Question Overview

You are required to design a simple library management system using Java. The system should manage books and users, allowing users to borrow and return books. The focus of this exercise is on applying object-oriented programming principles (such as encapsulation, inheritance, and polymorphism), using HashMaps for data storage, and handling errors with custom exceptions.

Requirements

1. **Classes and Objects:**
   - Create a class `Book` with the following properties:
     - `String title`
     - `String author`
     - `String ISBN`
     - `boolean isAvailable`
   - Create a class `User` with the following properties:
     - `String userId`
     - `String name`
     - `HashMap<String, Book> borrowedBooks` (Key: ISBN, Value: Book)
   - Create a class `Library` to manage books and users. This class should have the following properties:
     - `HashMap<String, Book> books` (Key: ISBN, Value: Book)
     - `HashMap<String, User> users` (Key: userId, Value: User)
2. **Custom Exceptions:**
   - Create a custom exception class `BookNotAvailableException` that is thrown when a user tries to borrow a book that is not available.
   - Create another custom exception class `UserNotFoundException` that is thrown when an operation is attempted on a non-existent user.
   - Create a custom exception class `BookLimitExceededException` that is thrown when a user tries to borrow more than 5 books.
3. **Methods in the `Library` Class:**
   - `public void addBook(Book book)`:
     - This method should add a book to the library. If a book with the same ISBN already exists, it should update the existing book's information.
     - **Note:** This method should demonstrate the concept of encapsulation by ensuring the `books` HashMap is manipulated correctly.
   - `public void registerUser(User user)`:
     - This method should register a new user in the library system. If a user with the same userId already exists, it should throw a `UserAlreadyExistsException`.
     - **Note:** This method should also demonstrate encapsulation.

- `public void borrowBook(String userId, String isbn)` throws BookNotAvailableException, UserNotFoundException, BookLimitExceededException:
  - This method should allow a user to borrow a book. The method should first check if the user exists, then check if the book is available. If the book is borrowed successfully, it should be marked as not available, and the book should be added to the user's `borrowedBooks`.
  - **Note:** This method should demonstrate the use of polymorphism, as it should handle multiple types of operations within one method.
- `public void returnBook(String userId, String isbn)` throws UserNotFoundException:
  - This method should allow a user to return a book. If the user does not exist, it should throw a `UserNotFoundException`. The book should be marked as available again and removed from the user's `borrowedBooks`.
  - **Note:** This method should demonstrate the use of abstraction by hiding the complex logic of returning a book behind a simple method call.
- `public List<Book> getUserBorrowedBooks(String userId)` throws UserNotFoundException:
  - This method should return a list of books that the user has currently borrowed.
  - **Note:** This method should show the use of HashMaps to efficiently retrieve data and should demonstrate the principle of inheritance by reusing the properties of the `User` class.
- `public void removeBook(String isbn)`:
  - This method should remove a book from the library. If the book is currently borrowed by any user, it should throw a `BookCurrentlyBorrowedException`.
  - **Note:** This method should demonstrate the use of custom exceptions to handle specific error cases.

4. **Main Class:**
   - Write a `main` method to demonstrate the working of the system by:
     - Adding several books to the library.
     - Registering multiple users.
     - Simulating a user borrowing and returning books.
     - Handling exceptions gracefully.