**Question Definition:**

Create a Python program that simulates a Library Management System. The program should cover key Object-Oriented Programming (OOP) concepts, including Classes, Objects, Inheritance, Polymorphism, Encapsulation, Abstraction, Composition, and Magic Methods. The objective is to design a system that manages library members, librarians, and books, allowing members to borrow and return books while a librarian manages the library.

**Requirements:**

**1. Classes and Objects:**

Person Class:

Attributes: name, age

Method: display()

Librarian Class (Inherits from Person):

Additional Attribute: employee\_id

Methods: display(), manage\_books()

Member Class (Inherits from Person):

Additional Attribute: member\_id

Methods: display(), borrow\_book(book), return\_book(book)

Book Class:

Attributes: title, author, isbn, available

Method: display()

Library Class:

Attribute: name

Contains a list of Book objects (Composition)

Methods: add\_book(book), display\_books()

**2. Encapsulation:**

Encapsulate data within the classes using the \_\_init\_\_ method.

Protect attributes by setting them as private or protected where necessary.

**3. Inheritance:**

Create a base class Person and derive Librarian and Member classes from it.

**4. Polymorphism:**

Override the display() method in the Librarian and Member classes.

**5. Abstraction:**

Design the system to hide complex operations such as managing books and borrowing/returning books.

**6. Composition:**

The Library class should contain a list of Book objects.

**7. Magic Methods (Dunder Methods):**

Create a BookWithMagicMethods class to demonstrate the use of \_\_str\_\_ and \_\_eq\_\_ methods.

**Tasks:**

1. Create the Person, Librarian, Member, Book, and Library classes as defined.
2. Implement encapsulation by initialising the attributes within the \_\_init\_\_ method of each class.
3. Use inheritance to derive the Librarian and Member classes from the Person class.
4. Override the display() method in Librarian and Member classes to provide specific outputs for each.
5. In the Library class, implement methods to add and display books, using composition to hold a list of Book objects.
6. Create a BookWithMagicMethods class and demonstrate the use of \_\_str\_\_ and \_\_eq\_\_ magic methods.
7. Write a main program to test all the functionalities, including borrowing and returning books, managing books by the librarian, and using the magic methods.

**Expected Output:**

The program should allow for the following interactions:

* Adding and displaying books in the library.
* A member borrowing and returning books.
* A librarian managing the books.
* Demonstrating dunder methods with custom book objects.

The output should be printed on the console, showing all actions taken by members, librarians, and the system’s responses.