**Exercise 2: E-commerce Platform Search Function**

**Code :**

import java.util.Arrays;  
import java.util.Comparator;  
  
class Book {  
 int bookId;  
 String title;  
 String genre;  
  
 public Book(int bookId, String title, String genre) {  
 this.bookId = bookId;  
 this.title = title;  
 this.genre = genre;  
 }  
  
 @Override  
 public String toString() {  
 return "[" + bookId + ", " + title + ", " + genre + "]";  
 }  
}  
  
public class LibrarySearch {  
 public static Book linearSearch(Book[] books, String targetTitle) {  
 for (Book book : books) {  
 if (book.title.equalsIgnoreCase(targetTitle)) {  
 return book;  
 }  
 }  
 return null;  
 }  
  
 public static Book binarySearch(Book[] books, String targetTitle) {  
 Arrays.*sort*(books, Comparator.*comparing*(b -> b.title.toLowerCase()));  
 int left = 0, right = books.length - 1;  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
 int cmp = targetTitle.compareToIgnoreCase(books[mid].title);  
 if (cmp == 0) return books[mid];  
 else if (cmp < 0) right = mid - 1;  
 else left = mid + 1;  
 }  
 return null;  
 }  
  
 public static void main(String[] args) {  
 Book[] books = {  
 new Book(201, "The Alchemist", "Fiction"),  
 new Book(202, "Data Structures", "Education"),  
 new Book(203, "Atomic Habits", "Self-help"),  
 new Book(204, "Clean Code", "Programming"),  
 new Book(205, "Harry Potter", "Fantasy")  
 };  
  
 String searchTitle = "Atomic Habits";  
 Book result1 = *linearSearch*(books, searchTitle);  
 System.*out*.println("Linear Search Result: " + (result1 != null ? result1 : "Book not found"));  
  
 Book result2 = *binarySearch*(books, searchTitle);  
 System.*out*.println("Binary Search Result: " + (result2 != null ? result2 : "Book not found"));  
 }  
}

### **1. **What is Big O Notation?****

**Big O notation tells us how fast or slow an algorithm is as the input size grows.  
It focuses on the number of steps, not exact time.**

**Example:**

* **O(n): goes through each item (like linear search)**
* **O(log n): cuts data in half each step (like binary search)**

### 2. ****Best, Average, Worst Cases for Search:****

| Search Type | Best Case | Average Case | Worst Case |
| --- | --- | --- | --- |
| Linear Search | O(1) (found early) | O(n/2) ≈ O(n) | O(n) (last or not found) |
| Binary Search | O(1) (middle) | O(log n) | O(log n) |

### 3. ****When to Use What?****

* **Use linear search if:**
  + **List is small**
  + **List is unsorted**
* **Use binary search if:**
  + **List is sorted**
  + **You want better performance (especially with big data)**

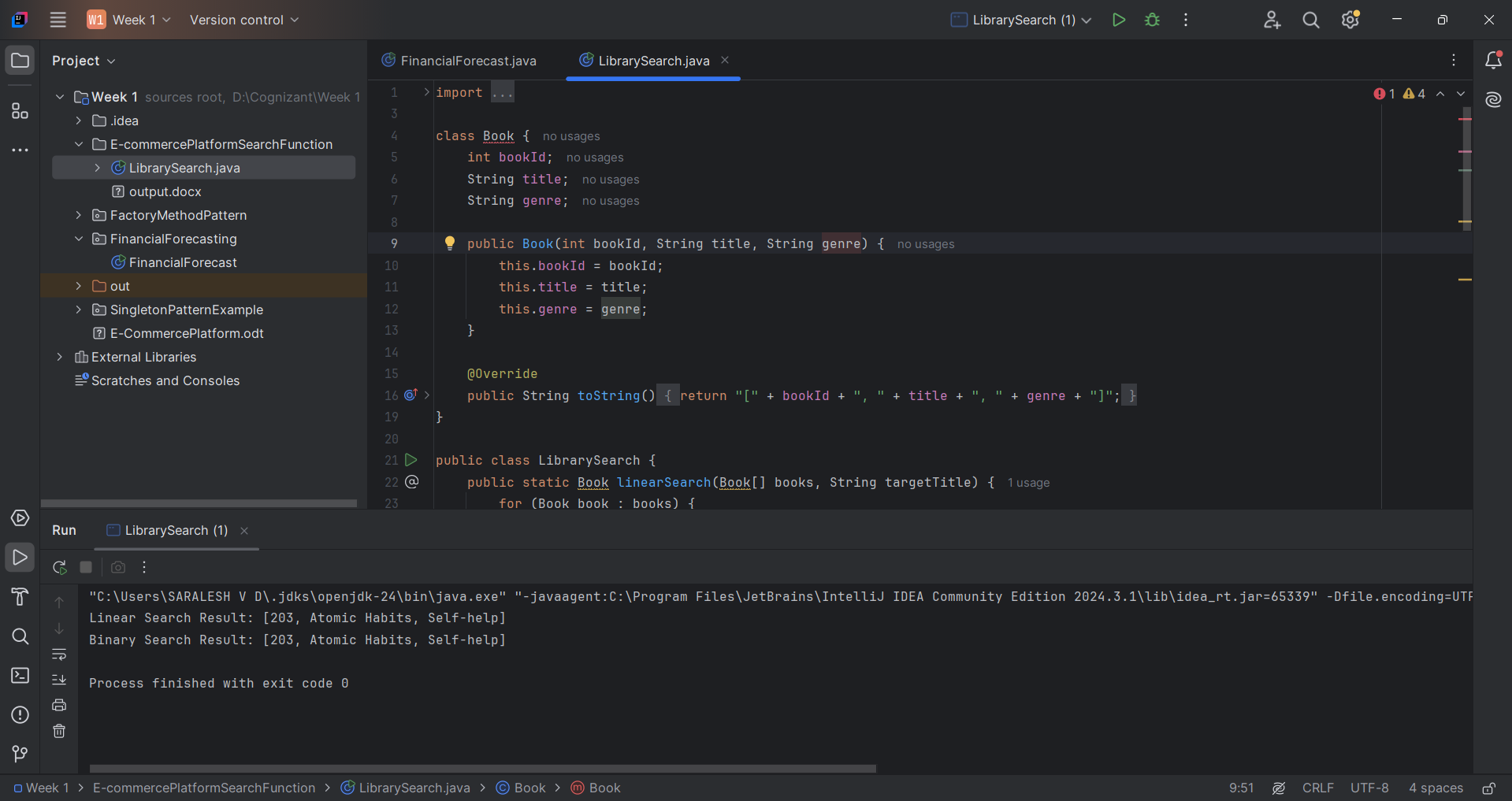
### 4. ****Which is Better?****

**Binary search is better for large, sorted data because:**

* **It is much faster (O(log n) instead of O(n))**
* **But it needs sorted data first**

**If sorting is expensive or the list is always changing, linear search may be simpler.**

**Output :**

****