**James Carl E. Bituin** **BSCPE - 2A**

**Laboraty # 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

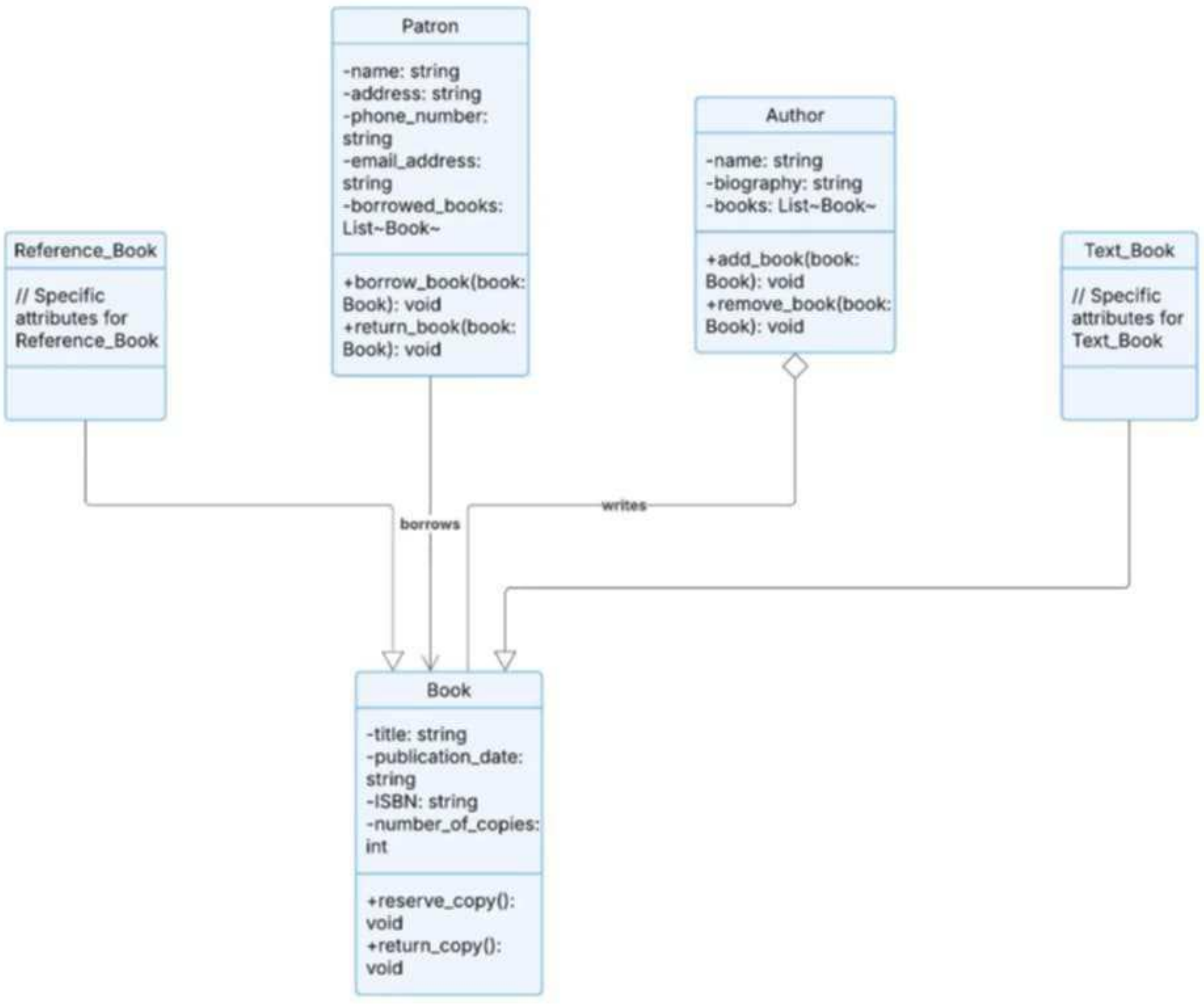
In addition to the above classes, you should create additional classes to represent the relationships between the classes, including:

- 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.

UML Diagram:



Code:

```
class Writer:
    def __init__(self, name, bio):
        self.name = name
        self.bio = bio
        self.works = []

    def add_work(self, work):
        self.works.append(work)

    def remove_work(self, work):
        if work in self.works:
            self.works.remove(work)

class Publication:
    def __init__(self, title, writer, release_date, ISBN, copies_available):
        self.title = title
        self.writer = writer
        self.release_date = release_date
        self.ISBN = ISBN
        self.copies_available = copies_available
        self.writer.add_work(self)

    def borrow_copy(self):
        if self.copies_available > 0:
            self.copies_available -= 1
            return True
        return False

    def return_copy(self):
        self.copies_available += 1

class StudyBook(Publication):
    def __init__(self, title, writer, release_date, ISBN, copies_available, topic, edition):
        super().__init__(title, writer, release_date, ISBN, copies_available)
        self.topic = topic
        self.edition = edition

class ResearchBook(Publication):
    def __init__(self, title, writer, release_date, ISBN, copies_available, section,
library_use_only):
        super().__init__(title, writer, release_date, ISBN, copies_available)
        self.section = section
        self.library_use_only = library_use_only

class Member:
    def __init__(self, name, address, phone, email):
        self.name = name
        self.address = address
        self.phone = phone
        self.email = email
        self.checked_out_books = []

    def checkout_book(self, book):
        if book.borrow_copy():
            self.checked_out_books.append(book)
            return True
        return False

    def return_book(self, book):
        if book in self.checked_out_books:
            book.return_copy()
            self.checked_out_books.remove(book)
author1 = Writer("J.K. Rowling", "British novelist famous for the Harry Potter
series.")
```

```
book1 = StudyBook("Harry Potter and the Sorcerer's Stone", author1, "1997",
"9780747532743", 5, "Fantasy", "1st")
book2 = ResearchBook("Harry Potter: A History of Magic", author1, "2017",
"9781408890776", 2, "Magic Studies", True)

member1 = Member("Alice Johnson", "123 Library St", "555-1234",
"alice@example.com")
member1.checkout_book(book1)
member1.return_book(book1)
```