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.

CODE:

```
class Book:

    def __init__(self, title, author, publication_date, ISBN, copies_available):

        self.title = title

        self.author = author

        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"One copy of '{self.title}' reserved.")
        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 Author:
    def __init__(self, name, biography):
        self.name = name
        self.biography = biography
        self.books = []

    def add_book(self, book):
        self.books.append(book)
        print(f"Book '{book.title}' added to author {self.name}.")

    def remove_book(self, book):
        if book in self.books:
            self.books.remove(book)
            print(f"Book '{book.title}' removed from author {self.name}.")
        else:
            print(f"Book '{book.title}' not found in {self.name}'s collection.")

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} does not have '{book.title}'.")


class TextBook(Book):
    def __init__(self, title, author, publication_date, ISBN, copies_available, subject, edition):
        super().__init__(title, author, publication_date, ISBN, copies_available)
        self.subject = subject
        self.edition = edition

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

author1 = Author("Lara Flores", "Famous for writing MUTYA NG SECTION E")
book1 = Book("Ang Mutya Ng Section E", author1, "2017", "123456789", 5)
author1.add_book(book1)
patron1 = Patron("Elaine Jane Ilola", "005 Street, City", "1234567890", "eilola08@gmail.com")
patron1.borrow_book(book1)
patron1.return_book(book1)
```

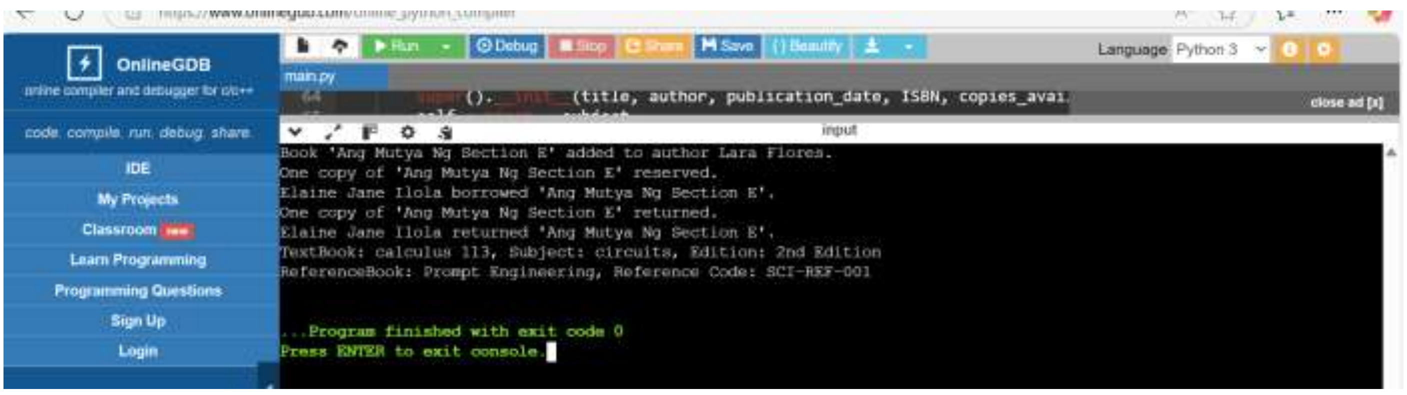
```
textbook1 = TextBook("calculus 113", "Dr. John Doe", "2020", "987654321", 3, "circuits", "2nd Edition")

refbook1 = ReferenceBook("Prompt Engineering", "Dr. Jane Doe", "2018", "1122334455", 2, "SCI-REF-001")

print(f"TextBook: {textbook1.title}, Subject: {textbook1.subject}, Edition: {textbook1.edition}")

    print(f"ReferenceBook: {refbook1.title}, Reference Code: {refbook1.reference_code}")
```

OUTPUT:

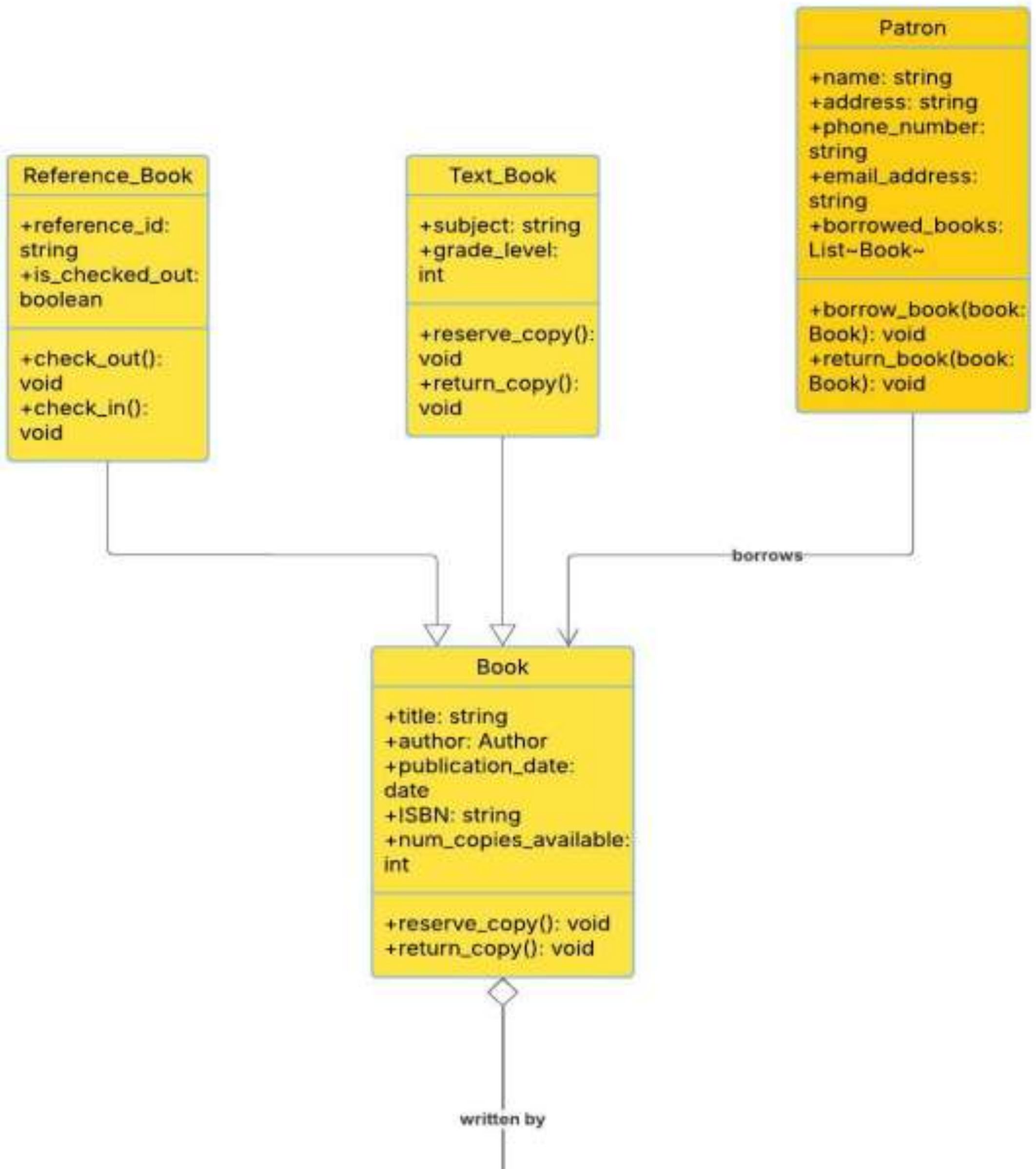


```
super().__init__(title, author, publication_date, ISBN, copies_avai
```

```
Book 'Ang Mutya Ng Section E' added to author Lara Flores.
One copy of 'Ang Mutya Ng Section E' reserved.
Elaine Jane Ilola borrowed 'Ang Mutya Ng Section E'.
One copy of 'Ang Mutya Ng Section E' returned.
Elaine Jane Ilola returned 'Ang Mutya Ng Section E'.
TextBook: calculus 113, Subject: circuits, Edition: 2nd Edition
ReferenceBook: Prompt Engineering, Reference Code: SCI-REF-001


...Program finished with exit code 0
Press ENTER to exit console.
```

CLASS DIAGRAM:


Patron
+name: string
+address: string
+phone_number: string
+email_address: string
+borrowed_books: List~Book~
+borrow_book(book: Book): void
+return_book(book: Book): void
Reference_Book
+reference_id: string
+is_checked_out: boolean
+check_out(): void
+check_in(): void
Text_Book
+subject: string
+grade_level: int
+reserve_copy(): void
+return_copy(): void
borrows
Book
+title: string
+author: Author
+publication_date: date
+ISBN: string
+num_copies_available: int
+reserve_copy(): void
+return_copy(): void
written by