PYTHON PROGRAMMING FINAL PROJECT

Exercise: Library Management System

Background

In this exercise, you are tasked with building a **Library Management System (LMS)**. The system will allow the library to manage books, track borrowing and returning of books by library members, and store data persistently in a file. The system is designed to facilitate easy interaction through a command-line interface. The core of the system involves implementing object-oriented programming (OOP), data structures, file handling, and exception handling in Python.

Learning Objectives

1. **Object-Oriented Programming (OOP):**
    - Design classes (Book, Library, Member) with relevant properties and methods.
    - Implement interactions between objects of different classes.
2. **User-Defined Functions:**
    - Create functions to perform various library operations such as adding, removing, borrowing, and returning books.
3. **Data Structures:**
    - Use lists and dictionaries to store books and members.
    - Use lists within the Member class to keep track of borrowed books.
4. **File Handling:**
    - Load and save library data from a file (library_data.txt) for persistence.
5. **Exception Handling:**
    - Implement error handling to manage cases such as invalid input, non-existing books or members, and file read/write errors.

Project Requirements

1. Class Definitions

Book Class

- **Attributes**:
    - title: (str) The title of the book.

- author: (str) The author of the book.
- book_id: (str) The unique identifier for the book.
- status: (str) The status of the book, either available or borrowed.

- **Methods**:
  - __str__(): String representation of a book in the form: "book_id: title by author - Status: status".

Library Class

- **Attributes**:
  - books: (list) A list of Book objects stored in the library.
  - members: (list) A list of Member objects registered in the library.
- **Methods**:
  - add_book(book): Adds a new book to the library.
  - remove_book(book_id): Removes a book from the library by its ID.
  - find_book_by_id(book_id): Searches for a book by its ID.
  - view_books(): Displays all available books in the library.
  - borrow_book(book_id, member): Allows a member to borrow a book if it is available.
  - return_book(book_id, member): Allows a member to return a borrowed book.
  - load_data(): Loads books and members from the file library_data.txt.
  - save_data(): Saves the current books and members to library_data.txt.

Member Class

- **Attributes**:
  - name: (str) The name of the library member.
  - member_id: (str) The unique identifier for the member.
  - borrowed_books: (list) A list of Book objects the member has borrowed.
- **Methods**:
  - __str__(): String representation of the member in the form: "name (ID: member_id)".

---

2. Core Functionalities

- **View All Books**: The user should be able to view all books in the library, with their details (ID, title, author, and status).

- **Add New Book**: The user should be able to add a new book to the library by providing its title, author, and unique ID.
- **Remove Book**: The user should be able to remove a book from the library by providing its unique ID.
- **Borrow Book**: A registered member should be able to borrow a book. The system should check if the book is available. If the book is already borrowed, it should not be allowed to borrow.
- **Return Book**: A member should be able to return a borrowed book. The system should check if the member has borrowed the book and update its status to available.
- **Save and Load Data**: The system should save the current state of the library (books and members) to a file (library_data.txt). Upon restarting the program, it should load the saved data.
- **Error Handling**:
  - If the user provides an invalid book ID or member ID, the system should print an error message and handle the exception.
  - If the system cannot load or save the data file, it should handle FileNotFoundError or IOError.
  - If the user inputs a non-integer value for numerical options, the system should handle ValueError.

---

3. User Interface

The library system should present the user with a simple menu to interact with the system. The menu options will be:

1. **View Books**: Display all books in the library.
2. **Add Book**: Add a new book to the library.
3. **Remove Book**: Remove a book from the library by its ID.
4. **Borrow Book**: Borrow a book by providing the book ID and member ID.
5. **Return Book**: Return a borrowed book by providing the book ID and member ID.
6. **Exit**: Exit the system.

The program should repeatedly show the menu and allow the user to select an option until they choose to exit.

Exercise Tasks

Task 1: Create the Classes

- Implement the Book, Library, and Member classes.

- Define their attributes and methods as outlined above.

Task 2: Implement Core Functionalities

- **Add a book**: Implement functionality to add a new book to the library.
- **Remove a book**: Implement functionality to remove a book from the library by its ID.
- **View all books**: Implement functionality to display all books in the library.
- **Borrow a book**: Implement the ability for members to borrow books. Ensure that books that are already borrowed are not allowed to be borrowed again.
- **Return a book**: Implement functionality for members to return books they have borrowed.

Task 3: File Handling

- **Save data**: Implement functionality to save the list of books and members to a text file (library_data.txt).
- **Load data**: Implement functionality to load the data from the file when the program starts.

Task 4: Exception Handling

- Implement exception handling to catch and display appropriate error messages for the following situations:
    - Invalid input (e.g., non-numeric input for selecting menu options).
    - Book or member not found.
    - File errors (e.g., unable to read or write the file).

Task 5: User Interface

- Implement a text-based menu interface that allows users to interact with the system. The menu should be displayed in a loop, allowing the user to continue performing actions until they choose to exit.

Deliverables

1. **Python Code**: A Python file (library_system.py) implementing the entire system.
2. **Text File**: A sample data file (library_data.txt) that contains book and member records.