Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Homework No:	**05**
Topic:	**HAS-A relationship and access modifier**
Submission Type:	**Soft Copy Only** **Submission Link:** **https://docs.google.com/forms/d/e/1FAIpQLSctxac8-BJ-NshrdCgJWhkgvmCHiunsvwjB3smpjgTXHCcrJQ/viewform**
Resources:	**1. Class lectures** **2. BuX lectures** **a. English: https://l8.nu/toN7** **b. Supplementary: https://l8.nu/toNa**

Task 1

You are tasked with designing a Library Management System following Object-Oriented Programming (OOP) principles. In the system, you will implement the concept of OOP, encapsulation, and how different classes can have “HAS-A” relationships between them.

Read the description below to understand the classes and their methods and attributes.

Classes	Attributes	Methods
Book	title (string), author (string), genre (string), available (boolean), borrower(None if no one borrowed the book; else object of LibraryMember)	set_title(title), get_title(), set_author(author), get_author(), set_genre(genre), get_genre(), set_availability(available), get_availability(), set_borrower(borrower), get_borrower(), display_info()
LibraryMember	member_id(string), name(string), borrowed_books(list of Book objects)	set_member_id(member_id), get_member_id(), set_name(name), get_name(), borrow_book(book), return_book(book), display_borrowed_books()
Library	books_available (list of Book objects), library_members (list of LibraryMember objects)	add_book(book), add_library_member(member), display_book_list(), display_library_members()

Scenario Explanation:

In this scenario, we have three main classes: **Book, LibraryMember, and Library**. Let's see how encapsulation and "HAS-A" relationship are demonstrated:

1. Encapsulation:

- The attributes of each class are kept private and accessed through getter and setter methods. For example, the Book class has a method named 'set_availability()' to control the availability status of a book instead of directly accessing its "available" attribute.

2. "HAS-A" Relationship:

- The Library class has two attributes, `books_available` and `library_members`, which are lists containing objects of the Book and LibraryMember classes, respectively. This demonstrates the "HAS-A" relationship between Library, Book, and LibraryMember classes.
- Additionally, the LibraryMember class has an attribute `borrowed_books`, which is a list containing Book objects. This shows that each LibraryMember "HAS" multiple Book objects that they have borrowed.

Method Description:

- ***Setter methods(value):*** Setter methods update the value of the corresponding private attribute.
- ***Getter methods():*** Getter methods return the corresponding private attribute values.

Class **Book**:

- ***display_info():*** This method shows the information about a Book object such as - the book title, author, genre, and availability status.

Class **LibraryMember**:

- ***borrow_book(book):*** This method takes a book object as an argument and adds it to the borrowed books list of a library member. Remember, it will make the book unavailable to other members. Also, the library member borrowing the book will be added to the borrower variable of the book object.
- ***return_book(book):*** This method takes a book object as an argument and withdraws it from the borrowed books list. Remember, it will make the book

available for other members. Also, the borrower variable of the book object will be set to None.
- ***display_borrowed_books():*** This method shows the books which are borrowed by a library member.

Class **Library**:
- ***add_book(book):*** This method adds a book object given as an argument under the available library books.
- ***add_library_member(member):*** This method adds a library member object to the library members list.
- ***display_book_list():*** This method shows the list of books in the library.
- ***display_library_members():*** This method shows the available library members.

This system is a simple Library Management System with classes that showcase encapsulation and "HAS-A" relationship between classes. The classes work together to allow library members to borrow and return books, and the Library class manages the collection of books and library members.

You also need to write the driver code yourself to check the functionality of each of the methods. However, you can find a demo driver code here to understand the workflow of the whole code and the purpose of each method.