BAGSIC, ATHEIA KLAIRE
BSCpE 2A2
Software Design

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

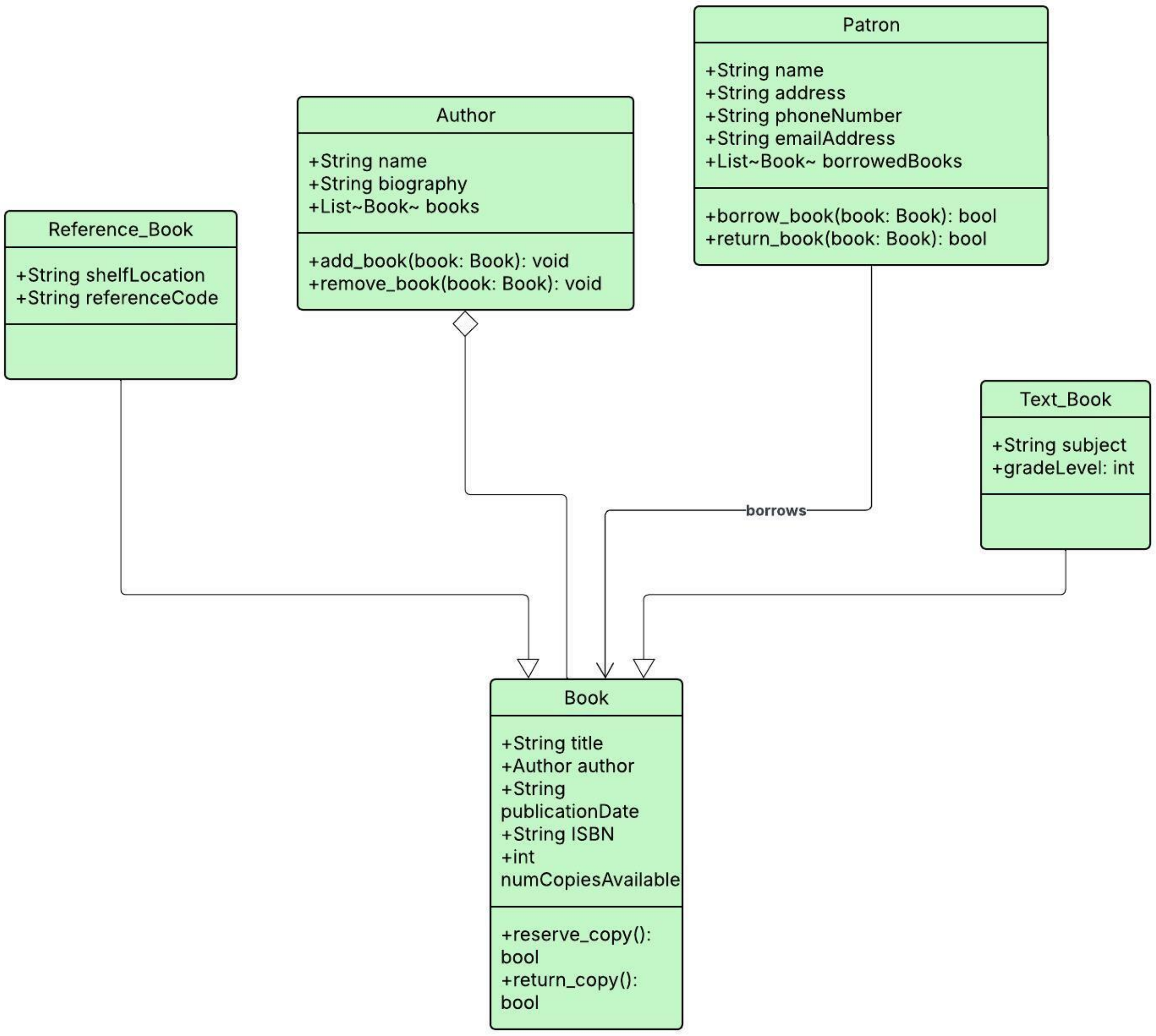
In addition to the above classes, you should create additional classes to represent the relationships between the classes, including:

- 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, publication_date, isbn, num_copies_available, author):
        self.title = title
        self.publication_date = publication_date
        self.isbn = isbn
        self.num_copies_available = num_copies_available
        self.author = author  # Association with Author

    def reserve_copy(self):
        if self.num_copies_available > 0:
            self.num_copies_available -= 1
            print(f"{self.title} reserved successfully.")
        else:
            print(f"No copies of {self.title} available for reservation.")

    def return_copy(self):
        self.num_copies_available += 1
        print(f"A copy of {self.title} has been returned.")

class TextBook(Book):
    def __init__(self, title, publication_date, isbn, num_copies_available, author, subject):
        super().__init__(title, publication_date, isbn, num_copies_available, author)
        self.subject = subject  # Additional attribute specific to TextBook

class ReferenceBook(Book):
    def __init__(self, title, publication_date, isbn, num_copies_available, author, reference_type):
        super().__init__(title, publication_date, isbn, num_copies_available, author)
        self.reference_type = reference_type  # Additional attribute specific to ReferenceBook

class Author:
    def __init__(self, name, biography):
        self.name = name
        self.biography = biography
        self.books = []  # Aggregation: Author has a list of 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_address):
        self.name = name
        self.address = address
        self.phone_number = phone_number
        self.email_address = email_address
        self.borrowed_books = []  # Association: Patron borrows books

    def borrow_book(self, book):
        if book.num_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}.")

# Example Usage
author1 = Author("George Orwell", "English novelist, essayist, journalist, and critic.")
book1 = TextBook("1984", "1949", "9780451524935", 5, author1, "Dystopian Fiction")
book2 = ReferenceBook("Animal Farm", "1945", "9780451526342", 3, author1, "Political Satire")
```

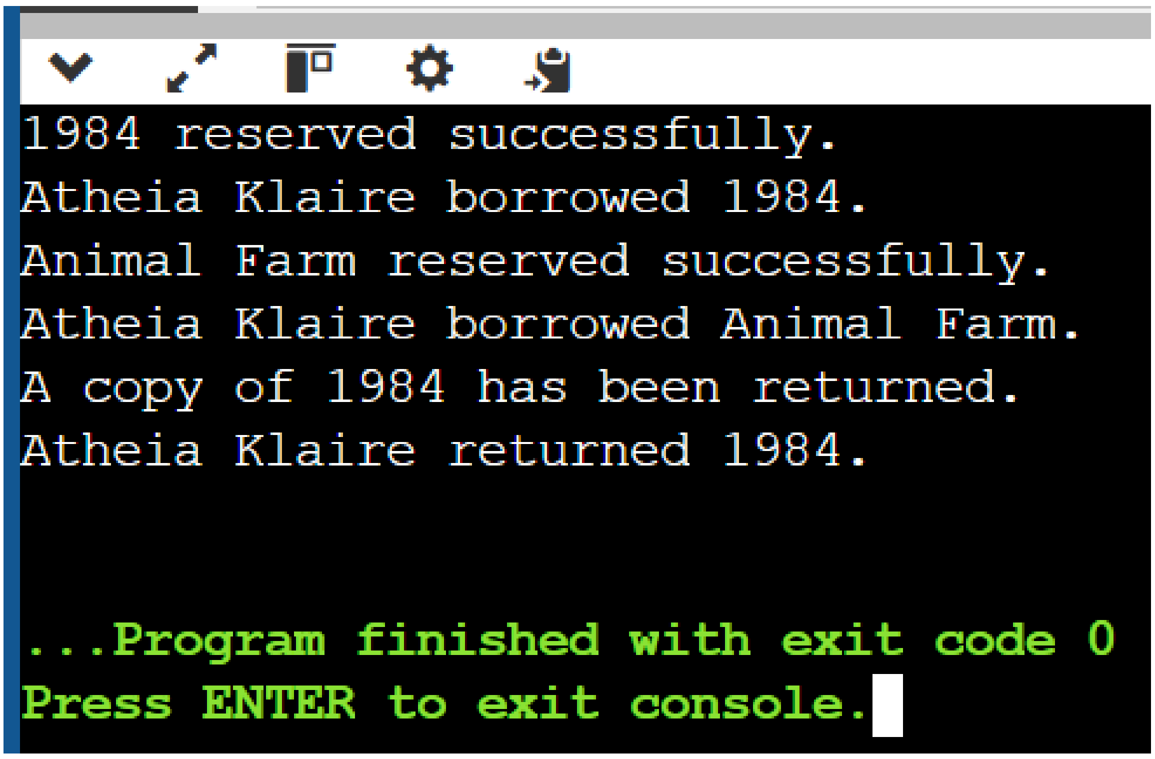
```
    author1.add_book(book1)
    author1.add_book(book2)

    patron1 = Patron("Atheia Klaire", "Surigao City", "09692083465", "atheia@email.com")

    patron1.borrow_book(book1)
    patron1.borrow_book(book2)

patron1.return_book(book1)
```

OUTPUT:



```
1984 reserved successfully.
Atheia Klaire borrowed 1984.
Animal Farm reserved successfully.
Atheia Klaire borrowed Animal Farm.
A copy of 1984 has been returned.
Atheia Klaire returned 1984.


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