**Exercise 6: Library Management System**

**Scenario:**

You are developing a library management system where users can search for books by title or author.

**Steps:**

1. **Understand Search Algorithms:**
   * Explain linear search and binary search algorithms.
2. **Setup:**
   * Create a class **Book** with attributes like **bookId**, **title**, and **author**.
3. **Implementation:**
   * Implement linear search to find books by title.
   * Implement binary search to find books by title (assuming the list is sorted).
4. **Analysis:**
   * Compare the time complexity of linear and binary search.
   * Discuss when to use each algorithm based on the data set size and order.

**1: Understand Search Algorithms**

**Linear Search**

* Goes through each element one by one.
* **Best case**: O(1) (first element)
* **Worst case**: O(n)
* Works on **unsorted data**.

**Binary Search**

* Requires the list to be **sorted**.
* Repeatedly divides the search space in half.
* **Best case**: O(1)
* **Worst case**: O(log n)
* **Much** **faster for large sorted data sets**

**CODE:-** import java.util.\*;

class Book {

    int bookId;

    String title;

    String author;

    public Book(int bookId, String title, String author) {

        this.bookId = bookId;

        this.title = title.toLowerCase();

        this.author = author;

    }

    public void display() {

        System.out.println("ID: " + bookId + ", Title: " + title + ", Author: " + author);

    }

}

public class LibraryManagementSystem {

    private List<Book> books = new ArrayList<>();

    // Add book to the list

    public void addBook(Book book) {

        books.add(book);

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

    }

    // Linear search by title

    public void linearSearch(String title) {

        boolean found = false;

        for (Book b : books) {

            if (b.title.equalsIgnoreCase(title)) {

                b.display();

                found = true;

            }

        }

        if (!found) {

            System.out.println("Book not found.");

        }

    }

    // Binary search by title (list must be sorted)

    public void binarySearch(String title) {

        // Sort books by title before searching

        books.sort(Comparator.comparing(b -> b.title));

        int low = 0, high = books.size() - 1;

        boolean found = false;

        while (low <= high) {

            int mid = (low + high) / 2;

            Book midBook = books.get(mid);

            int comparison = midBook.title.compareToIgnoreCase(title);

            if (comparison == 0) {

                midBook.display();

                found = true;

                break;

            } else if (comparison < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        if (!found) {

            System.out.println("Book not found.");

        }

    }

    // View all books

    public void displayAllBooks() {

        if (books.isEmpty()) {

            System.out.println("Library is empty.");

        } else {

            for (Book b : books) {

                b.display();

            }

        }

    }

    // Main method

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        LibraryManagementSystem lib = new LibraryManagementSystem();

        while (true) {

            System.out.println("\n1. Add Book\n2. Linear Search by Title\n3. Binary Search by Title\n4. View All Books\n5. Exit");

            System.out.print("Choose an option: ");

            int choice = sc.nextInt();

            sc.nextLine(); // consume newline

            switch (choice) {

                case 1:

                    System.out.print("Enter Book ID: ");

                    int id = sc.nextInt();

                    sc.nextLine();

                    System.out.print("Enter Title: ");

                    String title = sc.nextLine();

                    System.out.print("Enter Author: ");

                    String author = sc.nextLine();

                    lib.addBook(new Book(id, title, author));

                    break;

                case 2:

                    System.out.print("Enter Title to search (Linear): ");

                    String linTitle = sc.nextLine();

                    lib.linearSearch(linTitle);

                    break;

                case 3:

                    System.out.print("Enter Title to search (Binary): ");

                    String binTitle = sc.nextLine();

                    lib.binarySearch(binTitle);

                    break;

                case 4:

                    lib.displayAllBooks();

                    break;

                case 5:

                    System.out.println("Exiting...");

                    sc.close();

                    return;

                default:

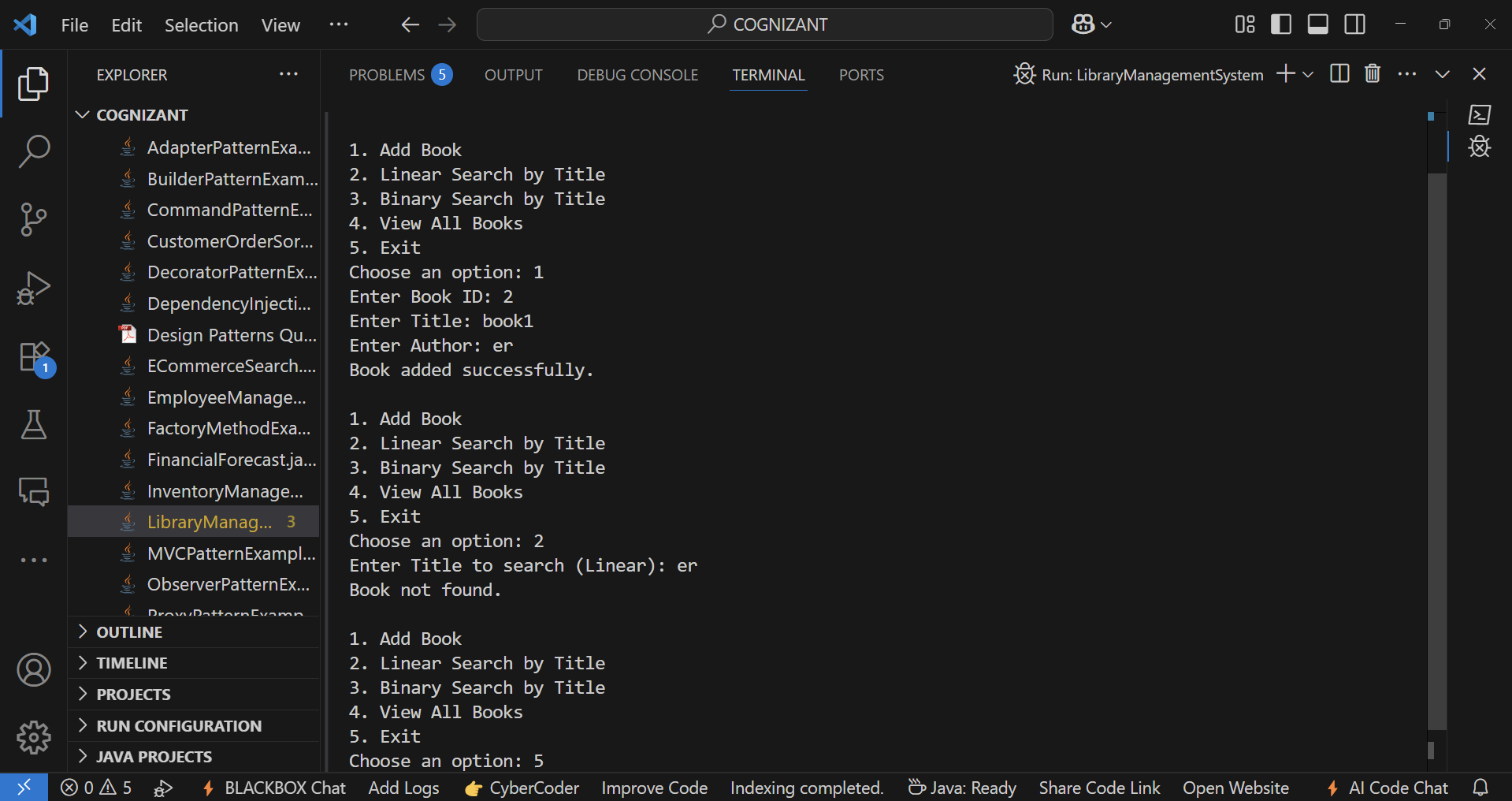
                    System.out.println("Invalid option.");

            }

        }

    }

}

**OUTPUT:-**

**4: Analysis:-**

Time Complexity:

| **Operation** | **Time Complexity** |
| --- | --- |
| Linear Search | O(n) |
| Binary Search | O(log n) |
| Sorting (required before binary search) | O(n log n) |

When to Use What:

| **Condition** | **Recommended Algorithm** |
| --- | --- |
| Data is unsorted | Linear Search |
| Data is sorted and large | Binary Search |
| Few elements or one-time search | Linear Search |
| Many repeated searches on sorted data | Binary Search |