CSCI 218: Programming II (Spring 2025)

Week 6 Lab Activity: Building a Library Application

Objective:

In this lab activity, you will implement classes to model a Book and a Library system. A Book should have attributes like title, author, and whether it is currently borrowed. A Library will store an array of books and allow operations such as borrowing and returning books, as well as adding new books to the library.

Step 1: Define the Book Class

Task:

1. Create a class named `Book`.

2. The `Book` class should have the following attributes:

    - `title` (String)
    - `author` (String)
    - `isBorrowed` (boolean) – This will keep track of whether the book is borrowed or not.
3. Implement the following methods in the `Book` class:

    - `getTitle()`: Returns the title of the book.
    - `getAuthor()`: Returns the author of the book.
    - `borrowBook()`: Marks the book as borrowed (set `isBorrowed` to `true`).
    - `returnBook()`: Marks the book as returned (set `isBorrowed` to `false`).
    - `toString()`: Returns a string representation of the book (e.g., `"Title: <title>, Author: <author>, Borrowed: <true/false>"`).

Step 2: Define the Library Class

Task:

1. Create a class named `Library`.
2. The `Library` class should contain:
    - A `Book[]` array to hold the collection of books.
    - A constructor to initialize the library with an array of books.
    - A method to borrow a book from the library:
        - `borrowBook(String title)`: This method searches for the book by its title, checks if it is available, and sets it as borrowed.
    - A method to return a book to the library:
        - `returnBook(String title)`: This method searches for the book by its title and sets it as returned.
    - A method to list all books in the library with their current borrow status:
        - `listBooks()`: This method displays all books in the library and whether they are borrowed or not.
    - A method to add a new book to the library:
        - `addBook(Book newBook)`: This method accepts a `Book` object and adds it to the library's collection. If the array of books is full, the library should expand the array to accommodate the new book.

Step 3: Test the Library and Book Classes

Task:

1. In your `main` method, create an array of `Book` objects and initialize them with sample data.
2. Create a `Library` object and pass the array of books to its constructor.
3. Demonstrate the functionality of the `Library` class by performing the following actions:
    - Borrow a book from the library by title.
    - Try to borrow a book that has already been borrowed.
    - Return a book to the library.
    - List all books in the library to check their borrow status.
    - Add a new book to the library using the `addBook` method.
    - List the updated collection of books after adding the new book.

Step 4: Additional Features

Task:

1. Add the following methods to the `Library` class:
    - `searchByTitle(String title)`: Returns the book that matches the given title (if found).
    - `searchByAuthor(String author)`: Returns a list of all books by a given author.
2. Add input validation to ensure that a user cannot borrow a book if it is already borrowed, or return a book that was never borrowed.
3. Implement the following method in the `Library` class:
    - `countAvailableBooks()`: Returns the number of books that are currently available in the library.

Expected Functionality for `addBook` Method:

- The `addBook(Book newBook)` method should:
    - Check if there is space in the current `Book[]` array. If there is space, it adds the new book.
    - If the array is full, it should dynamically resize the array to hold more books and then add the new book to the library's collection.

Submission Report: Students should submit a report that includes:

1. A description of the `Book` and `Library` classes.
2. An explanation of the methods implemented, especially the new `addBook` method.
3. Sample input/output from the program.
4. Any additional features implemented.

Evaluation Criteria

- Correct implementation of `Book` and `Library` classes.
- Proper use of object-oriented principles (encapsulation, methods).
- Successful handling of borrowing, returning, and adding books.

- Effective use of arrays to manage multiple books.
- Optional features are implemented and functional (if attempted).