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.

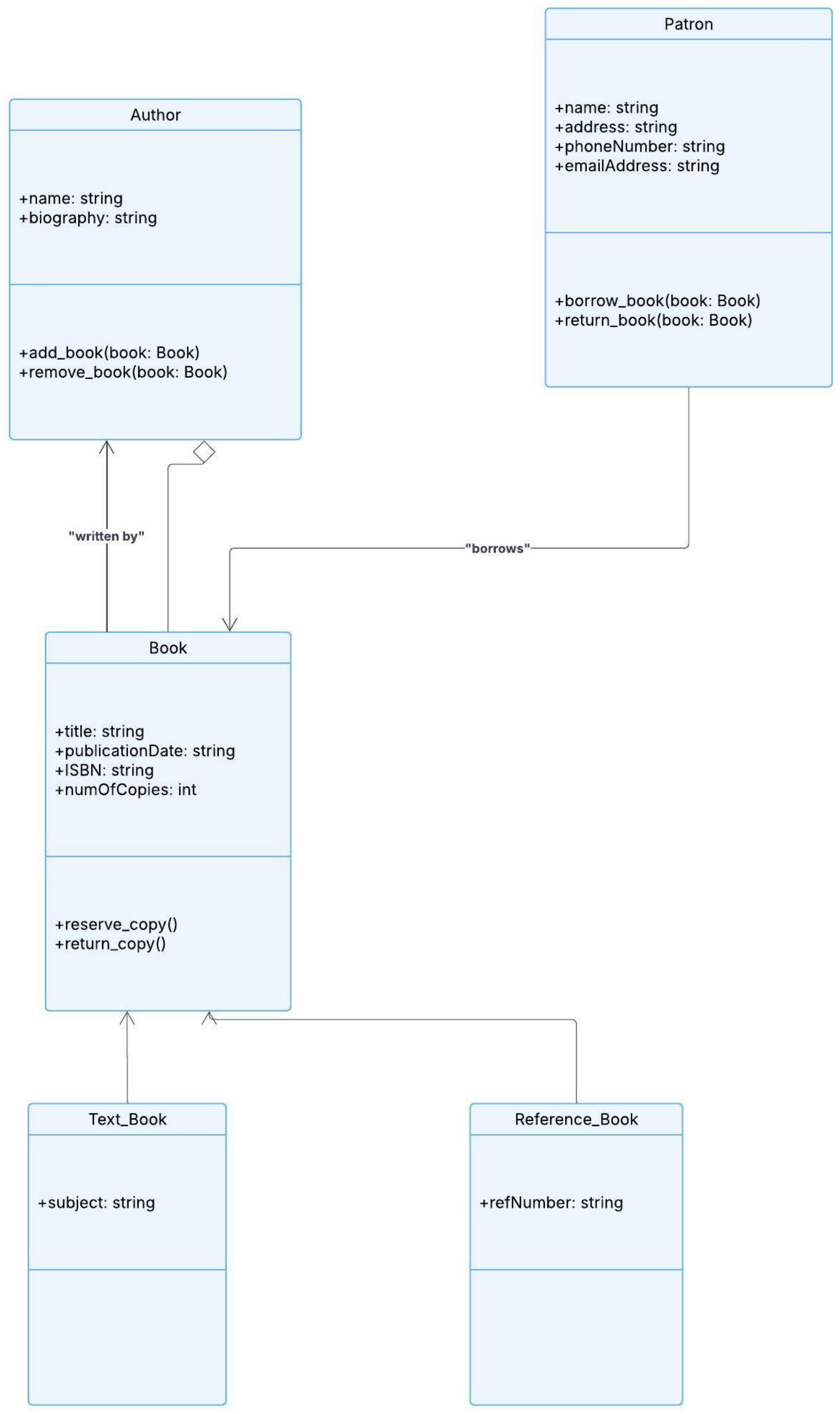
Implement this system in Python, using appropriate class structures and relationships to model the system. Also, create test cases to demonstrate the functionality of the system.

CODE:

```
class Book:
    def __init__(self, title, author, publication_date, ISBN, 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
            return True
        return False

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



class Text_Book(Book):
    def __init__(self, title, author, publication_date, ISBN, num_copies, subject):
        super().__init__(title, author, publication_date, ISBN, num_copies)
        self.subject = subject



class Reference_Book(Book):
    def __init__(self, title, author, publication_date, ISBN, num_copies, reference_section):
        super().__init__(title, author, publication_date, ISBN, num_copies)
        self.reference_section = reference_section
```

```
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, email):
        self.name = name
        self.address = address
        self.phone = phone
        self.email = email
        self.borrowed_books = []

    def borrow_book(self, book):
        if book.reserve_copy():
            self.borrowed_books.append(book)
            print(f"{self.name} borrowed {book.title}")
        else:
            print(f"Sorry, {book.title} is not available.")
    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} did not borrow {book.title}")



# Test Cases
author1 = Author("J.K. Rowling", "British author, best known for Harry Potter series.")
book1 = Book("Harry Potter and the Sorcerer's Stone", author1, "1997", "123456789", 5)
textbook1 = Text_Book("Advanced Physics", author1, "2010", "987654321", 2, "Physics")
reference_book1 = Reference_Book("Oxford Dictionary", author1, "2005", "1122334455", 3,
      "Language")


author1.add_book(book1)


david = Patron("David Smith", "123 Main St", "123-456-7890", "david@example.com")
david.borrow_book(book1)
david.borrow_book(reference_book1)
david.return_book(book1)
      david.return_book(reference_book1)
```

CLASS DIAGRAM:


Patron
+name: string
+address: string
+phoneNumber: string
+emailAddress: string
+borrow_book(book: Book)
+return_book(book: Book)
Author
+name: string
+biography: string
+add_book(book: Book)
+remove_book(book: Book)
"written by"
"borrows"
Book
+title: string
+publicationDate: string
+ISBN: string
+numOfCopies: int
+reserve_copy()
+return_copy()
Text_Book
+subject: string
Reference_Book
+refNumber: string