**Exercise 6: Library Management System**

**Scenario:**

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

**1. Understand Search Algorithms**

**Linear Search**:

* **Description**: Linear search involves checking each element in a list one by one until the desired element is found or the end of the list is reached.
* **Time Complexity**: O(n), where n is the number of elements in the list. It’s efficient for small or unsorted lists.

**Binary Search**:

* **Description**: Binary search works on sorted lists by repeatedly dividing the search interval in half. It compares the target value to the middle element and narrows down the search to either the left or right half.
* **Time Complexity**: O(log n), where n is the number of elements. It’s efficient for large, sorted lists.

**2. Setup**

class Book {

int bookId;

String title;

String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

@Override

public String toString() {

return "ID: " + bookId + ", Title: " + title + ", Author: " + author;

}

}

**3. Implementation**

import java.util.Arrays;

public class LibraryManagementSystem {

// Linear search method

public static Book linearSearch(Book[] books, String title) {

for (Book book : books) {

if (book.title.equalsIgnoreCase(title)) {

return book;

}

}

return null; // Book not found

}

// Binary search method (requires sorted array)

public static Book binarySearch(Book[] books, String title) {

int left = 0;

int right = books.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

int comparison = books[mid].title.compareToIgnoreCase(title);

if (comparison == 0) {

return books[mid];

} else if (comparison < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null; // Book not found

}

public static void main(String[] args) {

Book[] books = {

new Book(1, "The Great Gatsby", "F. Scott Fitzgerald"),

new Book(2, "To Kill a Mockingbird", "Harper Lee"),

new Book(3, "1984", "George Orwell")

};

// Sort books by title for binary search

Arrays.sort(books, (b1, b2) -> b1.title.compareToIgnoreCase(b2.title));

System.out.println("Linear Search:");

Book result = linearSearch(books, "1984");

System.out.println(result != null ? result : "Book not found");

System.out.println("\nBinary Search:");

result = binarySearch(books, "1984");

System.out.println(result != null ? result : "Book not found");

}

}

**4. Analysis**

**Time Complexity Comparison**:

* **Linear Search**: O(n) (iterates through the list sequentially).
* **Binary Search**: O(log n) (divides the search range in half each step, requires sorted data).

**When to Use Each Algorithm**:

* **Linear Search**: Use when the dataset is small or unsorted, as it doesn't require sorting.
* **Binary Search**: Use for large, sorted datasets where the performance gain of O(log n) is significant compared to O(n).