

### **Laboratory 3:**

#### **UML Class Diagram Assignment (V1)**

Generate a UML Class diagram and develop Python program for the following task: Design a library system that consists of three main classes: Book, Author, and Patron.

The Book class should have the following attributes and methods:

- title
- author (an Author object that wrote the book)
- publication date
- ISBN
- number of copies available
- reserve\_copy(): method to reserve a copy of the book
- return\_copy(): method to return a copy of the book

The Author class should have the following attributes and methods:

- name
- biography
- books (a list of Book objects written by the author)
- add\_book(book): method to add a Book object to the books list
- remove\_book(book): method to remove a Book object from the books list

The Patron class should have the following attributes and methods:

- name
- address
- phone number
- email address
- borrowed\_books (a list of Book objects that are currently borrowed by the patron)
- borrow\_book(book): method to borrow a Book object
- return\_book(book): method to return a Book object

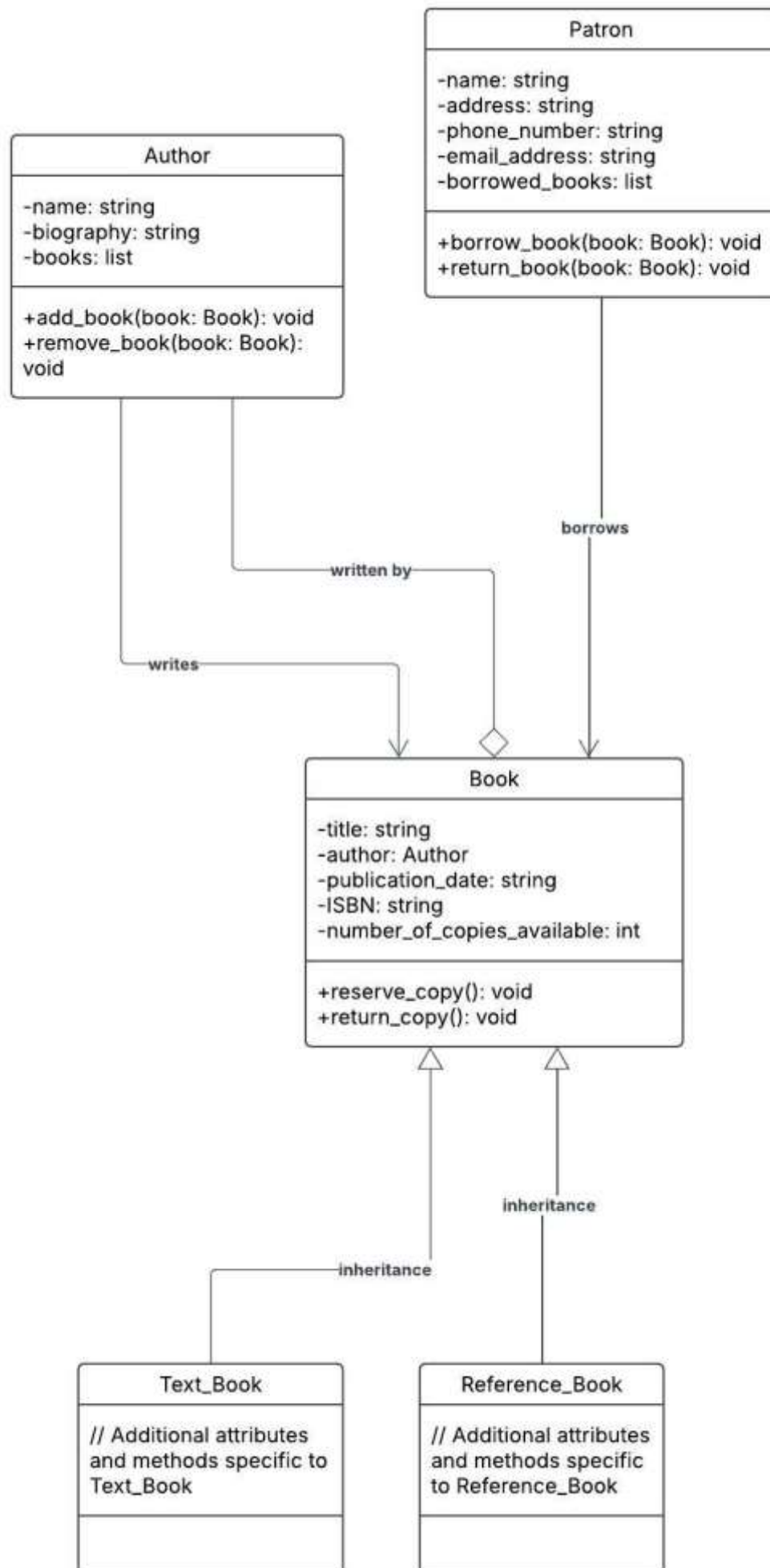
In addition to the above classes, you should create additional classes to represent the relationships between the classes, including:

- An association between Patron and Book, where a Patron can borrow multiple books.
- An aggregation relationship between Author and Book, where an Author can write multiple Books.

An inheritance relationship between Book and Text\_Book and Reference\_Book, where Text\_Book and Reference\_Book inherit from the Book class and have additional attributes and methods specific to their book type.

Implement this system in Python, using appropriate class structures and relationships to model the system. Also, create test cases to demonstrate the functionality of the system.

## CLASS DIAGRAM:



## PYTHON CODE:

class Book:

```
def __init__(self, title, author, publication_date, isbn, copies_available):
```

```
    self.title = title
```

```
    self.author = author # Author object
```

```
    self.publication_date = publication_date
```

```
    self.isbn = isbn
```

```
    self.copies_available = copies_available
```

```
def reserve_copy(self):
```

```
    if self.copies_available > 0:
```

```
        self.copies_available -= 1
```

```
        print(f"{self.title} reserved successfully.")
```

```
    else:
```

```
        print(f"No copies of {self.title} available.")
```

```
def return_copy(self):
```

```
    self.copies_available += 1
```

```
    print(f"One copy of {self.title} returned.")
```

class TextBook(Book):

```
def __init__(
```

```
    self, title, author, publication_date, isbn, copies_available, subject
```

```
):
```

```
    super().__init__(title, author, publication_date, isbn, copies_available)
```

```
    self.subject = subject
```

class ReferenceBook(Book):

```
def __init__(
```

```
    self, title, author, publication_date, isbn, copies_available, reference_type
```

```
):
```

```
    super().__init__(title, author, publication_date, isbn, copies_available)
```

```
    self.reference_type = reference_type
```

```
class Author:
```

```
    def __init__(self, name, biography):
```

```
        self.name = name
```

```
        self.biography = biography
```

```
        self.books = []
```

```
    def add_book(self, book):
```

```
        if book not in self.books:
```

```
            self.books.append(book)
```

```
    def remove_book(self, book):
```

```
        if book in self.books:
```

```
            self.books.remove(book)
```

```
class Patron:
```

```
    def __init__(self, name, address, phone_number, email):
```

```
        self.name = name
```

```
        self.address = address
```

```
        self.phone_number = phone_number
```

```
        self.email = email
```

```
        self.borrowed_books = []
```

```
    def borrow_book(self, book):
```

```
        if book.copies_available > 0:
```

```
            book.reserve_copy()
```

```
            self.borrowed_books.append(book)
```

```
            print(f"{self.name} borrowed {book.title}.")
```

```
        else:
```

```
            print(f"{book.title} is not available for borrowing.")
```

```
    def return_book(self, book):
```

```
        if book in self.borrowed_books:
```

```
            book.return_copy()
```

```
            self.borrowed_books.remove(book)
```

```
            print(f"{self.name} returned {book.title}.")
```

else:

```
print(f"{self.name} has not borrowed {book.title}.")
```

```
author1 = Author("Jose Rizal", "Filipino nationalist and writer.")
```

```
book1 = TextBook("Noli Me Tangere", author1, "1887", "1234567890", 5, "History")
```

```
book2 = ReferenceBook(  
    "El Filibusterismo", author1, "1891", "0987654321", 3, "Literature"  
)
```

```
author1.add_book(book1)
```

```
author1.add_book(book2)
```

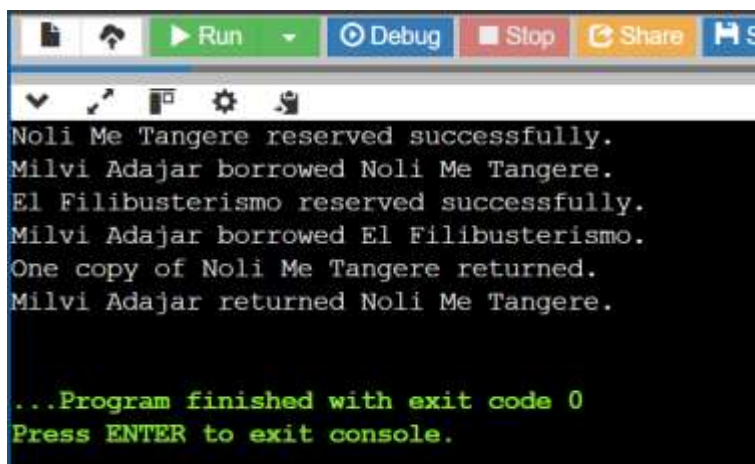
```
patron1 = Patron(  
    "Milvi Adajar",  
    "Surigao City, Brgy. Taft, Navarro St",  
    "09123204628",  
    "milvi@email.com",  
)
```

```
patron1.borrow_book(book1)
```

```
patron1.borrow_book(book2)
```

```
patron1.return_book(book1)
```

## OUTPUT:

A screenshot of a code editor's output console. The console has a toolbar at the top with icons for file operations, a 'Run' button, a 'Debug' button, a 'Stop' button, a 'Share' button, and a 'Save' button. Below the toolbar, the output text is displayed on a black background with white and green text. The text shows the successful reservation and borrowing of two books by Milvi Adajar, followed by the successful return of one copy of 'Noli Me Tangere'. The program ends with a green message indicating it finished with exit code 0 and a prompt to press ENTER to exit the console.

```
Noli Me Tangere reserved successfully.  
Milvi Adajar borrowed Noli Me Tangere.  
El Filibusterismo reserved successfully.  
Milvi Adajar borrowed El Filibusterismo.  
One copy of Noli Me Tangere returned.  
Milvi Adajar returned Noli Me Tangere.  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```