CONFUSED YOUTH

BORROWING BOOK AT THE LIBRARY

CONFUSED YOUTH

CONFUSED YOUTH TEAM

Indra

ITB STIKOM BALI

Gung Yoga

ITB STIKOM BALI

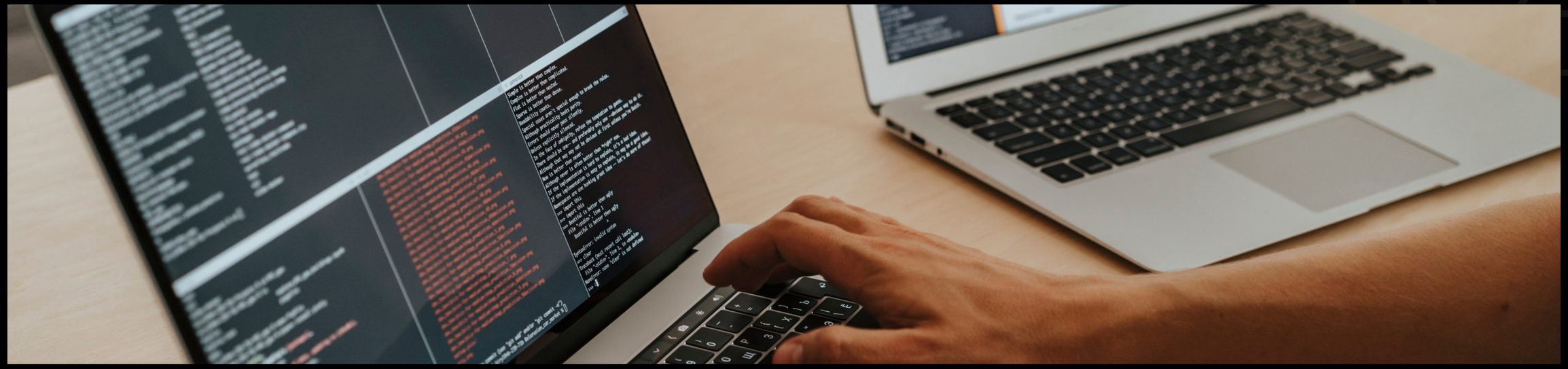
Genta

ITB STIKOM BALI

Kent

DNUI

CONFUSED YOUTH

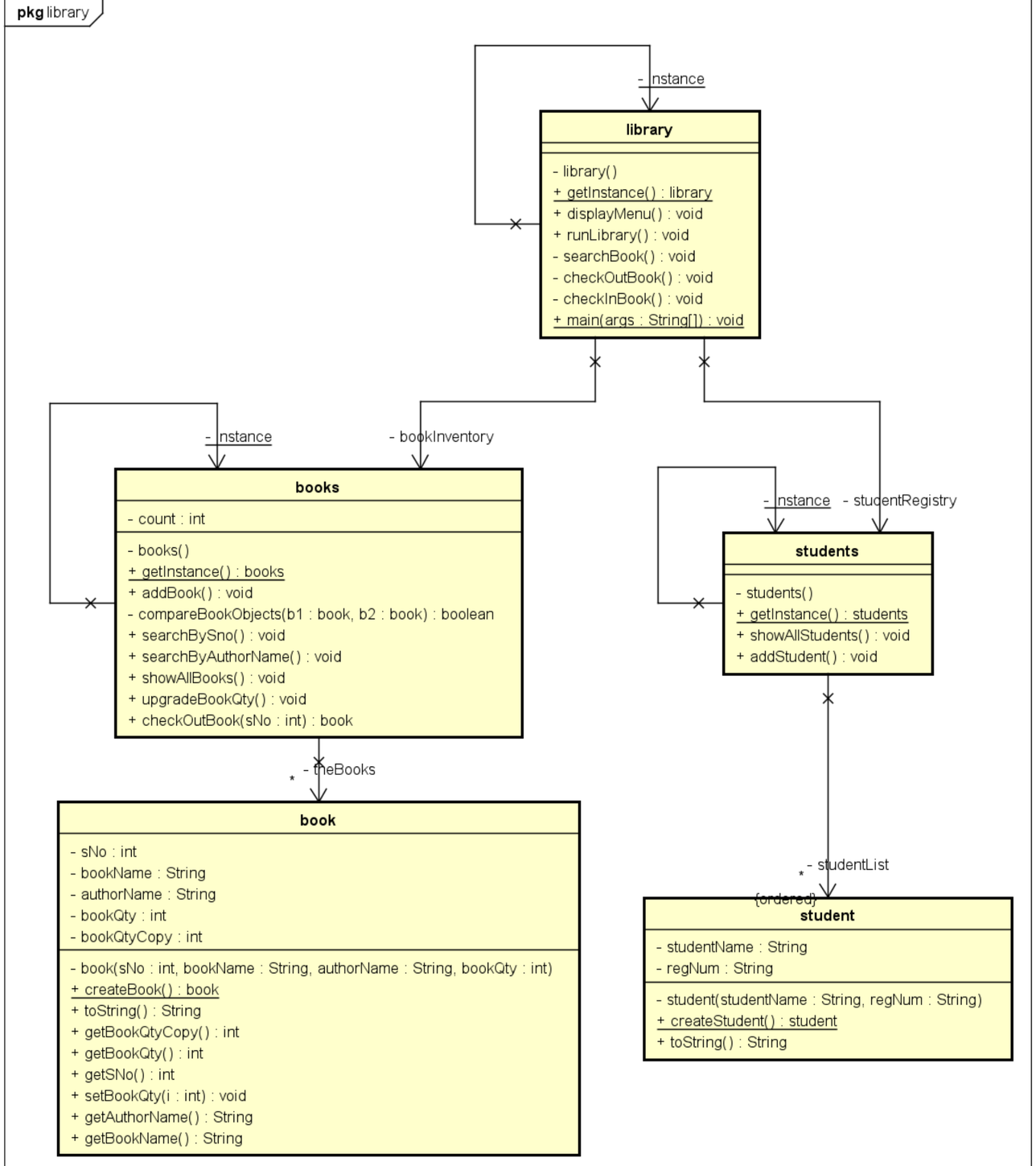


PROJECT INTRODUCTION

Developed in the Java programming language. This is an application that helps users manage borrowing books at the library. Users can add book, member, borrowing and returning data.

CONFUSED YOUTH

CLASS DIAGRAM



DESIGN PATTERN

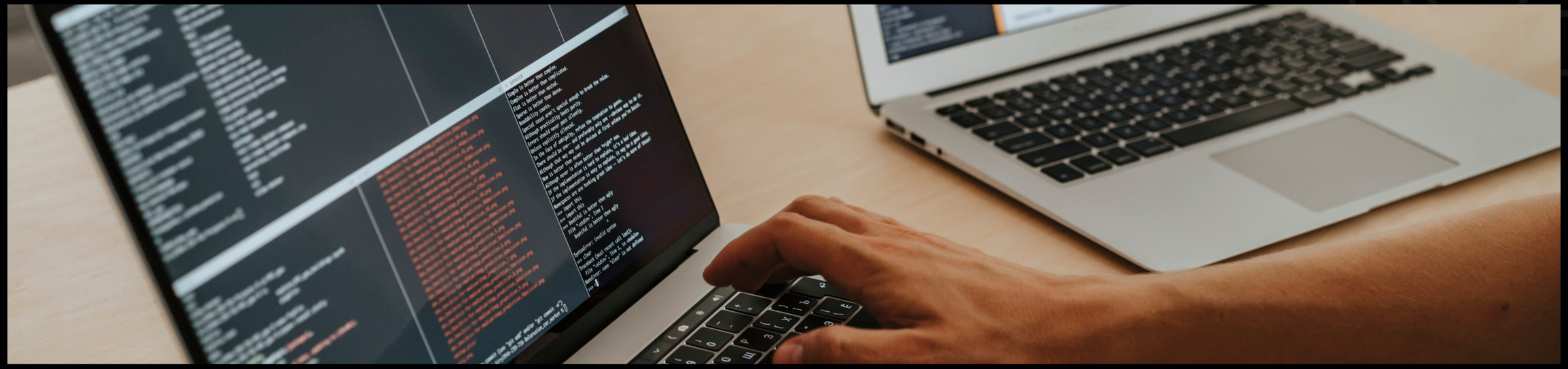
SINGLETON PATTERN

DESIGN PATTERN

THE SINGLETON PATTERN IS A CREATIONAL DESIGN PATTERN THAT RESTRICTS THE INSTANTIATION OF A CLASS TO A SINGLE INSTANCE AND PROVIDES A GLOBAL POINT OF ACCESS TO THAT INSTANCE. IT ENSURES THAT ONLY ONE INSTANCE OF A CLASS EXISTS IN THE SYSTEM AND PROVIDES A WAY TO ACCESS IT FROM ANYWHERE WITHIN THE CODEBASE.

THE SINGLETON PATTERN IS OFTEN USED IN SCENARIOS WHERE THERE IS A NEED FOR A SINGLE, SHARED RESOURCE, SUCH AS A DATABASE CONNECTION, FILE SYSTEM, LOGGER, OR THREAD POOL. BY USING THE SINGLETON PATTERN, YOU CAN ENSURE THAT MULTIPLE INSTANCES OF THESE RESOURCES ARE NOT CREATED, WHICH CAN HELP WITH RESOURCE MANAGEMENT AND COORDINATION.

CONFUSED YOUTH



LIBRARY PATTERN CODE

LIBRARY.JAVA

THIS CLASS REPRESENTS THE LIBRARY MANAGEMENT SYSTEM. IT CONTROLS THE FLOW OF THE APPLICATION AND INTERACTS WITH THE BOOKS AND STUDENTS CLASSES.
THE CLASS FOLLOWS THE SINGLETON PATTERN, ENSURING THAT ONLY ONE INSTANCE OF THE LIBRARY CLASS EXISTS.
THE RUNLIBRARY() METHOD STARTS THE LIBRARY MANAGEMENT SYSTEM AND DISPLAYS A MENU FOR VARIOUS OPERATIONS.
THE CLASS PROVIDES METHODS FOR ADDING BOOKS, UPGRADING BOOK QUANTITIES, SEARCHING BOOKS, DISPLAYING ALL BOOKS, REGISTERING STUDENTS, DISPLAYING REGISTERED STUDENTS, CHECKING OUT BOOKS, AND CHECKING IN BOOKS.

students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > library.java > library > getInstance()

You, 9 minutes ago | 1 author (You)

```java
package library;

import java.util.Scanner;

You, 9 minutes ago | 1 author (You)
public class library {
    private books bookInventory;
    private students studentRegistry;

    private static library instance;

    private library() {
        bookInventory = books.getInstance();
        studentRegistry = students.getInstance();
    }

    public static library getInstance() {
        if (instance == null) {
            instance = new library();       You, 9 minutes ago • menambahkan
        }
        return instance;
    }

    public void displayMenu() {
        System.out.println("Enter 0 to Exit Application.");
        System.out.println("Enter 1 to Add new Book.");
        System.out.println("Enter 2 to Upgrade Quantity of a Book.");
        System.out.println("Enter 3 to Search a Book.");
        System.out.println("Enter 4 to Show All Books.");
        System.out.println("Enter 5 to Register Student.");
        System.out.println("Enter 6 to Show All Registered Students.");
        System.out.println("Enter 7 to Check Out Book.");
        System.out.println("Enter 8 to Check In Book.");
    }

    public void runLibrary() {
        Scanner input = new Scanner(System.in);
        int choice;

        do {
            displayMenu();
            System.out.println("Enter your choice:");
            choice = input.nextInt();
            input.nextLine();

            switch (choice) {
                case 0:
                    System.out.println("Exiting application...");
```

students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > library.java > library > runLibrary()

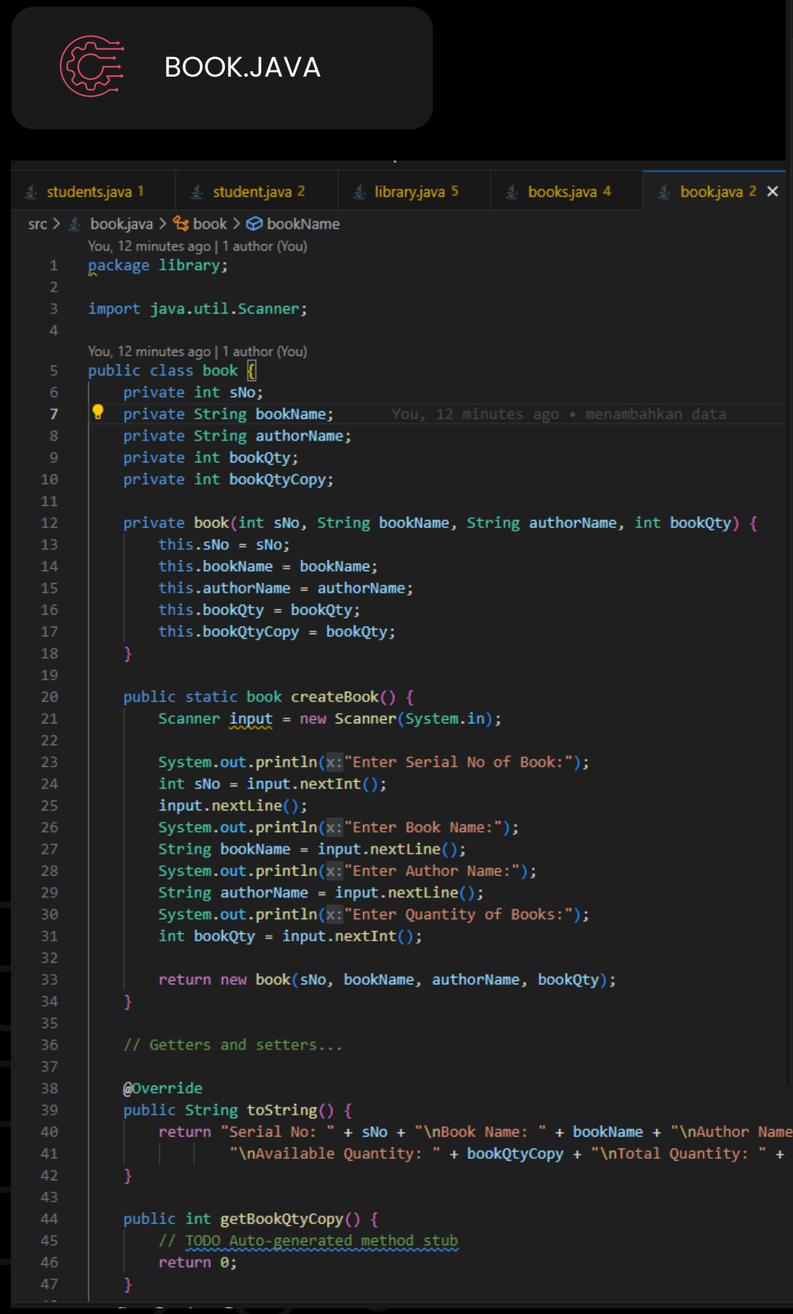
```java
                case 1:
                    bookInventory.addBook();
                    break;
                case 2:
                    bookInventory.upgradeBookQty();
                    break;
                case 3:
                    searchBook();
                    break;
                case 4:
                    bookInventory.showAllBooks();
                    break;
                case 5:
                    studentRegistry.addStudent();
                    break;
                case 6:
                    studentRegistry.showAllStudents();
                    break;
                case 7:
                    checkOutBook();
                    break;
                case 8:
                    checkInBook();
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
                    break;      You, 10 minutes ago • menambahkan data
            }

            System.out.println("----------------------------");
        } while (choice != 0);
    }

    private void searchBook() {
        Scanner input = new Scanner(System.in);
        int searchChoice;

        System.out.println("Enter 1 to Search by Serial Number.");
        System.out.println("Enter 2 to Search by Author Name.");
        System.out.println("Enter your choice:");
        searchChoice = input.nextInt();

        switch (searchChoice) {
            case 1:
                bookInventory.searchBySno();
                break;
            case 2:
                bookInventory.searchByAuthorName();
                break;
```

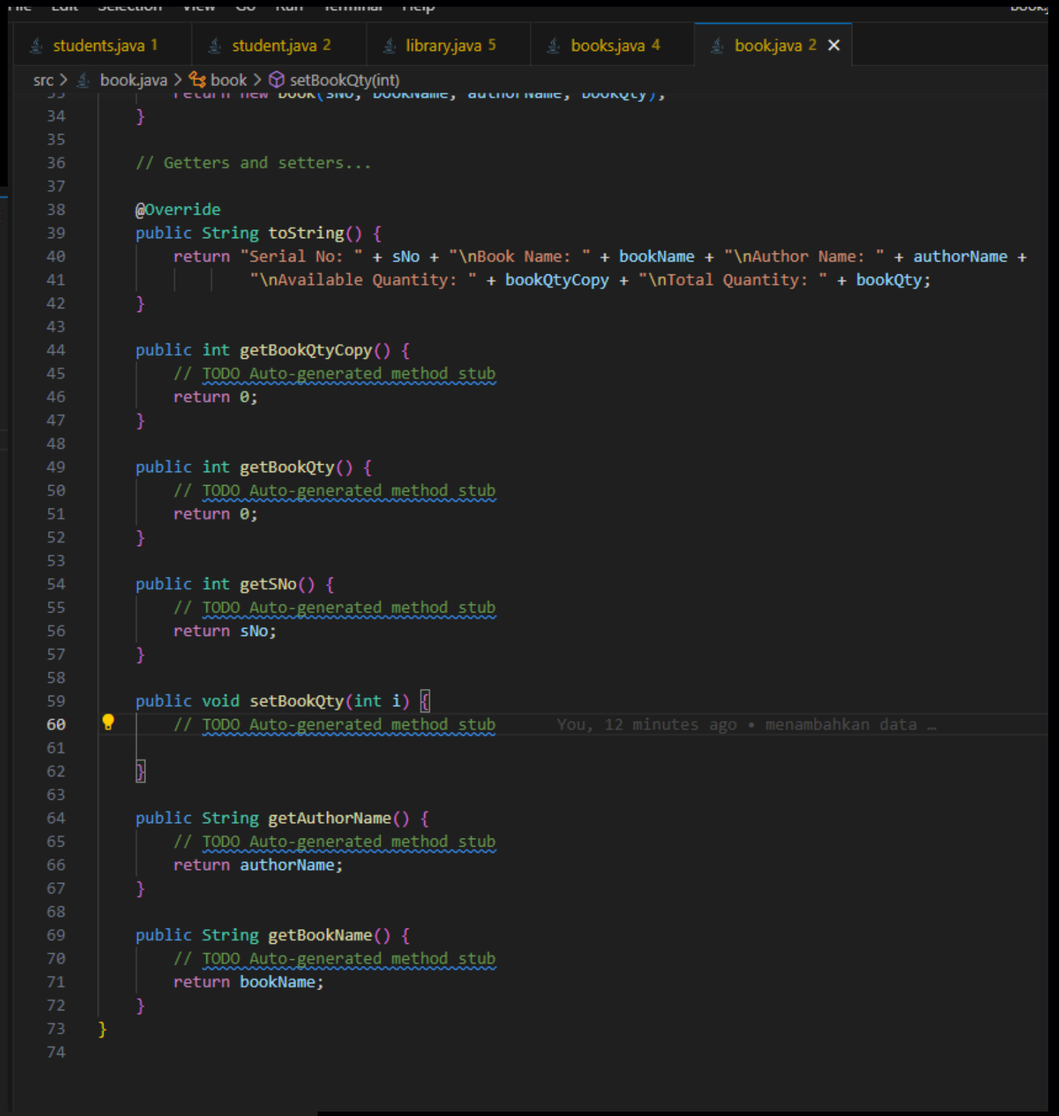
students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > library.java > library > checkInBook()

```java
                break;
            default:
                System.out.println("Invalid choice. Returning to main menu...");
                break;
        }
    }

    private void checkOutBook() {
        Scanner input = new Scanner(System.in);

        System.out.println("Enter Serial No of Book to be Checked Out:");
        int sNo = input.nextInt();
        input.nextLine();

        book checkedOutBook = bookInventory.checkOutBook(sNo);

        if (checkedOutBook != null) {
            System.out.println("Book checked out successfully!");
            // Perform additional operations if needed
        }
    }

    private void checkInBook() {
        Scanner input = new Scanner(System.in);

        System.out.println("Enter Serial No of Book to be Checked In:");
        int sNo = input.nextInt();
        input.nextLine();

        book checkedInBook = bookInventory.checkOutBook(sNo);

        if (checkedInBook != null) {      You, 11 minutes ago • menambahkan data
            System.out.println("Book checked in successfully!");
            // Perform additional operations if needed
        }
    }

    Run | Debug
    public static void main(String[] args) {
        library lib = library.getInstance();
        lib.runLibrary();
    }
}
```

BOOK.JAVA





THIS CLASS REPRESENTS A BOOK IN THE LIBRARY. IT HAS PRIVATE FIELDS FOR THE SERIAL NUMBER, BOOK NAME, AUTHOR NAME, BOOK QUANTITY, AND AVAILABLE BOOK QUANTITY.

THE CONSTRUCTOR IS PRIVATE TO PREVENT DIRECT INSTANTIATION. INSTEAD, THE ADDBOOK() METHOD IN THE BOOKS CLASS IS RESPONSIBLE FOR CREATING BOOK OBJECTS.

THE CLASS PROVIDES GETTERS AND SETTERS FOR ACCESSING AND MODIFYING THE BOOK DETAILS.

BOOKS.JAVA

THIS CLASS REPRESENTS THE BOOK INVENTORY IN THE LIBRARY. IT MAINTAINS AN ARRAY OF BOOK OBJECTS AND A COUNT OF THE NUMBER OF BOOKS.
THE CLASS FOLLOWS THE SINGLETON PATTERN, ENSURING THAT ONLY ONE INSTANCE OF THE BOOKS CLASS EXISTS.
THE ADDBOOK() METHOD ADDS A BOOK TO THE INVENTORY AFTER CHECKING FOR DUPLICATES BASED ON BOOK NAME AND SERIAL NUMBER.
THE CLASS PROVIDES METHODS FOR SEARCHING BOOKS BY SERIAL NUMBER AND AUTHOR NAME, UPGRADING THE QUANTITY OF A BOOK, AND DISPLAYING ALL BOOKS IN THE INVENTORY.

students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > books.java > books > books()

You, 13 minutes ago | 1 author (You)

```java
package library;

import java.util.Scanner;

You, 13 minutes ago | 1 author (You)
public class books {
    private book[] theBooks;
    private int count;

    private static books instance;

    private books() {
        theBooks = new book[50];
        count = 0;
    }       You, 13 minutes ago • menambahkan data

    public static books getInstance() {
        if (instance == null) {
            instance = new books();
        }
        return instance;
    }

    public void addBook() {
        if (count >= 50) {
            System.out.println("No Space to Add More Books.");
            return;
        }

        book newBook = book.createBook();

        for (int i = 0; i < count; i++) {
            if (compareBookObjects(newBook, theBooks[i])) {
                System.out.println("Book with the same Serial No or Name already exists.");
                return;
            }
        }

        theBooks[count] = newBook;
        count++;

        System.out.println("Book added successfully!");
    }

    private boolean compareBookObjects(book b1, book b2) {
        return b1.getBookName().equalsIgnoreCase(b2.getBookName()) ||
                b1.getSNo() == b2.getSNo();
    }
```

students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > books.java > books > books()

```java
    private boolean compareBookObjects(book b1, book b2) {
        return b1.getBookName().equalsIgnoreCase(b2.getBookName()) ||
                b1.getSNo() == b2.getSNo();
    }

    public void searchBySno() {
        System.out.println("Enter Serial No of Book:");
        Scanner input = new Scanner(System.in);
        int sNo = input.nextInt();
        boolean found = false;

        for (int i = 0; i < count; i++) {
            if (theBooks[i].getSNo() == sNo) {
                System.out.println(theBooks[i]);
                found = true;
            }
        }
        if (!found) {
            System.out.println("No book with Serial No " + sNo + " found.");
        }
    }

    public void searchByAuthorName() {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter Author Name:");
        String authorName = input.nextLine();
        boolean found = false;

        for (int i = 0; i < count; i++) {
            if (theBooks[i].getAuthorName().equalsIgnoreCase(authorName)) {
                System.out.println(theBooks[i]);
                found = true;
            }
        }

        if (!found) {
            System.out.println("No books by " + authorName + " found.");
        }
    }

    public void showAllBooks() {
        if (count == 0) {
            System.out.println("No books available.");
            return;
        }

        System.out.println("Showing all books:");
        for (int i = 0; i < count; i++) {
            System.out.println(theBooks[i]);
```

students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > books.java > books > checkOutBook(int)

```java
    public void showAllBooks() {
        if (count == 0) {
            System.out.println("No books available.");
            return;
        }

        System.out.println("Showing all books:");
        for (int i = 0; i < count; i++) {
            System.out.println(theBooks[i]);
            System.out.println("----------------------------");
        }
    }

    public void upgradeBookQty() {
        System.out.println("Enter Serial No of Book:");
        Scanner input = new Scanner(System.in);
        int sNo = input.nextInt();

        for (int i = 0; i < count; i++) {
            if (theBooks[i].getSNo() == sNo) {
                System.out.println("Enter No of Books to be Added:");
                int addingQty = input.nextInt();
                theBooks[i].setBookQty(theBooks[i].getBookQty() + addingQty);
                theBooks[i].setBookQty(theBooks[i].getBookQtyCopy() + addingQty);
                System.out.println("Quantity upgraded successfully!");
                return;
            }
        }

        System.out.println("No book with Serial No " + sNo + " found.");
    }

    public book checkOutBook(int sNo) {
        // TODO Auto-generated method stub
        return null;       You, 14 minutes ago • menambahkan data …
    }
}
```

STUDENT.JAVA

students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > student.java > {} library

```java
You, 15 minutes ago | 1 author (You)
package library;        You, 15 minutes ago • menambahkan data …

import java.util.Scanner;

You, 15 minutes ago | 1 author (You)
public class student {
    private String studentName;
    private String regNum;

    private student(String studentName, String regNum) {
        this.studentName = studentName;
        this.regNum = regNum;
    }

    public static student createStudent() {
        Scanner input = new Scanner(System.in);

        System.out.println(x:"Enter Student Name:");
        String studentName = input.nextLine();
        System.out.println(x:"Enter Reg Number:");
        String regNum = input.nextLine();

        return new student(studentName, regNum);
    }

    // Getters and setters...

    @Override
    public String toString() {
        return "Student Name: " + studentName + "\nRegistration Number: " + regNum;
    }
}
```

THIS CLASS REPRESENTS A STUDENT IN THE LIBRARY. IT HAS PRIVATE FIELDS FOR THE STUDENT'S NAME AND REGISTRATION NUMBER.

THE CONSTRUCTOR IS PRIVATE TO PREVENT DIRECT INSTANTIATION. INSTEAD, THE ADDSTUDENT() METHOD IN THE STUDENTS CLASS IS RESPONSIBLE FOR CREATING STUDENT OBJECTS.

THE CLASS PROVIDES GETTERS AND SETTERS FOR ACCESSING AND MODIFYING THE STUDENT DETAILS.

STUDENTS.JAVA

students.java 1 | student.java 2 | library.java 5 | books.java 4 | book.java 2

src > students.java > students

You, 16 minutes ago | 1 author (You)

```java
package library;

import java.util.ArrayList;
import java.util.List;

You, 16 minutes ago | 1 author (You)
public class students {
    private List<student> studentList;

    private static students instance;

    private students() {
        studentList = new ArrayList<>();
    }
       You, 16 minutes ago • menambahkan data
    public static students getInstance() {
        if (instance == null) {
            instance = new students();
        }
        return instance;
    }

    public void addStudent() {
        student newStudent = student.createStudent();
        studentList.add(newStudent);
        System.out.println(x:"Student registered successfully!");
    }

    public void showAllStudents() {
        if (studentList.isEmpty()) {
            System.out.println(x:"No students registered.");
            return;
        }

        System.out.println(x:"Showing all registered students:");
        for (student s : studentList) {
            System.out.println(s);
            System.out.println(x:"----------------------------");
        }
    }
}

```

THIS CLASS REPRESENTS THE STUDENT REGISTRY IN THE LIBRARY. IT MAINTAINS A LIST OF STUDENT OBJECTS.

THE CLASS FOLLOWS THE SINGLETON PATTERN, ENSURING THAT ONLY ONE INSTANCE OF THE STUDENTS CLASS EXISTS.

THE ADDSTUDENT() METHOD ADDS A STUDENT TO THE REGISTRY.

THE CLASS PROVIDES A METHOD FOR DISPLAYING ALL REGISTERED STUDENTS.

SALFORD & CO.

THANK YOU

CONFUSED YOUTH

OUR PROGRAM'S VIDEOS

CONFUSED YOUTH TEAM