



CONFUSED YOUTH

# BORROWING BOOK AT THE LIBRARY



CONFUSED YOUTH

# CONFUSED YOUTH TEAM



**Indra**  
ITB STIKOM BALI



**Gung Yoga**  
ITB STIKOM BALI

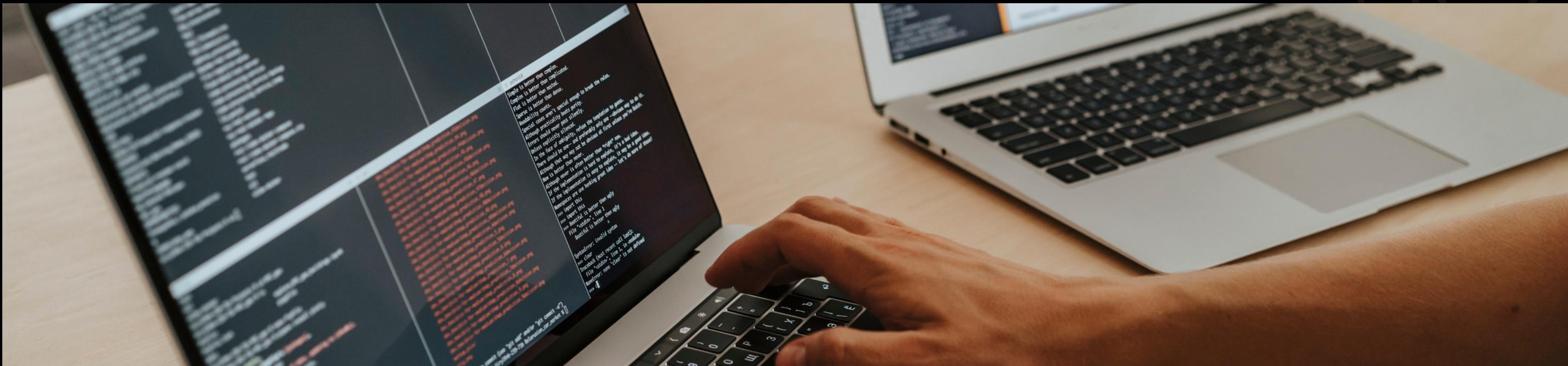


**Genta**  
ITB STIKOM BALI



**Kent**  
DNUI





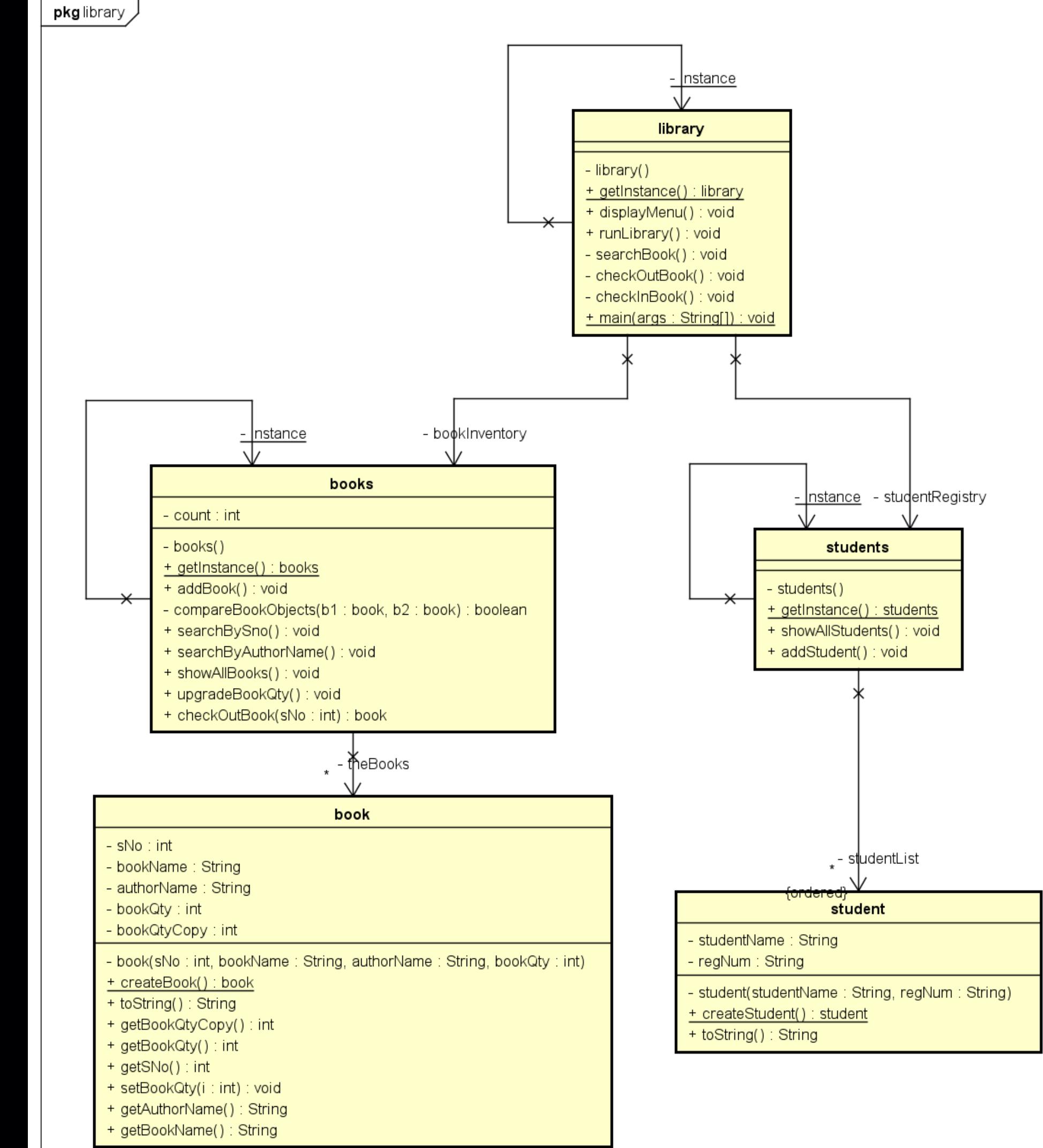
# PROJECT INTRODUCTION

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

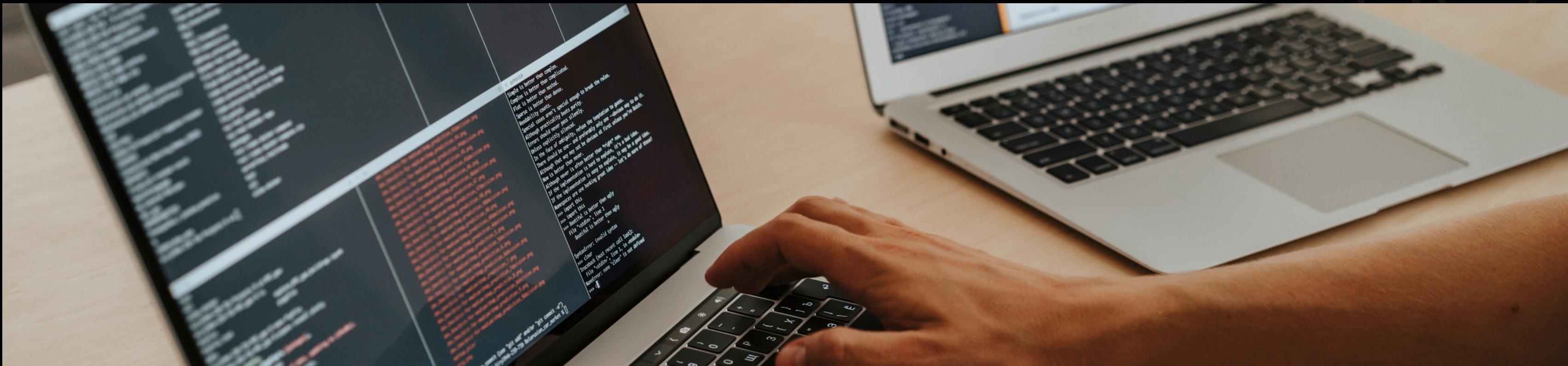
# DESIGN PATTERN

## SINGLETON 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.





# LIBRARY PATTERN CODE



## LIBRARY.JAVA

```
src > library.java > library > getInstance()
You, 9 minutes ago | 1 author (You)
1 package library;
2
3 import java.util.Scanner;
4
5 You, 9 minutes ago | 1 author (You)
6 public class library {
7     private books bookInventory;
8     private students studentRegistry;
9
10    private static library instance;
11
12    private library() {
13        bookInventory = books.getInstance();
14        studentRegistry = students.getInstance();
15    }
16
17    public static library getInstance() {
18        if (instance == null) {
19            instance = new library(); You, 9 minutes ago * menambahkan
20        }
21        return instance;
22    }
23
24    public void displayMenu() {
25        System.out.println("Enter 0 to Exit Application.");
26        System.out.println("Enter 1 to Add new Book.");
27        System.out.println("Enter 2 to Upgrade Quantity of a Book.");
28        System.out.println("Enter 3 to Search a Book.");
29        System.out.println("Enter 4 to Show All Books.");
30        System.out.println("Enter 5 to Register Student.");
31        System.out.println("Enter 6 to Show All Registered Students.");
32        System.out.println("Enter 7 to Check Out Book.");
33        System.out.println("Enter 8 to Check In Book.");
34
35    public void runLibrary() {
36        Scanner input = new Scanner(System.in);
37        int choice;
38
39        do {
40            displayMenu();
41            System.out.println("Enter your choice:");
42            choice = input.nextInt();
43            input.nextLine();
44
45            switch (choice) {
46                case 0:
47                    System.out.println("Exiting application...");
48
49                case 1:
50                    bookInventory.addBook();
51                    break;
52                case 2:
53                    bookInventory.upgradeBookQty();
54                    break;
55                case 3:
56                    searchBook();
57                    break;
58                case 4:
59                    bookInventory.showAllBooks();
60                    break;
61                case 5:
62                    studentRegistry.addStudent();
63                    break;
64                case 6:
65                    studentRegistry.showAllStudents();
66                    break;
67                case 7:
68                    checkOutBook();
69                    break;
70                case 8:
71                    checkInBook();
72                    break;
73                default:
74                    System.out.println("Invalid choice. Please try again.");
75                    break;
76
77                }
78
79            System.out.println("-----");
80        } while (choice != 0);
81
82    private void searchBook() {
83        Scanner input = new Scanner(System.in);
84        int searchChoice;
85
86        System.out.println("Enter 1 to Search by Serial Number.");
87        System.out.println("Enter 2 to Search by Author Name.");
88        System.out.println("Enter your choice:");
89        searchChoice = input.nextInt();
90
91        switch (searchChoice) {
92            case 1:
93                bookInventory.searchBySno();
94                break;
95            case 2:
96                bookInventory.searchByAuthorName();
97                break;
98        }
99
100    }
101
102    private void checkOutBook() {
103        Scanner input = new Scanner(System.in);
104
105        System.out.println("Enter Serial No of Book to be Checked Out:");
106        int sNo = input.nextInt();
107        input.nextLine();
108
109        book checkedOutBook = bookInventory.checkOutBook(sNo);
110
111        if (checkedOutBook != null) {
112            System.out.println("Book checked out successfully!");
113            // Perform additional operations if needed
114        }
115    }
116
117    private void checkInBook() {
118        Scanner input = new Scanner(System.in);
119
120        System.out.println("Enter Serial No of Book to be Checked In:");
121        int sNo = input.nextInt();
122        input.nextLine();
123
124        book checkedInBook = bookInventory.checkOutBook(sNo);
125
126        if (checkedInBook != null) {
127            System.out.println("Book checked in successfully!");
128            // Perform additional operations if needed
129        }
130    }
131
132
133
134    Run | Debug
135    public static void main(String[] args) {
136        library lib = library.getInstance();
137        lib.runLibrary();
138    }
139}
```

```
src > library.java > library > runLibrary()
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
```

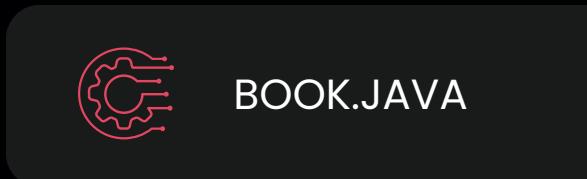
THIS CLASS REPRESENTS THE LIBRARY MANAGEMENT SYSTEM. IT CONTROLS THE FLOW OF THE APPLICATION AND INTERACTS WITH THE BOOKS AND STUDENTS CLASSES.

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.

```
src > library.java > library > checkInBook()
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
```



## BOOK.JAVA

```
students.java 1 student.java 2 library.java 5 books.java 4 book.java 2 X

src > book.java > book > bookName
You, 12 minutes ago | 1 author (You)
1 package library;
2
3 import java.util.Scanner;
4
5 You, 12 minutes ago | 1 author (You)
public class book {
6     private int sNo;
7     private String bookName;      You, 12 minutes ago • menambahkan data
8     private String authorName;
9     private int bookQty;
10    private int bookQtyCopy;
11
12    private book(int sNo, String bookName, String authorName, int bookQty) {
13        this.sNo = sNo;
14        this.bookName = bookName;
15        this.authorName = authorName;
16        this.bookQty = bookQty;
17        this.bookQtyCopy = bookQty;
18    }
19
20    public static book createBook() {
21        Scanner input = new Scanner(System.in);
22
23        System.out.println("Enter Serial No of Book:");
24        int sNo = input.nextInt();
25        input.nextLine();
26        System.out.println("Enter Book Name:");
27        String bookName = input.nextLine();
28        System.out.println("Enter Author Name:");
29        String authorName = input.nextLine();
30        System.out.println("Enter Quantity of Books:");
31        int bookQty = input.nextInt();
32
33        return new book(sNo, bookName, authorName, bookQty);
34    }
35
36    // Getters and setters...
37
38    @Override
39    public String toString() {
40        return "Serial No: " + sNo + "\nBook Name: " + bookName + "\nAuthor Name: " + authorName +
41               "\nAvailable Quantity: " + bookQtyCopy + "\nTotal Quantity: " + bookQty;
42    }
43
44    public int getBookQtyCopy() {
45        // TODO Auto-generated method stub
46        return 0;
47    }
48
49    public int getBookQty() {
50        // TODO Auto-generated method stub
51        return 0;
52    }
53
54    public int getSNo() {
55        // TODO Auto-generated method stub
56        return sNo;
57    }
58
59    public void setBookQty(int i) {
60        // TODO Auto-generated method stub      You, 12 minutes ago • menambahkan data ...
61    }
62
63
64    public String getAuthorName() {
65        // TODO Auto-generated method stub
66        return authorName;
67    }
68
69    public String getBookName() {
70        // TODO Auto-generated method stub
71        return bookName;
72    }
73}
74
```

```
students.java 1 student.java 2 library.java 5 books.java 4 book.java 2 X

src > book.java > book > setBookQty(int)
You, 12 minutes ago | 1 author (You)
34    }
35
36    // Getters and setters...
37
38    @Override
39    public String toString() {
40        return "Serial No: " + sNo + "\nBook Name: " + bookName + "\nAuthor Name: " + authorName +
41               "\nAvailable Quantity: " + bookQtyCopy + "\nTotal Quantity: " + bookQty;
42    }
43
44    public int getBookQtyCopy() {
45        // TODO Auto-generated method stub
46        return 0;
47    }
48
49    public int getBookQty() {
50        // TODO Auto-generated method stub
51        return 0;
52    }
53
54    public int getSNo() {
55        // TODO Auto-generated method stub
56        return sNo;
57    }
58
59    public void setBookQty(int i) {
60        // TODO Auto-generated method stub      You, 12 minutes ago • menambahkan data ...
61    }
62
63
64    public String getAuthorName() {
65        // TODO Auto-generated method stub
66        return authorName;
67    }
68
69    public String getBookName() {
70        // TODO Auto-generated method stub
71        return bookName;
72    }
73}
74
```

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

```

src > books.java > books > books()
44     private boolean compareBookObjects(book b1, book b2) {
45         return b1.getBookName().equalsIgnoreCase(b2.getBookName()) ||
46             b1.getsNo() == b2.getsNo();
47     }

48
49     public void searchBySno() {
50         System.out.println("Enter Serial No of Book:");
51         Scanner input = new Scanner(System.in);
52         int sNo = input.nextInt();
53         boolean found = false;

54         for (int i = 0; i < count; i++) {
55             if (theBooks[i].getsNo() == sNo) {
56                 System.out.println(theBooks[i]);
57                 found = true;
58             }
59         }
60         if (!found) {
61             System.out.println("No book with Serial No " + sNo + " found.");
62         }
63     }

64
65     public void searchByAuthorName() {
66         Scanner input = new Scanner(System.in);
67         System.out.println("Enter Author Name:");
68         String authorName = input.nextLine();
69         boolean found = false;

70         for (int i = 0; i < count; i++) {
71             if (theBooks[i].getAuthorName().equalsIgnoreCase(authorName)) {
72                 System.out.println(theBooks[i]);
73                 found = true;
74             }
75         }
76         if (!found) {
77             System.out.println("No books by " + authorName + " found.");
78         }
79     }

80
81     public void showAllBooks() {
82         if (count == 0) {
83             System.out.println("No books available.");
84             return;
85         }
86         System.out.println("Showing all books:");
87         for (int i = 0; i < count; i++) {
88             System.out.println(theBooks[i]);
89         }
90     }

91
92     public void addBook() {
93         if (count >= 50) {
94             System.out.println("No Space to Add More Books.");
95             return;
96         }
97         book newBook = book.createBook();

98         for (int i = 0; i < count; i++) {
99             if (compareBookObjects(newBook, theBooks[i])) {
100                 System.out.println("Book with the same Serial No or Name already exists.");
101                 return;
102             }
103         }

104         theBooks[count] = newBook;
105         count++;

106         System.out.println("Book added successfully!");
107     }

108
109     private boolean compareBookObjects(book b1, book b2) {
110         return b1.getBookName().equalsIgnoreCase(b2.getBookName()) ||
111             b1.getsNo() == b2.getsNo();
112     }

```

```

src > books.java > books > books()
44     private boolean compareBookObjects(book b1, book b2) {
45         return b1.getBookName().equalsIgnoreCase(b2.getBookName()) ||
46             b1.getsNo() == b2.getsNo();
47     }

48
49     public void searchBySno() {
50         System.out.println("Enter Serial No of Book:");
51         Scanner input = new Scanner(System.in);
52         int sNo = input.nextInt();
53         boolean found = false;

54         for (int i = 0; i < count; i++) {
55             if (theBooks[i].getsNo() == sNo) {
56                 System.out.println(theBooks[i]);
57                 found = true;
58             }
59         }
60         if (!found) {
61             System.out.println("No book with Serial No " + sNo + " found.");
62         }
63     }

64
65     public void searchByAuthorName() {
66         Scanner input = new Scanner(System.in);
67         System.out.println("Enter Author Name:");
68         String authorName = input.nextLine();
69         boolean found = false;

70         for (int i = 0; i < count; i++) {
71             if (theBooks[i].getAuthorName().equalsIgnoreCase(authorName)) {
72                 System.out.println(theBooks[i]);
73                 found = true;
74             }
75         }
76         if (!found) {
77             System.out.println("No books by " + authorName + " found.");
78         }
79     }

80
81     public void showAllBooks() {
82         if (count == 0) {
83             System.out.println("No books available.");
84             return;
85         }
86         System.out.println("Showing all books:");
87         for (int i = 0; i < count; i++) {
88             System.out.println(theBooks[i]);
89             System.out.println("-----");
90         }
91     }

92
93     public void upgradeBookQty() {
94         System.out.println("Enter Serial No of Book:");
95         Scanner input = new Scanner(System.in);
96         int sNo = input.nextInt();

97         for (int i = 0; i < count; i++) {
98             if (theBooks[i].getsNo() == sNo) {
99                 System.out.println("Enter No of Books to be Added:");
100                int addingQty = input.nextInt();
101                theBooks[i].setBookQty(theBooks[i].getBookQty() + addingQty);
102                theBooks[i].setBookQty(theBooks[i].getBookQtyCopy() + addingQty);
103                System.out.println("Quantity upgraded successfully!");
104            }
105        }

106        System.out.println("No book with Serial No " + sNo + " found.");
107    }

108
109    public book checkOutBook(int sNo) {
110        // TODO Auto-generated method stub
111        return null;
112    }

```

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.





## STUDENT.JAVA

```
student.java 1 student.java 2 X library.java 5 books.java 4 book.java 2
src > student.java > {} library
You, 15 minutes ago | 1 author (You)
1 package library; You, 15 minutes ago • menambahkan data ...
2
3 import java.util.Scanner;
4
5 You, 15 minutes ago | 1 author (You)
6 public class student {
7     private String studentName;
8     private String regNum;
9
10    private student(String studentName, String regNum) {
11        this.studentName = studentName;
12        this.regNum = regNum;
13    }
14
15    public static student createStudent() {
16        Scanner input = new Scanner(System.in);
17
18        System.out.println("Enter Student Name:");
19        String studentName = input.nextLine();
20        System.out.println("Enter Reg Number:");
21        String regNum = input.nextLine();
22
23        return new student(studentName, regNum);
24    }
25
26    // Getters and setters...
27
28    @Override
29    public String toString() {
30        return "Student Name: " + studentName + "\nRegistration Number: " + regNum;
31    }
32}
```

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 X studentjava 2 libraryjava 5 books.java 4 book.java 2
src > students.java > ↗ students
You, 16 minutes ago | 1 author (You)
1 package library;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 You, 16 minutes ago | 1 author (You)
7 public class students {
8     private List<student> studentList;
9
10    private static students instance;
11
12    private students() {
13        studentList = new ArrayList<>();
14    }
15
16    You, 16 minutes ago • menambahkan data
17    public static students getInstance() {
18        if (instance == null) {
19            instance = new students();
20        }
21        return instance;
22    }
23
24    public void addStudent() {
25        student newStudent = student.createStudent();
26        studentList.add(newStudent);
27        System.out.println("Student registered successfully!");
28    }
29
30    public void showAllStudents() {
31        if (studentList.isEmpty()) {
32            System.out.println("No students registered.");
33            return;
34        }
35
36        System.out.println("Showing all registered students:");
37        for (student s : studentList) {
38            System.out.println(s);
39            System.out.println("-----");
40        }
41    }
}
```

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