**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:**
* Linear search is a straightforward algorithm that checks each element of a list sequentially until the desired element is found or the list ends.
* Time Complexity: O(n), where n is the number of elements in the list.
* **Binary Search:**
* Binary search is a more efficient algorithm that only works on sorted lists. It repeatedly divides the list in half, comparing the target value to the middle element and discarding the half where the target cannot be.
* Time Complexity: O(log n), where n is the number of elements in the list.

**2. Setup**

//Creating the Book Class

public class Book {

private int bookId;

private String title;

private String author;

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

this.bookId = bookId;

this.title = title;

this.author = author;

}

public int getBookId() {

return bookId;

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

@Override

public String toString() {

return "Book{" +

"bookId=" + bookId +

", title='" + title + '\'' +

", author='" + author + '\'' +

'}';

}

}

**3. Implementation**

//Linear Search Implementation

import java.util.List;

public class Library {

public Book linearSearchByTitle(List<Book> books, String title) {

for (Book book : books) {

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

return book;

}

}

return null;

}

}

//Binary Search Implementation

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

public class Library {

// Linear search

public Book linearSearchByTitle(List<Book> books, String title) {

for (Book book : books) {

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

return book;

}

}

return null;

}

// Binary search

public Book binarySearchByTitle(List<Book> books, String title) {

// Ensuring the list is sorted

Collections.sort(books, Comparator.comparing(Book::getTitle));

int left = 0;

int right = books.size() - 1;

while (left <= right) {

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

Book midBook = books.get(mid);

int cmp = midBook.getTitle().compareToIgnoreCase(title);

if (cmp == 0) {

return midBook;

} else if (cmp < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

}

//main class too call the methods

import java.util.ArrayList;

import java.util.List;

public class Main {

public static void main(String[] args) {

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

books.add(new Book(1, "The Catcher in the Rye", "J.D. Salinger"));

books.add(new Book(2, "To Kill a Mockingbird", "Harper Lee"));

books.add(new Book(3, "1984", "George Orwell"));

Library library = new Library();

//calling linearSearchByTitle

Book foundBookLinear = library.linearSearchByTitle(books, "1984");

System.out.println("Linear Search Result: " + foundBookLinear);

//calling binarySearchByTitle method

Book foundBookBinary = library.binarySearchByTitle(books, "1984");

System.out.println("Binary Search Result: " + foundBookBinary);

}

}

**4. Analysis**

* **Time Complexity:**
* Linear Search: O(n)
* Binary Search: O(log n)
* **Usage:**
* Linear Search:
  + Used when the dataset is small or unsorted.
  + Simple to implement and does not require the list to be sorted.
* Binary Search:
  + Used when the dataset is large and sorted.
  + More efficient than linear search for large datasets but requires initial sorting.