# AP® Computer Science A

## Library Management System Lab

Dr. Daniel Szelogowski

March 29, 2024

## 1 Introduction

This project involves creating a Library Management System (LMS) to manage books and patrons using Java. You will employ object-oriented programming principles, including classes and inheritance, use collections such as arrays and ArrayLists, and implement a binary search algorithm using recursion.

## 2 Setup

Begin by setting up your Java development environment. Your project will have the following structure:

- **LibrarySystem Interface**: Describes the methods for the Library to implement.
- **Media Class**: A base class for library items.
- **Book Class**: Inherits from Media.
- **Patron Class**: Represents a library patron.
- **Transaction Class**: Records book checkouts and returns.
- **BinarySearchUtil Class**: For searching books.
- **Library Class**: Implements the LibrarySystem interface.
- **LibraryManager Class**: The main class.

# 3 Step-by-Step Instructions

## 3.1 Media and Book Classes

1. Begin by implementing the **Media** class. This base class should include attributes common to all media types, such as title and ISBN.

2. Extend the **Media** class to create the **Book** class, adding book-specific attributes, including the author's name. Ensure you use inheritance to properly extend the Media class.

## 3.2 Patron Class

Implement the **Patron** class to represent library patrons, including:

- Attributes for the patron's name and an ID.
- An ArrayList to track the books currently checked out by the patron.
- Methods for checking out and checking in books, updating the ArrayList accordingly.

## 3.3 Transaction Class Integration

The **Transaction** class plays a critical role in tracking the circulation of books. Each time a book is checked out or returned, the **Library** class should:

1. Create a new **Transaction** instance when a book is checked out, recording the patron's ID, book's ISBN, and checkout date.

2. Update the corresponding **Transaction** when the book is returned, noting the return date.

3. Maintain a list (or a log) of transactions, allowing for historical tracking of book circulation.

## 3.4 BinarySearchUtil Class

Tasked with implementing binary search:

- Develop methods in the **BinarySearchUtil** class for searching books by title or author using recursion, optimized for sorted collections (see *Section 3.5*).

### 3.5 Library Class

The **Library** class should implement the **LibrarySystem** interface with functionalities such as:

- Adding and removing books and patrons to and from the system.
- Checking books in and out, which should interact directly with the **Transaction** class to record these actions. Use the provided `getDateToday()` for checkin/out. It will also be helpful to display an error message if the book/patron is not found or if the book is already checked in/out.
- Ensure books are maintained in a sorted order for efficient binary search implementation.

As part of managing the collection of books within the Library class, it is imperative that you maintain the books in a sorted order. This will facilitate efficient searching, especially for the implementation of the binary search algorithm. When adding a new book to the collection, ensure that it is inserted into the correct position to maintain the sorted order. Consider the following aspects:

- Books should be kept sorted by title for the title-based binary search.
- If implementing an author-based binary search, consider maintaining a separate collection sorted by author, or devise a strategy to efficiently search by author in the title-sorted collection.

### 3.6 LibraryManager Class and Main Program Functionality

Finally, in the **LibraryManager** class:

1. Create a new **Library** and populate it using the provided data file (containing ISBNs, titles, and authors).
2. Create a user interface for interacting with the Library system. This includes options for adding/removing books and patrons, and performing book checkouts and returns.
3. Utilize the **Transaction** class to log these activities, ensuring that every checkout and return is recorded.
4. Implement search functionality, allowing users to find books by title or author through the binary search methods provided by the **BinarySearchUtil** class. If the book is not found, use the `findClosestBook` method to suggest a close match.

Menu example:

```
Loading catalog...success.

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 1
Enter Name: Joe
Enter Patron ID: 123
Patron added successfully.

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 6
Enter Title: Basic Nursing
Book found: Book{title='Basic Nursing: Theory
↪  and Practice', isbn='0801678765',
↪  author='Patricia A. Potter'}

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 4
Enter ISBN: 0801678765
Enter Patron ID: 123
Book checked out successfully.

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 7
Enter ISBN: 0801678765
Transaction{isbn='0801678765', patronId='123',
↪  checkoutDate='2024-03-29',
↪  returnDate='null'}

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 5
Enter ISBN: 0801678765
Enter Patron ID: 123
Book checked in successfully.

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 7
Enter ISBN: 0801678765
Transaction{isbn='0801678765', patronId='123',
↪  checkoutDate='2024-03-29',
↪  returnDate='2024-03-29'}

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 2
Enter ISBN: 1122334455
Enter Title: Testing User Interfaces 101
Enter Author: George Oregon
Book added successfully.

Menu:
1. Add Patron
2. Add Book
3. Remove Book
4. Checkout Book
5. Return Book
6. Search Book
7. Search Book Transaction
8. Exit
Enter choice: 3
Enter ISBN: 1122334455
Book removed successfully.

Menu:
1. Add Patron
...
8. Exit
Enter choice: 8
Exiting...
```

# 4 Free Response Questions

Name:

1. Describe how inheritance is used in the Library Management System project. Specifically, discuss how the `Media` and `Book` classes demonstrate this concept. Additionally, explain how polymorphism could be utilized within this system if more types of media were to be added (e.g., Magazines, DVDs).

2. Explain the importance of encapsulation and data hiding in the context of the `Patron` class. How do getter and setter methods contribute to encapsulation, and why is it important to restrict direct access to the `checkedOutBooks` ArrayList?

3. The Library Management System uses both arrays and ArrayLists. Discuss the advantages of using an ArrayList over a traditional array when managing the collection of `Book` objects within the `Library` class. Provide examples from the project to support your discussion.

4. Describe how the binary search algorithm works and why recursion is used for its implementation in the `BinarySearchUtil` class. Provide a high-level overview of how you would implement a recursive binary search to find a book by its title in an ArrayList of Book objects sorted by title.

5. The project includes a `LibrarySystem` interface that the `Library` class implements. Discuss the purpose of using an interface in this context and how it benefits the design and scalability of the Library Management System.

6. The `LibraryManager` class serves as the main interface between the user and the Library Management System. Outline the steps you would take to design a user-friendly console interface in `LibraryManager` for adding a new book to the library, including handling user input and providing feedback. How does this design approach facilitate ease of use and maintainability?