Dyna Pajo Buna BSCpE-2A

Laboratory no. 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

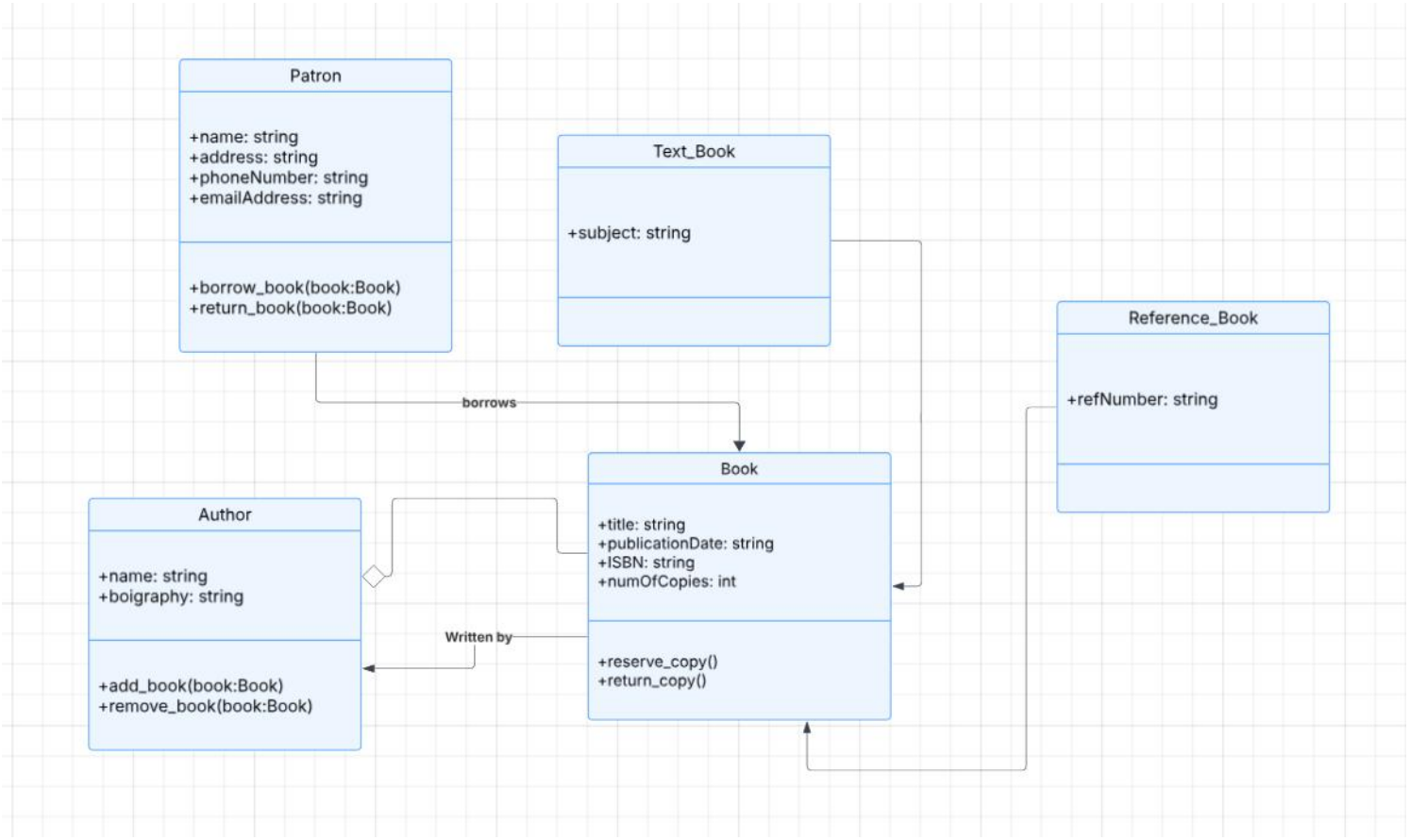
In addition to the above classes, you should create additional classes to represent the relationships betweenthe classes, including:

• 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 Book:
    def __init__(self, title, author, publication_date, isbn, num_copies):
        self.title = title
        self.author = author
        self.publication_date = publication_date
        self.isbn = isbn
        self.num_copies = num_copies

    def reserve_copy(self):
        if self.num_copies > 0:
            self.num_copies -= 1
            print(f"One copy of '{self.title}' has been reserved.")
        else:
            print(f"Sorry, no copies of '{self.title}' are available for reservation.")
```

```
    def return_copy(self):
        self.num_copies += 1
        print(f"One copy of '{self.title}' has been returned.")


# Author class
class Author:
    def __init__(self, name, biography):
        self.name = name
        self.biography = biography
        self.books = []

    def add_book(self, book):
        if isinstance(book, Book):
            self.books.append(book)
            print(f"'{book.title}' has been added to {self.name}'s books list.")

    def remove_book(self, book):
        if book in self.books:
            self.books.remove(book)
            print(f"'{book.title}' has been removed from {self.name}'s books list.")


# Patron class
class Patron:
    def __init__(self, name, address, phone_number, email_address):
        self.name = name
        self.address = address
        self.phone_number = phone_number
        self.email_address = email_address
        self.borrowed_books = []

    def borrow_book(self, book):
        if isinstance(book, Book):
            if book.num_copies > 0:
                self.borrowed_books.append(book)
                book.reserve_copy()
                print(f"{self.name} has borrowed '{book.title}'.")
            else:
                print(f"Sorry, '{book.title}' is currently out of stock.")
        else:
            print("Invalid book.")

    def return_book(self, book):
        if book in self.borrowed_books:
            self.borrowed_books.remove(book)
            book.return_copy()
            print(f"{self.name} has returned '{book.title}'.")
        else:
            print(f"{self.name} did not borrow '{book.title}'.")
```

```
class TextBook(Book):
    def __init__(self, title, author, publication_date, isbn, num_copies, subject):
        super().__init__(title, author, publication_date, isbn, num_copies)
        self.subject = subject

    def display_info(self):
        print(f"Title: {self.title}, Author: {self.author.name}, Subject: {self.subject}")

class ReferenceBook(Book):
    def __init__(self, title, author, publication_date, isbn, num_copies, reference_code):
        super().__init__(title, author, publication_date, isbn, num_copies)
        self.reference_code = reference_code

    def display_info(self):
        print(f"Title: {self.title}, Author: {self.author.name}, Reference Code: {self.reference_code}")


author1 = Author("J.K. Rowling", "British author, best known for the Harry Potter series.")
author2 = Author("J.R.R. Tolkien", "English writer, best known for The Lord of the Rings series.")
book1 = Book("Harry Potter and the Sorcerer's Stone", author1, "1997", "978-0439708180", 5)
book2 = TextBook("Calculus: Early Transcendentals", author2, "2007", "978-0495582321", 3, "Mathematics")
book3 = ReferenceBook("Oxford English Dictionary", author2, "1989", "978-0198611868", 2, "OED123")

author1.add_book(book1)
author2.add_book(book2)
author2.add_book(book3)

patron1 = Patron("Alice", "123 Main St", "555-1234", "alice@example.com")

patron1.borrow_book(book1)
patron1.borrow_book(book2)
patron1.return_book(book1)
patron1.borrow_book(book3)

book2.display_info()
book3.display_info()
```