**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:**

**1.** Big O notation describes the performance or complexity of an algorithm in terms of input size (n). It expresses how the runtime or space requirements grow as the input size increases, helping developers:

* Analyze scalability.
* Compare different algorithms.
* Choose the most efficient one for large datasets.

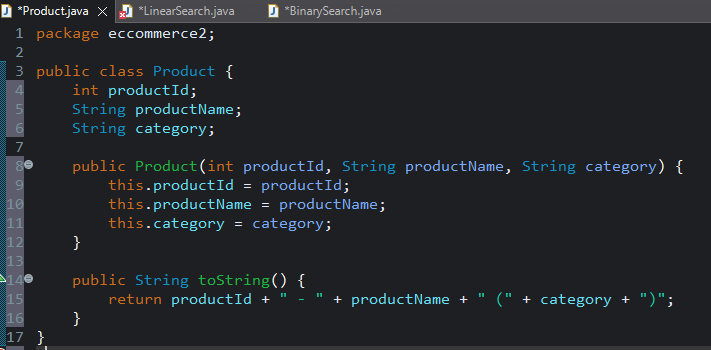
| Notation | Description | Example Use Case |
| --- | --- | --- |
| O(1) | Constant time | Accessing an array element |
| O(n) | Linear time | Linear search |
| O(log n) | Logarithmic time | Binary search |
| O(n²) | Quadratic time | Bubble sort |

**2.**

| Search Type | Best Case | Average Case | Worst Case |
| --- | --- | --- | --- |
| Linear Search | Element at first position | Element in middle | Element not found or last |
| Binary Search | Element at mid | Depends on distribution | Element not found |

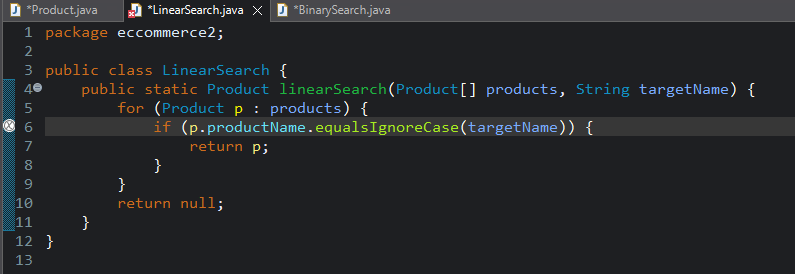
**2. Setup: Product Class:**

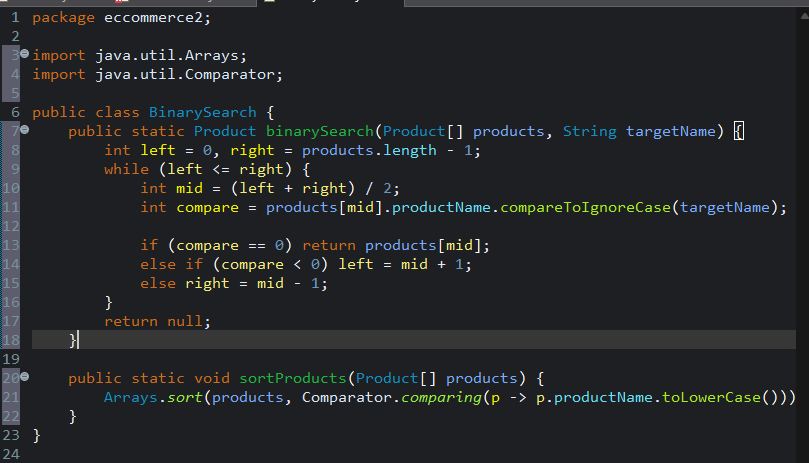
create a basic Product class in Java



**3. Implementation**

Linear Search Algorithm:



Binary Search Algorithm:

**4. Analysis**

**Time Complexity Comparison:**

| **Algorithm** | **Time Complexity (Best)** | **Time Complexity (Average)** | **Time Complexity (Worst)** |
| --- | --- | --- | --- |
| **Linear Search** | O(1) | O(n/2) ≈ O(n) | O(n) |
| **Binary Search** | O(1) | O(log n) | O(log n) |

**Which Is Better?**

| **Criteria** | **Linear Search** | **Binary Search** |
| --- | --- | --- |
| **Requires Sorting** | No | Yes |
| **Performance** | Slower on large datasets | Much faster (logarithmic time) |
| **Flexibility** | Works on any unsorted array | Only works on sorted data |
| **Best Use Case** | Small lists or frequent updates | Large static or infrequently updated lists |

**Suitable Algorithm for E-commerce:**

For **large e-commerce platforms**:

* **Binary Search** is more efficient due to logarithmic time complexity.
* But sorting must be ensured — either via preprocessing or by using sorted data structures (like TreeMap or database indexing).

If the product list is **small or constantly changing**, **Linear Search** may be sufficient.