**Big O Notation -**

**Big O notation** is a mathematical representation used to describe the *upper bound* of an algorithm's time or space complexity as the input size grows. It helps estimate performance and scalability.

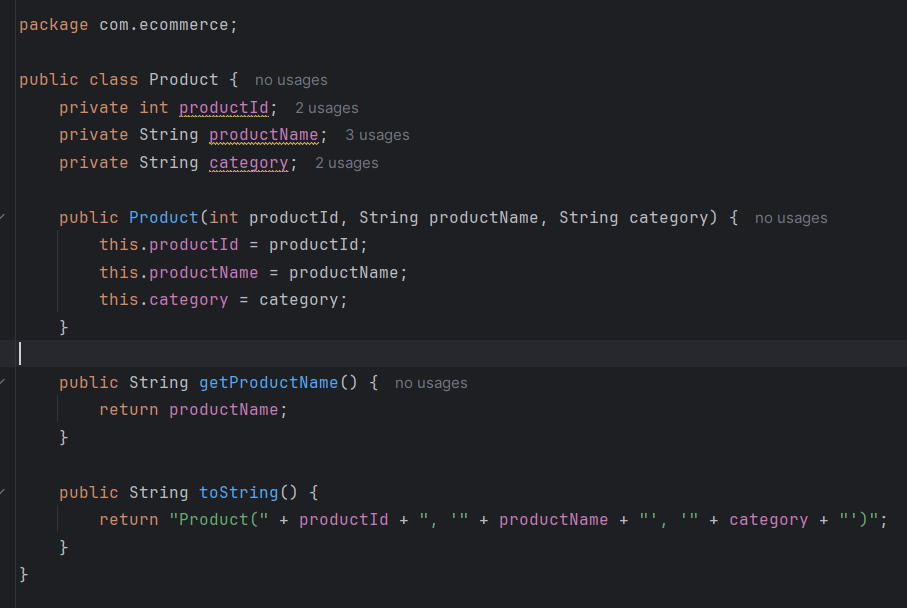
* **Why it's useful**: It abstracts away hardware and implementation details to focus on the efficiency of the algorithm.
* Common complexities:
  + **O(1)**: Constant time
  + **O(log n)**: Logarithmic time (e.g., binary search)
  + **O(n)**: Linear time (e.g., linear search)
  + **O(n log n)**: Log-linear time (e.g., efficient sorts)
  + **O(n²)**: Quadratic time (e.g., nested loops)

**Search Scenarios**

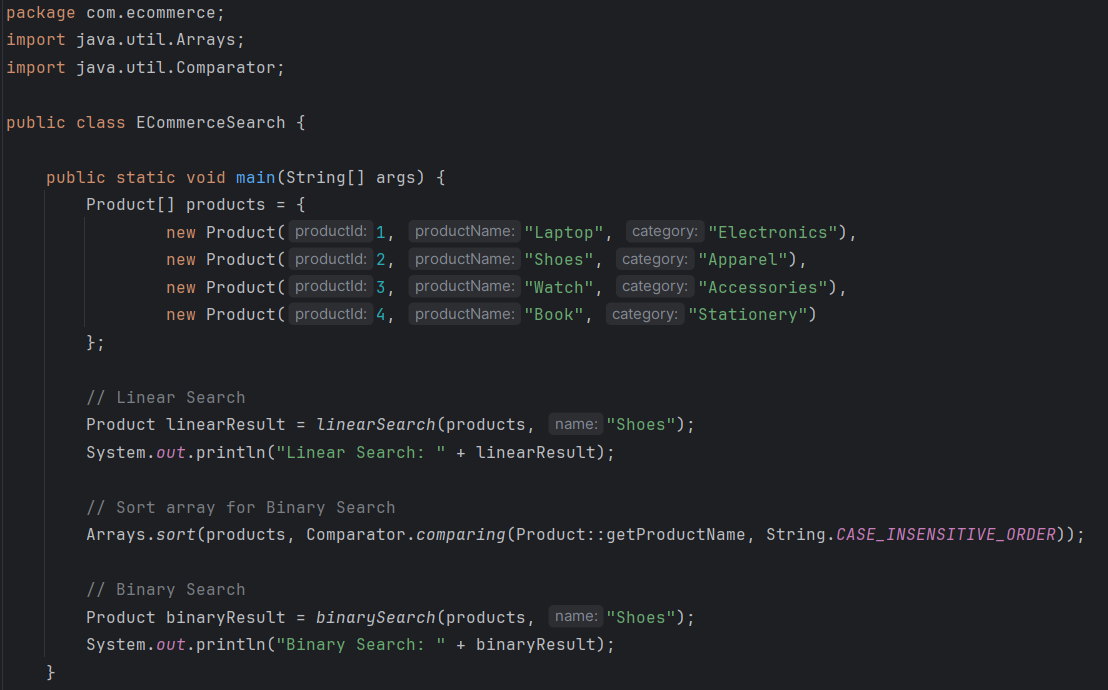
* **Best case**: The search target is found at the beginning (e.g., first item in linear search).
* **Average case**: The target is somewhere in the middle.
* **Worst case**: The item is at the end or not present at all.

Ecommerce Search Platform –

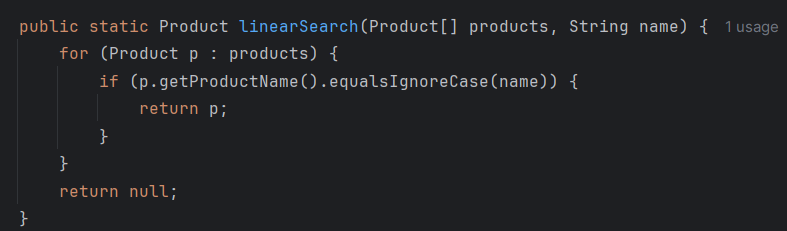
Class - Products.java



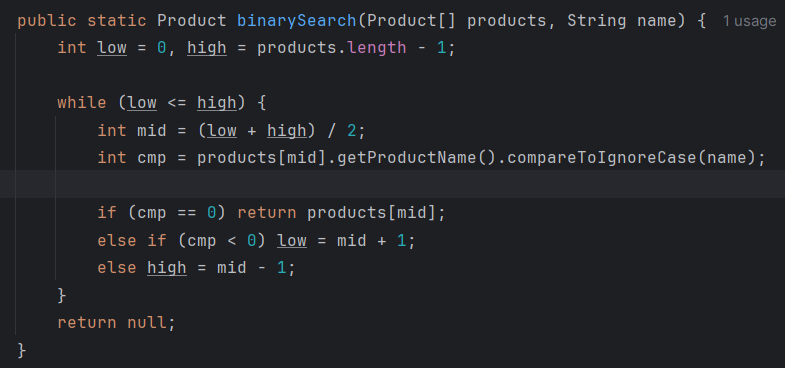
Class - ECommerceSearch.java



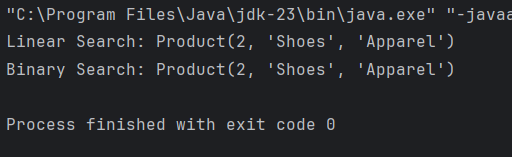
Linear Search Products –



Binary Search Products –



Output –



# **Time Complexity Comparison-**

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Sorted Required?** |
| --- | --- | --- | --- | --- |
| **Linear Search** | O(1) | O(n) | O(n) | No |
| **Binary Search** | O(1) | O(log n) | O(log n) | Yes |

# Which Algorithm is Better for ECommerce Platform?

**Binary Search is more suitable when:**

* The product list is **sorted by product name**.
* You are doing **many frequent searches**.
* Performance and speed are critical (e.g., large catalog with thousands of items).

**Linear Search is acceptable when:**

* The list is **unsorted** and small.
* You **don’t want to sort** the list before searching.
* You need a **quick, simple search** without setup.