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

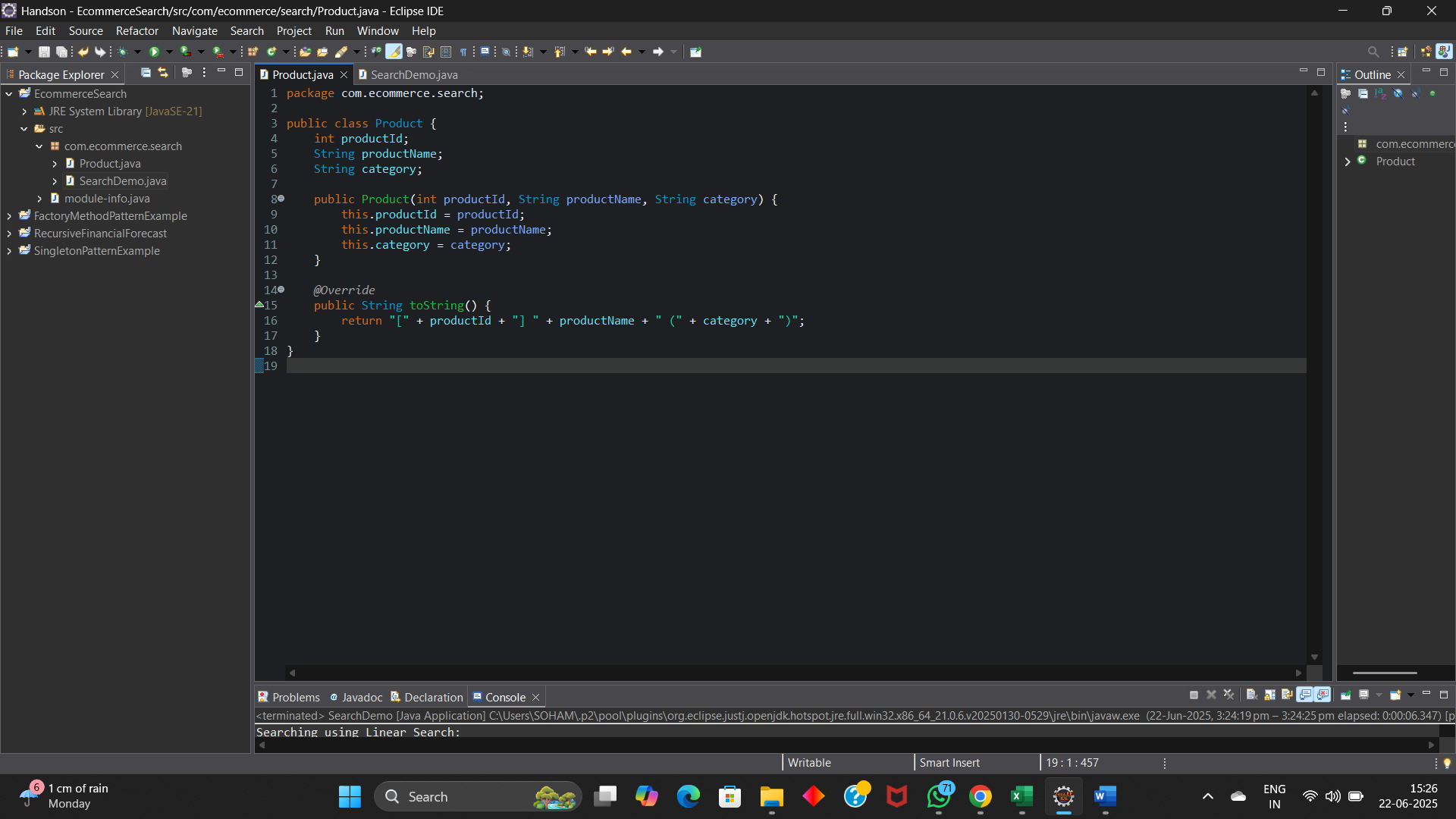
**Scenario:**

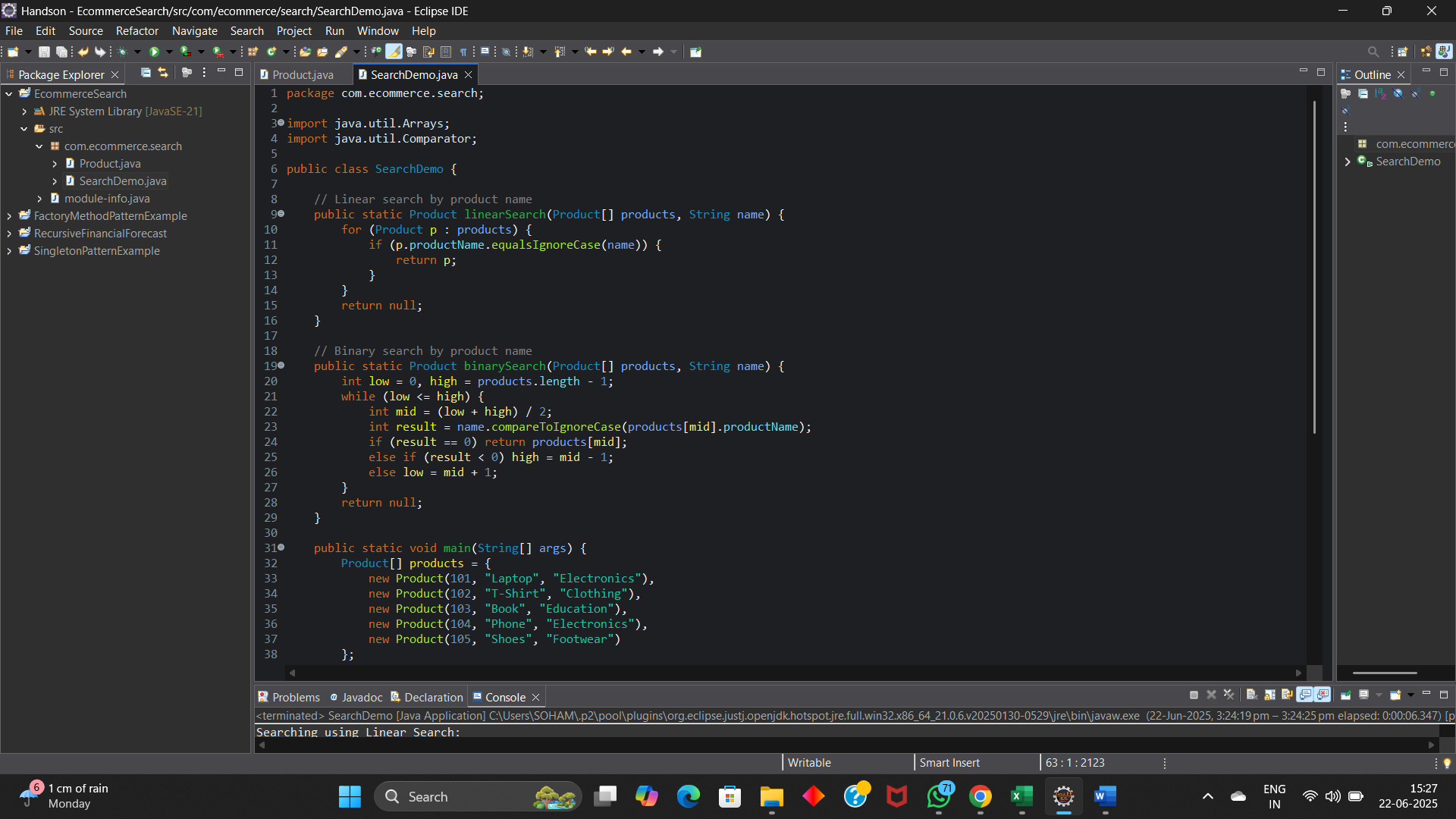
You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

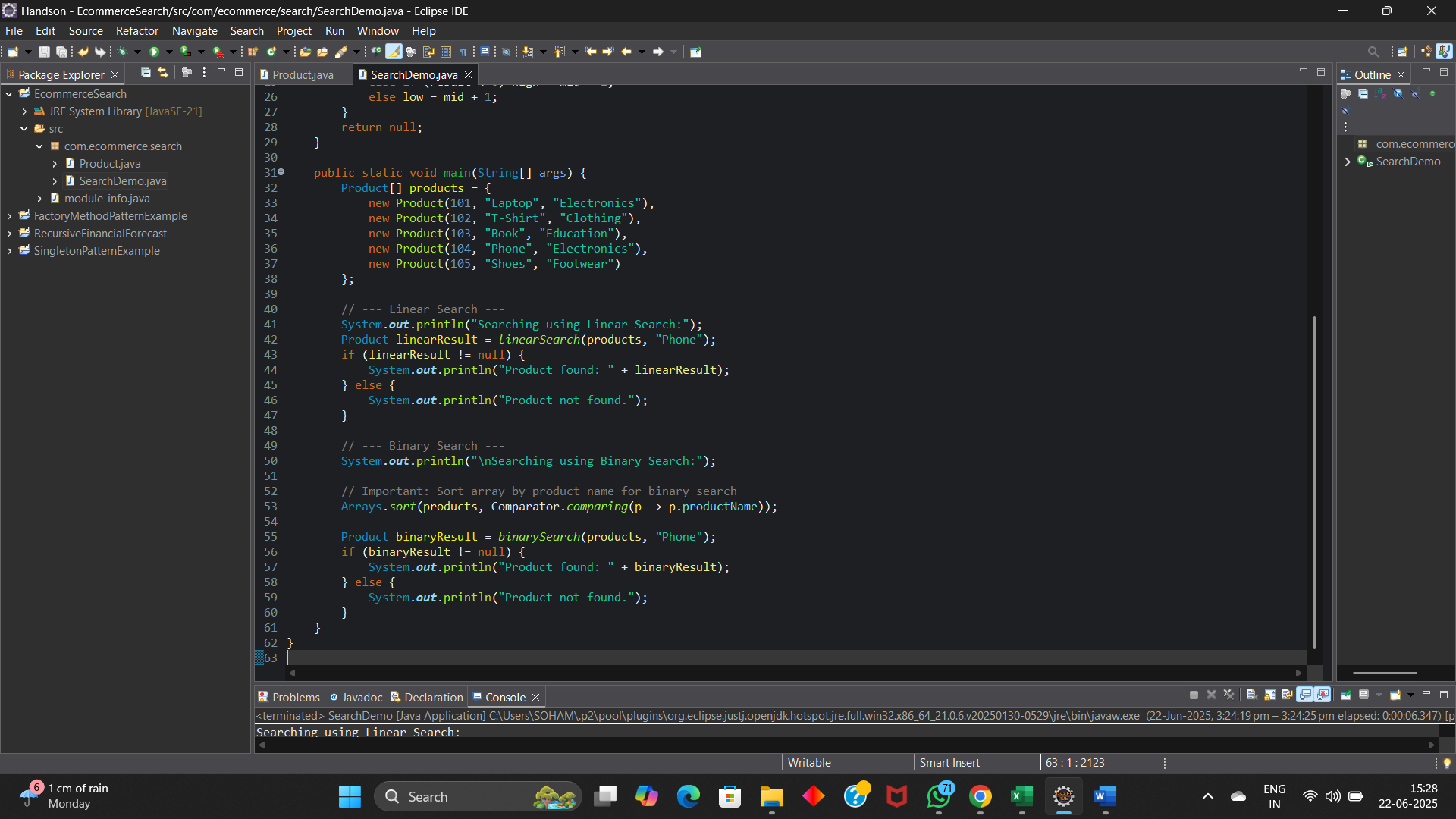
**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

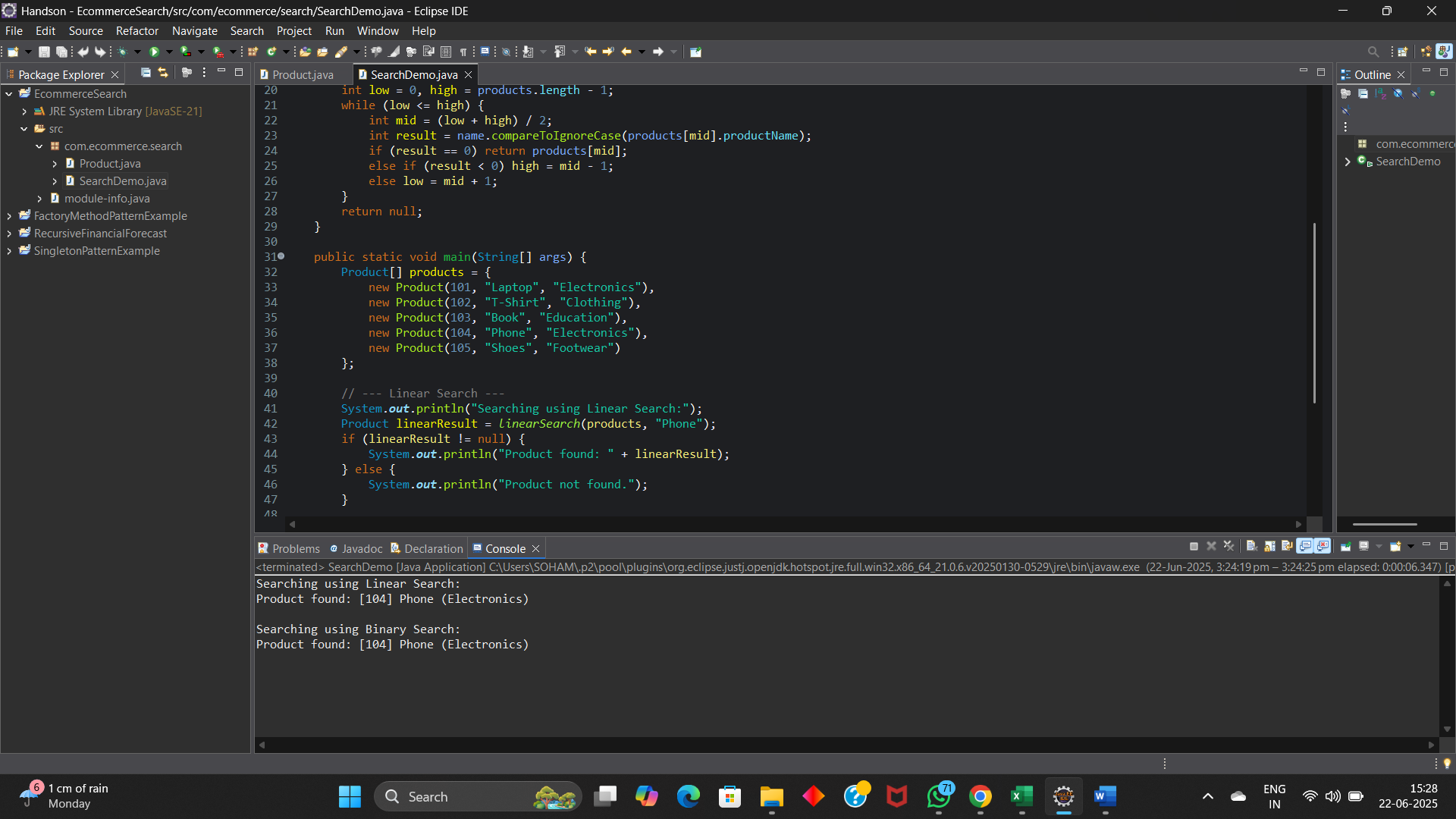
**SOLUTION:**







**OUTPUT:**



**BIG O NOTATION:**

Big O notation is used to describe how fast or slow an algorithm runs based on the size of the input. It shows how the time or space needed by an algorithm grows when the input becomes larger.

It helps us compare different algorithms by giving a general idea of their performance. For example, an algorithm with **O(n)** means the time increases linearly with input size, while **O(log n)** is much faster for large inputs.

Big O focuses on the most important part how an algorithm behaves when the data grows big. This helps us choose the best and most efficient algorithm for a task.

**BEST CASE, AVERAGE CASE AND WORST CASE SCENARIOS:**

* Best Case happens when the item is found right away. For example, in linear search, it’s at the start (O(1)). In binary search, it's in the middle (O(1)).
* Average Case is the general case. In linear search, we may check half the list, so it's O(n). For binary search, since we keep dividing the list, it's O(log n).
* Worst Case is when the item is at the end or not present. In linear search, that’s O(n). In binary search, even if not found, we divide until the list ends, so it’s still O(log n).

**TIME COMPLEXITY:**

**Linear Search** : Best Case– O(1)

Worst Case-O(n)

**Binary Search** : Best Case– O(1)

Worst Case-O(logn)

**Which algorithm is more suitable for my platform and why?**

Binary search is more suitable for an e-commerce platform because it's much faster (O(log n)) than linear search, especially when there are many products. As long as the product list is sorted, binary search gives better performance and quicker results for users.