attribute.
     + addBorrowedBook(String bookId): Adds a book to the borrowed books list.
     + removeBorrowedBook(String bookId): Removes a book from the borrowed books list.
     + addNotification(String notification): Adds a notification to the member's list.
     + clearNotifications(): Clears all notifications.
     + toString(): Displays member details.
3. **Library Class:**
   * **Attributes:**
     + books: A collection (ArrayList<Book>) of all books in the library.
     + members: A collection (ArrayList<Member>) of all registered members.
     + overdueBooks: A collection (HashMap<String, Integer>) mapping book IDs to the number of overdue days.
   * **Methods:**
     + addBook(Book book): Adds a new book to the library.
     + removeBook(String bookId): Removes a book by its ID.
     + registerMember(Member member): Registers a new member.
     + removeMember(String memberId): Removes a member by their ID.
     + issueBook(String bookId, String memberId): Issues a book to a member.
     + returnBook(String bookId, String memberId): Returns a book from a member.
     + searchBookByTitle(String title): Searches for books by title.
     + searchMemberByName(String name): Searches for members by name.
     + reserveBook(String bookId, String memberId): Allows a member to reserve a book.
     + manageOverdueBooks(): Checks and updates overdue books.
     + notifyMembers(): Sends notifications to members about reservations or overdue books.
4. **Custom Exceptions:**
   * BookNotFoundException: Thrown when a requested book is not found.
   * MemberNotFoundException: Thrown when a requested member is not found.
   * InvalidOperationException: Thrown for invalid operations (e.g., issuing an unavailable book).
   * OverdueBookException: Thrown when an overdue book is attempted to be returned without clearing dues.
5. **Overdue Management and Notifications:**
   * Attributes and methods for managing overdue books and sending notifications to members regarding their borrowing status or book availability.

#### **Day 1: Project Setup and Basic Classes**

* **Objective:** Set up the project environment and implement the basic Book class.
* **Tasks:**
  + **Set Up Project Environment:** Create a new Java project in your preferred IDE (e.g., IntelliJ, Eclipse).
  + **Create Basic Folder Structure:** Organize your project with packages like models (for classes like Book, Member), services (for Library), and exceptions (for custom exceptions).
  + **Implement Book Class:**
    - Define attributes: id (String), title (String), author (String), genre (String), isAvailable (boolean).
    - Implement constructors to initialize a Book object.
    - Implement getter and setter methods for each attribute.
    - Implement a toString() method to display book details.

#### **Day 2: Implement Member Class**

* **Objective:** Create and implement the Member class with attributes and basic methods.
* **Tasks:**
  + **Create Member Class:**
    - Define attributes: id (String), name (String), membershipType (String), borrowedBooks (List<String>), notifications (List<String>).
    - Implement constructors to initialize a Member object.
    - Implement getter and setter methods for each attribute.
    - Add methods to manage the borrowedBooks list (addBorrowedBook(String bookId), removeBorrowedBook(String bookId)).
    - Implement methods to manage notifications (addNotification(String notification), clearNotifications()).
    - Implement a toString() method to display member details.

#### **Day 3: Create Basic Methods in Library Class**

* **Objective:** Start building the Library class with core attributes and methods for managing books and members.
* **Tasks:**
  + **Create Library Class:**
    - Define attributes: books (ArrayList<Book>), members (ArrayList<Member>).
  + **Implement Basic Methods:**
    - addBook(Book book): Adds a new book to the books list.
    - removeBook(String bookId): Removes a book by its ID from the books list.
    - registerMember(Member member): Adds a new member to the members list.
    - removeMember(String memberId): Removes a member by their ID from the members list.

#### **Day 4: Implement Book and Member Management**

* **Objective:** Enhance the Library class with search functionality and input validation checks.
* **Tasks:**
  + **Implement Search Methods:**
    - searchBookByTitle(String title): Iterates over the books list and returns books matching the given title.
    - searchMemberByName(String name): Iterates over the members list and returns members matching the given name.
  + **Add Input Validation:**
    - Ensure that methods like addBook, removeBook, registerMember, and removeMember include checks for duplicate entries or non-existent entries.
    - Enhance existing methods with appropriate validation logic to avoid errors or duplicates.

#### **Day 5: Develop Display Methods**

* **Objective:** Implement methods to display all books and members in the library.
* **Tasks:**
  + **Implement Display Methods:**
    - displayBooks(): Loops through the books list and prints out details for each book.
    - displayMembers(): Loops through the members list and prints out details for each member.
  + **Testing and Debugging:**
    - Test all implemented methods so far to ensure they work as expected.
    - Debug any issues that arise during testing.

#### **Day 6: Exception Handling Basics**

* **Objective:** Introduce basic exception handling with custom exceptions for better error management.
* **Tasks:**
  + **Create Custom Exceptions:**
    - BookNotFoundException: Thrown when a book is not found.
    - MemberNotFoundException: Thrown when a member is not found.
  + **Update Methods to Use Exceptions:**
    - Modify methods like removeBook, removeMember, issueBook to throw exceptions when a book or member is not found.
    - Use try-catch blocks in methods to handle these exceptions appropriately and provide meaningful error messages.

#### **Day 7: Review and Testing**

* **Objective:** Conduct a comprehensive review and testing of all functionalities implemented during the week.
* **Tasks:**
  + **Review Code:**
    - Go through all the code written so far, looking for any inconsistencies, inefficiencies, or improvements.
  + **Conduct Basic Testing:**
    - Test each functionality with various test cases to ensure everything works correctly.
  + **Debug and Refactor:**
    - Fix any bugs found during testing.
    - Refactor the code for better readability and performance where necessary.

### **Week 2: Adding Complexity and New Features**

#### **Day 8: Implement Book Issuing System**

* **Objective:** Develop the book issuing system to allow members to borrow books.
* **Tasks:**
  + **Implement issueBook(String bookId, String memberId) Method:**
    - Add logic to check if the book is available (isAvailable is true).
    - Update the isAvailable attribute to false if the book is issued.
    - Add the book ID to the member's borrowedBooks list.
    - Add error handling to ensure a book can't be issued if it's already borrowed or if the member/book doesn't exist.

#### **Day 9: Develop Book Returning System**

* **Objective:** Develop the book returning system to allow members to return borrowed books.
* **Tasks:**
  + **Implement returnBook(String bookId, String memberId) Method:**
    - Check if the book is currently borrowed by the member.
    - Update the isAvailable attribute of the book to true.
    - Remove the book ID from the member's borrowedBooks list.
    - Add error handling for cases like returning a book that wasn't borrowed or a book that doesn't exist.

#### **Day 10: Reservation System Implementation**

* **Objective:** Implement a reservation system to allow members to reserve books.
* **Tasks:**
  + **Implement reserveBook(String bookId, String memberId) Method:**
    - Add logic to check if a book is not available and allow the member to reserve it.
    - Update the reservationList in the Book class to include the member's ID.
    - Implement logic to automatically issue the book to the next member in the reservationList when it becomes available.

#### **Day 11: Overdue Book Management**

* **Objective:** Introduce overdue management to track and handle overdue books.
* **Tasks:**
  + **Implement Overdue Management:**
    - Define attributes and methods to manage overdue books in the Library class (HashMap<String, Integer> to track overdue days).
    - Implement manageOverdueBooks() to iterate through borrowed books and update overdue status based on borrowing period.
    - Add logic to calculate overdue days and update the overdueBooks map.

#### **Day 12: Develop Notification System**

* **Objective:** Develop a notification system to inform members about book availability, reservations, and overdue books.
* **Tasks:**
  + **Implement notifyMembers() Method:**
    - Add logic to send notifications to members when books become available, are reserved, or are overdue.
    - Update the Member class with methods to manage notifications (addNotification(String notification), clearNotifications()).
    - Ensure notifications are added to the member's notifications list and are displayed in the toString() method.

#### **Day 13: Exception Handling Enhancements**

* **Objective:** Enhance exception handling to cover more cases and improve robustness.
* **Tasks:**
  + **Introduce OverdueBookException:**
    - Create a new custom exception to handle cases when an overdue book is returned without clearing dues.
  + **Modify Methods for Exception Handling:**
    - Update the returnBook method to throw OverdueBookException if a book is overdue.
    - Ensure all relevant methods include appropriate try-catch blocks to handle exceptions gracefully.

#### **Day 14: Review and Code Optimization**

* **Objective:** Review and optimize the code to ensure it is efficient, readable, and maintainable.
* **Tasks:**
  + **Review All Code:**
    - Go through all the code to find any redundant or inefficient code blocks.
  + **Optimize Code:**
    - Refactor methods for better performance and readability.
    - Ensure that all methods follow best practices and are properly documented.
  + **Test Optimizations:**
    - Conduct thorough testing to ensure all optimizations work correctly without breaking functionality.

### **Week 3: Final Touches, Enhancements, and Testing**

#### **Day 15: Advanced Member Operations**

* **Objective:** Enhance member-related functionalities with more advanced operations.
* **Tasks:**
  + **Implement Additional Member Features:**
    - Add functionality to update member details (e.g., change membershipType).
    - Implement methods to clear notifications or reset a member's borrowing history.
  + **Test Member Functionalities:**
    - Test all member-related functionalities to ensure correct implementation and integration.

#### **Day 16: Advanced Book Operations**

* **Objective:** Enhance book-related functionalities with more advanced operations.
* **Tasks:**
  + **Implement Additional Book Features:**
    - Add functionality to update book details (e.g., change author or genre).
    - Implement methods to handle bulk addition or removal of books.
  + **Test Book Functionalities:**
    - Test all book-related functionalities to ensure they work as intended.

#### **Day 17: User Interface Improvements**

* **Objective:** Improve the text-based user interface to enhance user experience.
* **Tasks:**
  + **Enhance User Interface:**
    - Develop a more user-friendly text-based interface for the system.
    - Implement input validation, clear prompts, and feedback messages for user actions.
  + **Command Processing:**
    - Implement methods to handle various commands and navigate the system efficiently.

#### **Day 18: Error Handling and Validation Improvements**

* **Objective:** Improve error handling and validation across the system to handle edge cases more robustly.
* **Tasks:**
  + **Review and Enhance Error Handling:**
    - Ensure all methods have proper try-catch blocks and handle exceptions gracefully.
    - Implement meaningful error messages for users.
  + **Improve Input Validation:**
    - Add additional input validation checks for all user inputs (e.g., check for invalid IDs, empty inputs).
    - Ensure all inputs are sanitized and validated before processing.