Objective: Create a Python program that simulates a simple library management system, utilizing object-oriented programming principles.

Classes to implement: 1. `Book`: Represents a book in the library 2. `Member`: Represents a library member 3. `Library`: Manages the overall system

Features to implement: 1. Add new books to the library 2. Register new library members 3. Display available books 4. Allow members to borrow books 5. Allow members to return books 6. Search for books by title or author 7. Display borrowed books for a member

Suggested Implementation:

1. `Book` class:

    - Attributes: title, author, ISBN, publicationyear, isavailable
    - Methods: displayinfo(), markasborrowed(), markas_returned()

2. `Member` class:

    - Attributes: name, memberid, booksborrowed (list)
    - Methods: borrowbook(book), returnbook(book), displayborrowedbooks()

3. `Library` class:

    - Attributes: books (list of Book objects), members (list of Member objects)
    - Methods: addbook(), registermember(), displayavailablebooks(), borrowbook(member, book), returnbook(member, book), search_books(query)

Example Usage:

```
# Create a library
library = Library()

# Add books
library.add_book(Book("1984", "George Orwell", "9780451524935", 1949))
library.add_book(Book("To Kill a Mockingbird", "Harper Lee", "978044631078

# Register members
library.register_member(Member("Alice", "M001"))
library.register_member(Member("Bob", "M002"))

# Display available books
library.display_available_books()

# Borrow a book
member = library.members[0]
book = library.books[0]
library.borrow_book(member, book)
```

```
# Display borrowed books for a member
member.display_borrowed_books()

# Return a book
library.return_book(member, book)

# Search for books
library.search_books("1984")
```

Main Program: Implement a command-line interface that allows the user to interact with the library system, similar to the address book project, but utilizing the classes and methods of the library system.

Learning Outcomes: - Practice designing and implementing a system using multiple interrelated classes - Gain experience with object-oriented programming concepts such as encapsulation and composition - Learn to manage relationships between objects (e.g., a member borrowing a book) - Implement search functionality within a collection of objects - Practice working with lists of custom objects - Reinforce concepts of class methods vs. instance methods - Implement basic error handling and input validation in an OOP context