CSCI 218: Programming II (Spring 2025)

Week 6 Lab Activity: Building a Library Application using Interfaces

---

Objective:

In this lab activity, you will implement interfaces to define common behaviors for `Book` and `Library` classes. You'll learn to use interfaces to enforce consistency and flexibility in object-oriented programming. You'll also practice defining common actions for different types of objects and using interfaces to interact with those objects.

Step 1: Define the Book Interface

Task:

1. Create an interface named `BookInterface` that defines the common behavior for all types of books.
    - Define the following methods:
        - `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>"`).
2. Ensure that the `Book` class implements the `BookInterface` interface. This guarantees that the `Book` class must define all of the methods specified in the `BookInterface`.

Step 2: Define the Library Interface

Task:

1. Create an interface named `LibraryInterface` that defines common behaviors for any library.
    - Define the following methods:
        - `addBook(BookInterface newBook)`: Adds a new book to the library.
        - `borrowBook(String title)`: Allows the user to borrow a book by its title.
        - `returnBook(String title)`: Allows the user to return a book by its title.
        - `listBooks()`: Displays all books in the library and their borrow status.
        - `countAvailableBooks()`: Returns the count of available books in the library.
2. Ensure that the `Library` class implements the `LibraryInterface` interface. This ensures that the `Library` class follows the behavior outlined by the interface.

Step 3: Implement the Book Class

Task:

1. Create a class named `Book` that implements the `BookInterface`.
    - Implement the methods declared in `BookInterface`, including:
        - `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.
2. Add an `isBorrowed` boolean attribute to keep track of whether a book is borrowed or available.

Step 4: Implement the Library Class

Task:

1. Create a class named `Library` that implements the `LibraryInterface`.
   - Implement the methods declared in `LibraryInterface`, including:
     - `addBook(BookInterface newBook)`: Adds a new book to the library.
     - `borrowBook(String title)`: Allows the user to borrow a book by its title.
     - `returnBook(String title)`: Allows the user to return a book by its title.
     - `listBooks()`: Displays all books in the library with their borrow status.
     - `countAvailableBooks()`: Returns the number of books currently available (not borrowed).
2. Use a dynamic structure (like `ArrayList<BookInterface>`) to store books, so you can add new books at runtime (this allows for resizing the collection dynamically).

Step 5: Testing the Library and Book Classes

Task:

1. In your `main` method, create a few `Book` objects and initialize them with sample data.
2. Create a `Library` object to store these books.
3. Test the functionality of the `Library` and `Book` classes by:
   - Adding books to the library using the `addBook()` method.
   - Borrowing and returning books.
   - Listing all books in the library.
   - Counting the number of available books.
4. Test the interface-based interaction to ensure the methods are correctly enforcing the behavior.

Step 6: Additional Features

Task:

1. Enhance the `LibraryInterface` with additional features like:
   - `searchByTitle(String title)`: Returns the book that matches the title.
   - `searchByAuthor(String author)`: Returns a list of books by a specific author.

2. Implement a `countBorrowedBooks()` method in the `Library` class that returns the number of borrowed books.

Expected Functionality with Interfaces:

- **BookInterface**:
    - Ensures that every book, regardless of its type (e.g., physical book, eBook), will implement the same common set of behaviors (e.g., `borrowBook()`, `returnBook()`).
- **LibraryInterface**:
    - Guarantees that any class that implements it (like `Library`) will offer core methods such as adding, borrowing, returning, and listing books, which can be used uniformly across different types of libraries.
- **Book and Library Classes**:
    - Implement the interface methods, ensuring consistent and predictable behavior for books and libraries.

Evaluation Criteria:

- Correct use of interfaces to define shared behaviors.
- Proper implementation of the `BookInterface` and `LibraryInterface` in the respective classes.
- Effective use of object-oriented principles (encapsulation, inheritance, interfaces).
- Proper handling of borrowing, returning, and adding books using the interfaces.
- Optional features (like search methods) are implemented successfully.

Lab Report:

Students should submit a report that includes:

1. An explanation of the `BookInterface` and `LibraryInterface`.
2. A description of how interfaces were used in the `Book` and `Library` classes.
3. Sample input/output from the program.
4. Any additional features implemented (optional).