**Introduction**:

For this project, I have chosen to create an object-oriented application for a library. The library will have books, authors, and patrons. The application will allow patrons to borrow and return books, search for books by title or author, and display all books currently available and checked out in the library.

**Design**:

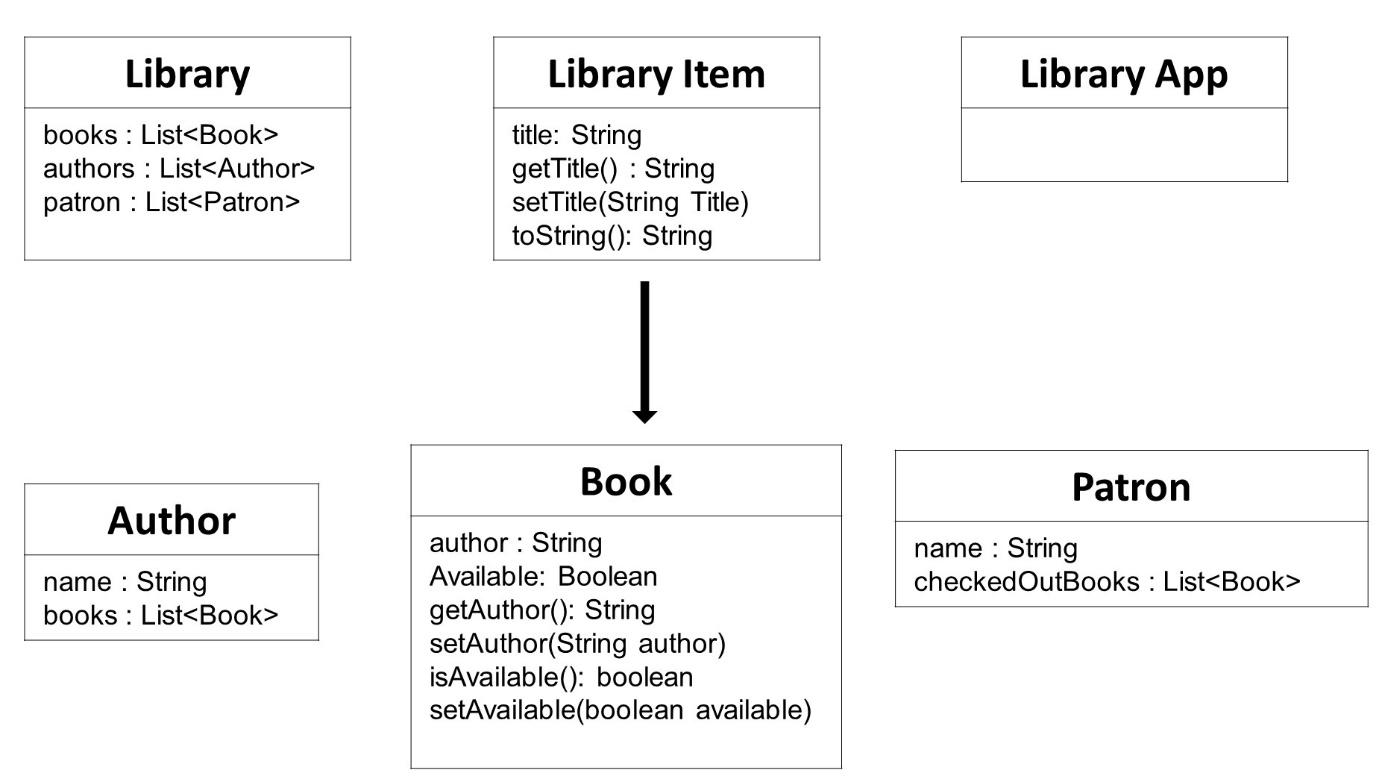
The simplified class diagram for the library application is as follows:

**Main/driver class**: LibraryApp

**Parent class**: LibraryItem

**Child classes**: Book, Author, Patron

**Group class**: Library



* The Library class will contain collections of books, authors, and patrons. It will also have methods for adding and removing items from these collections, as well as methods for searching for books by title or author.
* The Book class will have attributes for the title, author, and availability status. It will also have methods for checking out and returning the book.
* The Author class will have attributes for the author's name and a list of books written by the author.
* The Patron class will have attributes for the patron's name and a list of books currently checked out by the patron.

**Code**:

public abstract class **LibraryItem** {

private String title;

public LibraryItem(String title) {

this.title = title;

}

public String **getTitle**() {

return title;

}

public void **setTitle**(String title) {

this.title = title;

}

@Override

public String toString() {

return title;

}

}

public class **Book** extends LibraryItem {

private String author;

private boolean available;

public Book(String title, String author) {

super(title);

this.author = author;

this.available = true;

}

public String **getAuthor**() {

return author;

}

public void **setAuthor**(String author) {

this.author = author;

}

public boolean **isAvailable**() {

return available;

}

public void **setAvailable**(boolean available) {

this.available = available;

}

@Override

public String toString() {

return super.toString() + " by " + author + " (" + (available ? "Available" : "Checked Out") + ")";

}

}

public class **Author** {

private String name;

private List<Book> books;

public Author(String name) {

this.name = name;

this.books = new ArrayList<>();

}

public String **getName**() {

return name;

}

public void **setName**(String name) {

this.name = name;

}

public List<Book> **getBooks**() {

return books;

}

public void **setBooks**(List<Book> books) {

this.books = books;

}

public void **addBook**(Book book) {

books.add(book);

}

public void **removeBook**(Book book) {

books.remove(book);

}

}

public class **Patron** {

private String name;

private List<Book> checkedOutBooks;

public Patron(String name) {

this.name = name;

this.checkedOutBooks = new ArrayList<>();

}

public String **getName**() {

return name;

}

public void **setName**(String name) {

this.name = name;

}

public List<Book> **getCheckedOutBooks**() {

return checkedOutBooks;

}

public void **setCheckedOutBooks**(List<Book> checkedOutBooks) {

this.checkedOutBooks = checkedOutBooks;

}

public void **addBook**(Book book) {

checkedOutBooks.add(book);

}

public void **removeBook**(Book book) {

checkedOutBooks.remove(book);

}

}

import java.util.ArrayList;

import java.util.List;

public class **Library** {

private List<Book> books;

private List<Author> authors;

private List<Patron> patrons;

public Library() {

books = new ArrayList<>();

authors = new ArrayList<>();

patrons = new ArrayList<>();

}

public void **addBook**(Book book) {

books.add(book);

}

public void **addAuthor**(Author author) {

authors.add(author);

}

public void **addPatron**(Patron patron) {

patrons.add(patron);

}

public void **checkOutBook**(String title, String patronName) {

for (Book book : books) {

if (book.getTitle().equals(title) && book.isAvailable()) {

book.setAvailable(false);

for (Patron patron : patrons) {

if (patron.getName().equals(patronName)) {

patron.addBook(book);

break;

}

}

break;

}

}

}

public void **returnBook**(String title, String patronName) {

for (Book book : books) {

if (book.getTitle().equals(title) && !book.isAvailable()) {

book.setAvailable(true);

for (Patron patron : patrons) {

if (patron.getName().equals(patronName)) {

patron.removeBook(book);

break;

}

}

break;

}

}

}

public List<Book> **searchBooksByAuthor**(String authorName) {

List<Book> booksByAuthor = new ArrayList<>();

for (Book book : books) {

for (Author author : authors) {

if (author.getName().equals(authorName) && book.getAuthor().equals(author.getName())) {

booksByAuthor.add(book);

}

}

}

return booksByAuthor;

}

public void **displayBooks**() {

for (Book book : books) {

System.out.println(book);

}

}

}

import java.util.List;

public class **LibraryApp** {

public static void **main**(String[] args) {

Library library = new Library();

**// Add books to the library**

library.addBook(new Book("Chitra", "Rabindranath Tagore"));

library.addBook(new Book("The Great Gatsby", "F. Scott Fitzgerald"));

library.addBook(new Book("The Great", "F. Scott Fitzgerald"));

library.addBook(new Book("Great", "F. Scott Fitzgerald"));

library.addBook(new Book("To Kill a Mockingbird", "Harper Lee"));

library.addBook(new Book("Adventures of Sherlock Holmes", "Sir Arthur Conan Doyle"));

library.addBook(new Book("Alchemist", "Paulo Coelho"));

library.addBook(new Book("Anand Math", "Bankim Chandra Chatterjee"));

library.addBook(new Book("Area of Darkness", "V.S. Naipaul"));

library.addBook(new Book("As you like it", "William Shakespeare"));

library.addBook(new Book("Bitter Sweet", "Noel Coward"));

library.addBook(new Book("1984", "George Orwell"));

**// Add authors to the library**

library.addAuthor(new Author("Rabindranath Tagore"));

library.addAuthor(new Author("F. Scott Fitzgerald"));

library.addAuthor(new Author("Harper Lee"));

library.addAuthor(new Author("Sir Arthur Conan Doyle"));

library.addAuthor(new Author("Paulo Coelho"));

library.addAuthor(new Author("Bankim Chandra Chatterjee"));

library.addAuthor(new Author("V.S. Naipaul"));

library.addAuthor(new Author("William Shakespeare"));

library.addAuthor(new Author("Noel Coward"));

library.addAuthor(new Author("George Orwell"));

**// Add patrons to the library**

library.addPatron(new Patron("John Doe"));

library.addPatron(new Patron("Jane Doe"));

// Check out books to patrons

library.checkOutBook("The Great Gatsby", "John Doe");

library.checkOutBook("To Kill a Mockingbird", "Jane Doe");

**//Print the status of all books**

library.displayBooks();

**// Search books by author**

List<Book> booksByFitzgerald = library.searchBooksByAuthor("F. Scott Fitzgerald");

System.out.println("\nBooks by F. Scott Fitzgerald:");

for (Book book : booksByFitzgerald) {

System.out.println(book);

}

}

}

**Explanation**:

* **LibraryItem**: This is the base class for all library items. It has a title attribute and methods to get and set the title. It also overrides the toString() method to return the title.
* **Book**: This class extends LibraryItem and adds an author attribute and an available attribute to represent whether the book is available for checkout. It has methods to get and set the author and availability status. It also overrides the toString() method to include the author and availability status.
* **Author**: This class represents an author. It has a name attribute and a list of books written by the author. It has methods to get and set the name and books list. It also has methods to add and remove books from the list.
* **Patron**: This class represents a library patron. It has a name attribute and a list of checkedOutBooks that the patron has checked out. It has methods to get and set the name and checked out books list. It also has methods to add and remove books from the list.
* **Library**: This class represents a library. It has lists of books, authors, and patrons. It has methods to add books, authors, and patrons to the library. It also has methods to check out and return books, search books by author, and display all books in the library.
* **LibraryApp**: This class contains the main() method and is the entry point of the application. It creates a Library object and adds books, authors, and patrons to the library. It checks out books to patrons and displays the status of all books in the library. It also searches for books by author and displays the results.

The main() method in LibraryApp creates a Library object and adds books, authors, and patrons to the library. It checks out books to patrons and displays the status of all books in the library. It also searches for books by author and displays the results.

**Inheritance**:

Inheritance is used in the Book, Author, and Patron classes. The Book class inherits from the LibraryItem class to share common attributes and methods related to library items. The Author and Patron classes do not have a parent class, but they could potentially inherit from a Person class if more functionality related to people is needed in the future.

**Overloading**:

Overloading is used in the Library class for the **addBook**() method to provide flexibility and convenience when adding books to the library.

**Overriding**:

The toString() method is overridden in the Book and LibraryItem classes. The toString() method is a method of the Object class, which is the superclass of all classes in Java. It is overridden in the Book and LibraryItem classes to provide a custom string representation of the objects.

The toString() method is overridden in the Book class to provide a string representation of a book object that includes the title, author, and availability status. This allows the Book objects to be easily printed to the console or used in string concatenation.

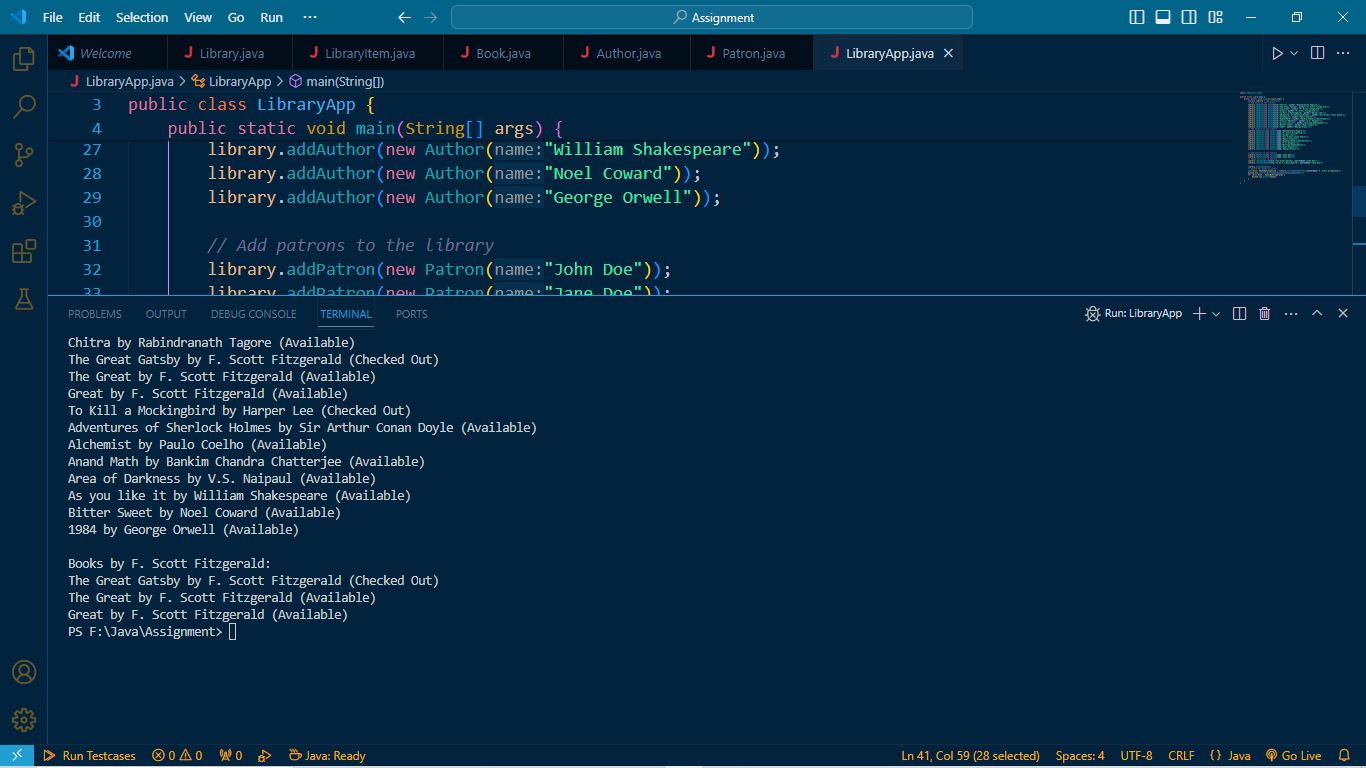
The toString() method is overridden in the LibraryItem class to provide a string representation of a library item object that includes the title. This allows the LibraryItem objects to be easily printed to the console or used in string concatenation.

In both cases, the **toString**() method is overridden to provide a more meaningful string representation of the objects, which can be useful for debugging and logging purposes.

**Polymorphism:**

Polymorphism simplifies the code by allowing methods like **addBook()** to accept any object that is a subclass of **LibraryItem**, not just instances of the **LibraryItem** class itself. This flexibility makes the code more adaptable to changes and promotes better code organization and reusability.

**Screen-shot:**

****