***Skill – Algorithms\_Data Structures***

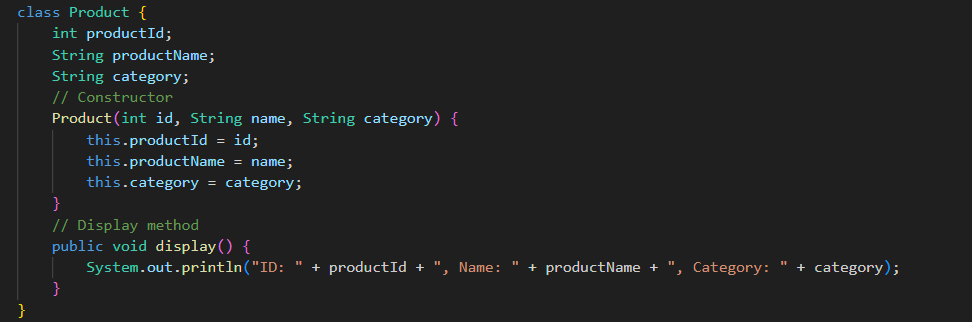
**Exercise 2: E-commerce Platform Search Function**

**Purpose:-The E-commerce Search Function exercise demonstrates how to implement and analyze basic search algorithms within the context of a real-world application. It introduces Big O notation to evaluate algorithm efficiency, builds a simple product model, and contrasts linear and binary search to guide informed decisions based on data size and organization. This pattern helps students understand the trade-offs between algorithm simplicity and performance, and prepares them to write scalable, maintainable code for search functionality.**

**-> Understand Asymptotic Notation**  
**Big O Notation describes the time complexity of an algorithm in terms of input size `n`.  
  
- Linear Search:  
 - Best Case: O(1) (first item)  
 - Average Case: O(n)  
 - Worst Case: O(n)  
  
- Binary Search (on sorted data):  
 - Best Case: O(1)  
 - Average Case: O(log n)  
 - Worst Case: O(log n)  
  
Binary Search is faster but requires sorted data. Linear Search is simpler and works on unsorted data.**

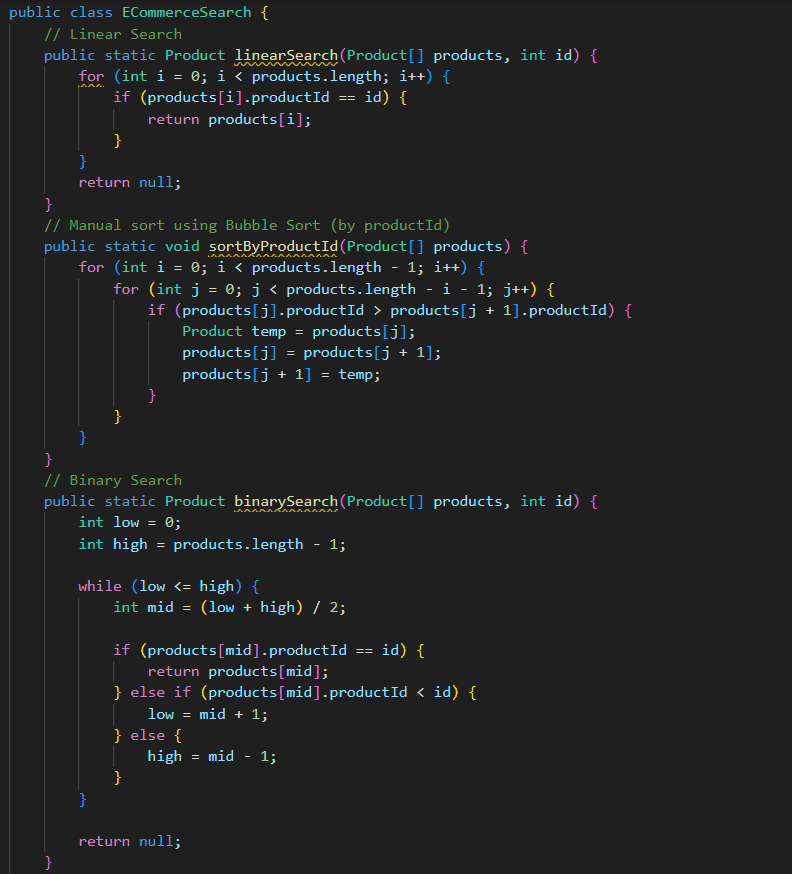
**->Product Class**

**Code:**



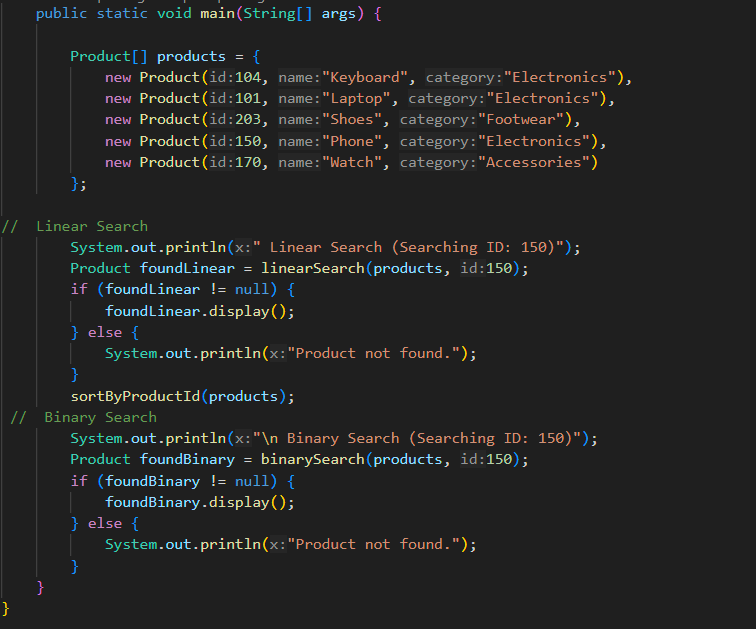
**-> Implementation of Linear and Binary Search**

**Code:**

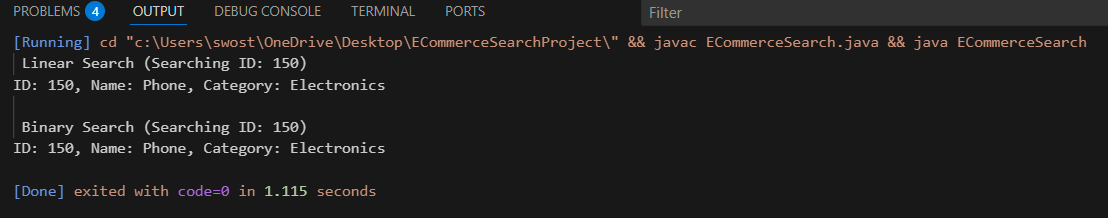
****

**->Main**

**Code:**

****

**Output:**

****

**->Time Complexity Analysis**

**Linear Search:  
- Best Case: O(1)  
- Average/Worst Case: O(n)  
  
Binary Search (after sorting):  
- Sorting (Bubble Sort): O(n^2)  
- Search: O(log n)  
  
->Binary Search is suitable for large, sorted datasets.  
->Linear Search is good for small or unsorted datasets.**

**In this scenario, Binary Search is the preferred method because it provides faster search performance with a time complexity of O(log n), which is ideal for handling large and sorted product datasets on an e-commerce platform. Although it requires the data to be sorted beforehand, the performance gain in search speed makes it well-suited for real-time product lookups. On the other hand, Linear Search is easier to implement but becomes inefficient as the number of products increases, making it less optimal for this use case.**