Algorithms\_Data Structures

Exercise 2: E-commerce platform search function

1. **UNDERSTAND ASYMPTOTIC NOTATION:**

**A. Explain Big O notation and how it helps in analyzing algorithms.**

=> Big O notationis used to describe the time complexity or space complexity of algorithms. Big-O is a way to express the upper boundof an algorithm’s time or space complexity.

* It describes the order of growth of time or space in terms of input size of a function.
* It provides anupper limit on the time taken by an algorithm in terms of the size of the input.
* We mainly consider the worst case scenario of the algorithm to find its time complexity in terms of Big O.
* It is denoted asO(f(n)), where f(n) is a function that represents the number of operations that an algorithm performs to solve a problem of size n.

Overview of Big O notation:

* O(1) - Constant: The runtime or space usage remains the same regardless of the input size.
* O(log n) - Logarithmic: The runtime increases logarithmically with the input size. This is very efficient for large datasets.
* O(n) - Linear: The runtime increases linearly with the input size.
* O(n log n) - Log-linear: The runtime is a combination of linear and logarithmic.
* O(n²) - Quadratic: The runtime increases quadratically with the input size.
* O(2ⁿ) - Exponential: The runtime grows exponentially with the input size.

**Importance in analyzing algorithms:**

* Big O notation helps in analyzing the efficiency of an algorithm.
* It allows programmers to compare different algorithms and choose the most efficient one for a specific problem.
* It helps in understanding the scalability of algorithms and predicting how they will perform as the input size grows.
* It enables developers to optimize code and improve overall performance

**B. Best ,average and worst case scenarios for search algorithms.**

=> When analyzing search operations, it's useful to consider the best, average, and worst-case scenarios:

1. Linear Search (O(n))

• Best Case:

Item is at the first index : O(1)

•Average Case:

Item is somewhere in the middle : O(n/2) ~ O(n)

•Worst Case:

Item is at the last index or not present : O(n)

2. Binary Search (O(log n) in sorted arrays)

•Best Case:

Item is at the middle : O(1)

•Average Case:

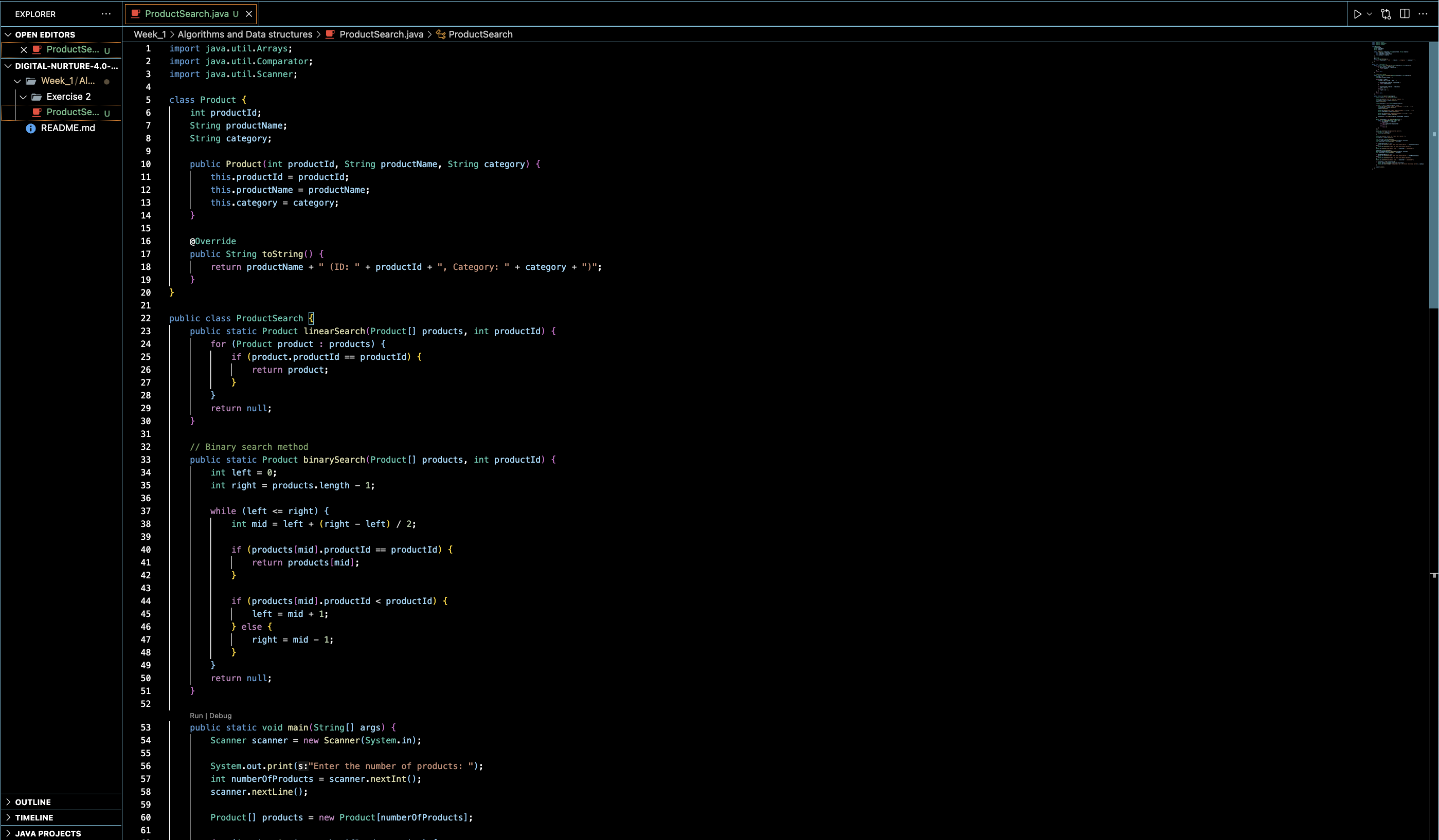
Multiple comparisons to halve the array repeatedly : O(log n)

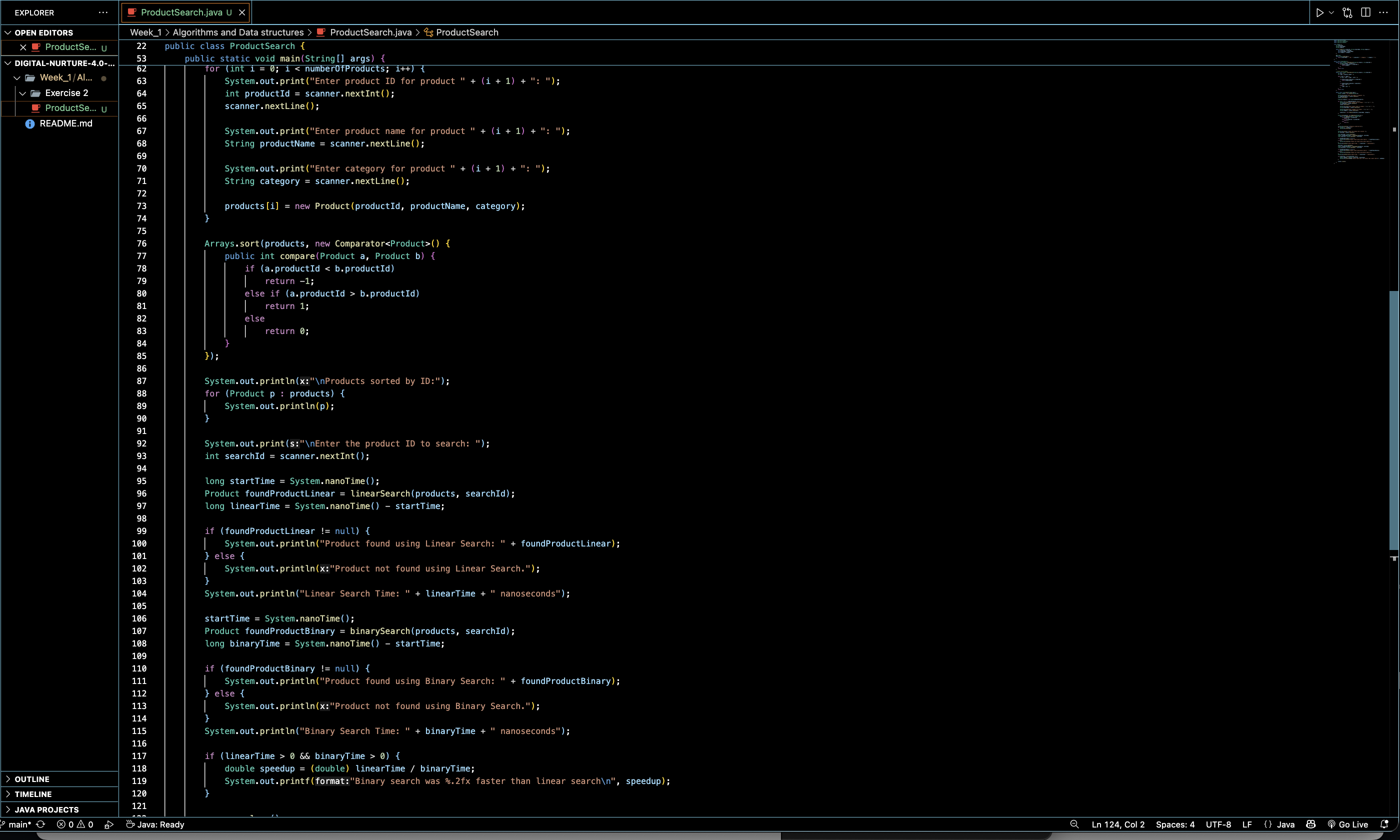
•Worst Case:

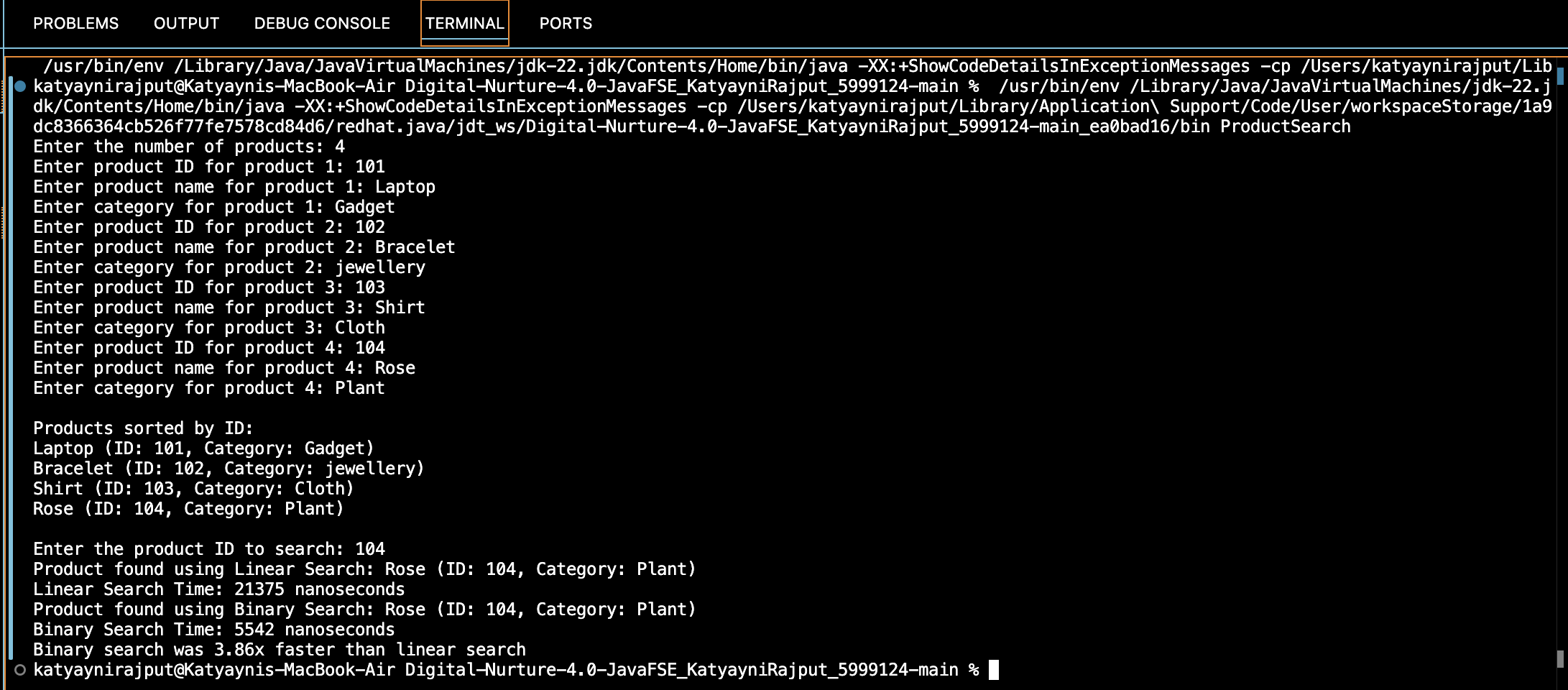
Item is not found or found after all log(n) steps : O(log n)

**2. SETUP AND IMPLEMENTATION:**

**A. Create a class Product with attributes for searching, such as productId, productName, and category.**

**B. Implement linear search and binary search.**

****

**Output:**

Binary Search is more suitable for an e-commerce platform because:

* It is more suitable if data is sorted.
* It is much faster on larger datasets.
* It is perfect for platforms with thousand of products like e commerce platform.