**Exercise 6: Library Management System**

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;

this.author = author;

}

public String toString() {

return "BookID: " + bookId + ", Title: '" + title + "', Author: " + author;

}

}

public class LibraryManagementSystem {

// Linear Search: O(n)

public static List<Book> linearSearch(List<Book> books, String targetTitle) {

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

for (Book book : books) {

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

result.add(book);

}

}

return result;

}

// Binary Search: O(log n), assumes books are sorted by title

public static List<Book> binarySearch(List<Book> books, String targetTitle) {

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

int low = 0;

int high = books.size() - 1;

targetTitle = targetTitle.toLowerCase();

while (low <= high) {

int mid = (low + high) / 2;

String midTitle = books.get(mid).title.toLowerCase();

if (midTitle.equals(targetTitle)) {

result.add(books.get(mid));

int left = mid - 1;

while (left >= 0 && books.get(left).title.equalsIgnoreCase(targetTitle)) {

result.add(0, books.get(left));

left--;

}

// Look right for duplicates

int right = mid + 1;

while (right < books.size() && books.get(right).title.equalsIgnoreCase(targetTitle)) {

result.add(books.get(right));

right++;

}

return result;

} else if (midTitle.compareTo(targetTitle) < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return result;

}

public static void main(String[] args) {

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

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

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

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

books.add(new Book(4, "The Catcher in the Rye", "J.D. Salinger"));

books.add(new Book(5, "The Hobbit", "J.R.R. Tolkien"));

books.add(new Book(6, "1984", "George Orwell"));

// Sort for binary search

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

// Linear Search

System.out.println("Linear Search Results for '1984':");

List<Book> linearResults = linearSearch(books, "1984");

for (Book b : linearResults) {

System.out.println(b);

}

// Binary Search

System.out.println("\nBinary Search Results for '1984':");

List<Book> binaryResults = binarySearch(books, "1984");

for (Book b : binaryResults) {

System.out.println(b);

}

}

}

Analysis:

1. Linear Search:

- Time Complexity: O(n)

- Best for small or unsorted lists.

- Easy to implement.

2. Binary Search:

- Time Complexity: O(log n)

- Requires sorted list.

- Much faster for large datasets.

When to Use:

- Use Linear Search for small or infrequent searches on unsorted data.

- Use Binary Search for frequent searches on large, sorted data.

