

Rey Mart V. Sanchez Object Oriented Programming

21-ITE-03

OUTPUT AND EXPLANTION

Approaching the Problem

Classroom Design (Book and Library Classes):

* There are two classes created: Book, which represents individual books, and Library, which manages a collection of books. Such a design clarifies the responsibility of each class, which follows the OOP principle of encapsulation. Each class handles its own data and tasks.

Access Modifiers:

* Public: Used for properties and methods that need to be accessed from outside the class. For example, the Book title and Library name are public for easy access or viewing.
* Protected: Used for properties that should only be accessible within the class and its subclasses. For example, the author of a Book is protected so that access is limited but can be extended through inheritance.
* Private: Used for properties that should only be accessed within the class. For example, the price of the Book and the collection of books in the Library are private for stricter control and data protection.

Constructors:

* Constructors (\_\_construct) are used in both classes to initialize objects with specific values.
* In Book, the constructor initializes title, author, and price, ensuring that each object's data is consistent.
* In Library, the constructor is initialized with the name of the library, which gives each instance an identity.

Destructors:

* The destructor (\_\_destruct) in the Library is used to clean up when the object is destroyed or finished using. For example, it is used to display a message saying that the library is closed, which describes a cleanup such as closing resources or real-life connections.

Method Overloading in \_\_call():

* The magic method \_\_call() is used in the Book class to manage calls to methods that do not exist in the class, simulating overloading. For example, when updateStock() is called on a Book object, \_\_call() is triggered, which allows dynamic handling of implicit methods. This approach provides flexibility to interact with objects even if not every method is specified exactly.

Encapsulation and Data Management:

* In the Library class, the addBook() and removeBook() methods manage the collection of books, demonstrating encapsulation by keeping the collection of books private and controlling access using public methods. This protects the internal state of the Library, and only allows data modification through proper processes.

Implementation and Engagement:

Here are the steps on how to use the system:

* Create objects for the Book and an instance of the Library.
* Add books to the library, update stock using \_\_call(), remove books, and list books.
* At the end, when the Library object is destroyed, the destructor will trigger, which shows how cleanup is handled in OOP.