**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.**

**Solution:**

**LibraryManagementSystem.java**

import java.util.Arrays;

import java.util.Scanner;

class Book implements Comparable<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 String toString() {

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

}

@Override

public int compareTo(Book other) {

return this.title.compareTo(other.title);

}

}

public class LibraryManagementSystem {

static Scanner sc = new Scanner(System.in);

static Book[] books;

public static void main(String[] args) {

System.out.print("Enter number of books: ");

int n = sc.nextInt();

sc.nextLine();

books = new Book[n];

for (int i = 0; i < n; i++) {

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();

books[i] = new Book(id, title, author);

}

System.out.print("\nEnter title to search: ");

String searchTitle = sc.nextLine().toLowerCase();

long startLinear = System.nanoTime();

linearSearch(searchTitle);

long endLinear = System.nanoTime();

System.out.println("Linear Search Time: " + (endLinear - startLinear) + " nanoseconds");

Arrays.sort(books);

long startBinary = System.nanoTime();

binarySearch(searchTitle);

long endBinary = System.nanoTime();

System.out.println("Binary Search Time: " + (endBinary - startBinary) + " nanoseconds");

printAnalysis();

}

static void linearSearch(String title) {

boolean found = false;

for (Book book : books) {

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

System.out.println("Found (Linear): " + book);

found = true;

}

}

if (!found) {

System.out.println("Book not found in Linear Search.");

}

}

static void binarySearch(String title) {

int left = 0, right = books.length - 1;

boolean found = false;

while (left <= right) {

int mid = (left + right) / 2;

int cmp = books[mid].title.compareTo(title);

if (cmp == 0) {

System.out.println("Found (Binary): " + books[mid]);

found = true;

int i = mid - 1;

while (i >= 0 && books[i].title.equals(title)) {

System.out.println("Found (Binary): " + books[i--]);

}

i = mid + 1;

while (i < books.length && books[i].title.equals(title)) {

System.out.println("Found (Binary): " + books[i++]);

}

break;

} else if (cmp < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

if (!found) {

System.out.println("Book not found in Binary Search.");

}

}

static void printAnalysis() {

System.out.println("\n--- Time Complexity Analysis ---");

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

System.out.println(" - Best Case: O(1)");

System.out.println(" - Worst Case: O(n)");

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

System.out.println(" - Best Case: O(1)");

System.out.println(" - Worst Case: O(log n) [requires sorted array]");

System.out.println("\n--- When to Use Each ---");

System.out.println("• Use Linear Search when the data is unsorted or small.");

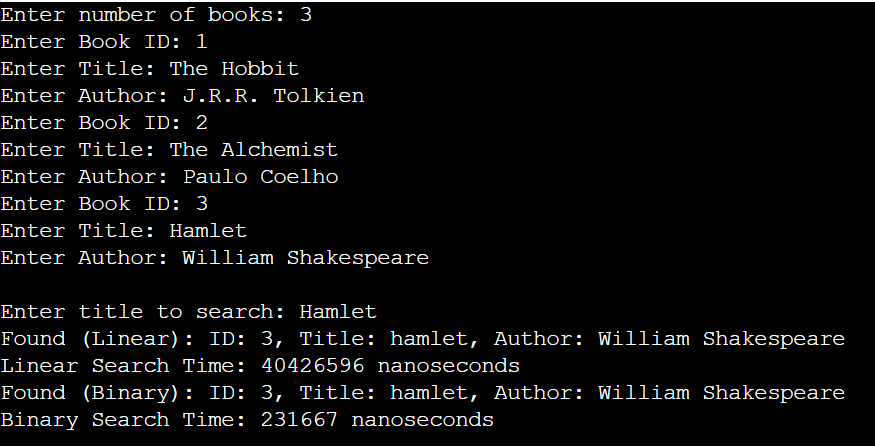
System.out.println("• Use Binary Search for large, sorted datasets for faster lookup.");

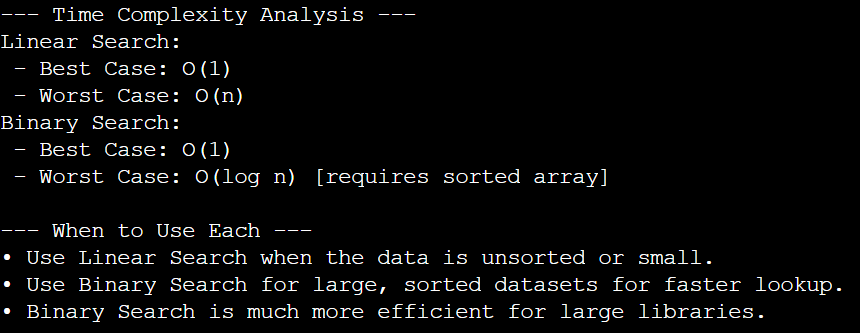
System.out.println("• Binary Search is much more efficient for large libraries.");

}

}

**Output:**

****

****