**Output of Week-1 Data Structure & Algorithms**

**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.

🡪 Big O notation describes the **upper bound** of an algorithm's running time or space usage in terms of input size n.

🡪 It helps developers analyze how algorithms scale and compare their efficiency, especially as the input size grows. It's crucial in search functionality for e-commerce platforms where large product datasets are common.

* + Describe the best, average, and worst-case scenarios for search operations.

🡪Best Case:

1. Linear Search : O(1) if the target is the first item.
2. Binary Search : O(1) if the target is the middle element

🡪Average Case:

1. Linear Search : O(n) as random distribution.
2. Binary Search : O(logn) as random distribution.

🡪Worst Case:

1. Linear Search : O(n) if the target is the last item or not present.
2. Binary Search : O(logn) due to multiple recursive splits until one remains.
3. **Analysis:**
   1. Compare the time complexity of linear and binary search algorithms.

🡪Linear Search O(n) time complexity:

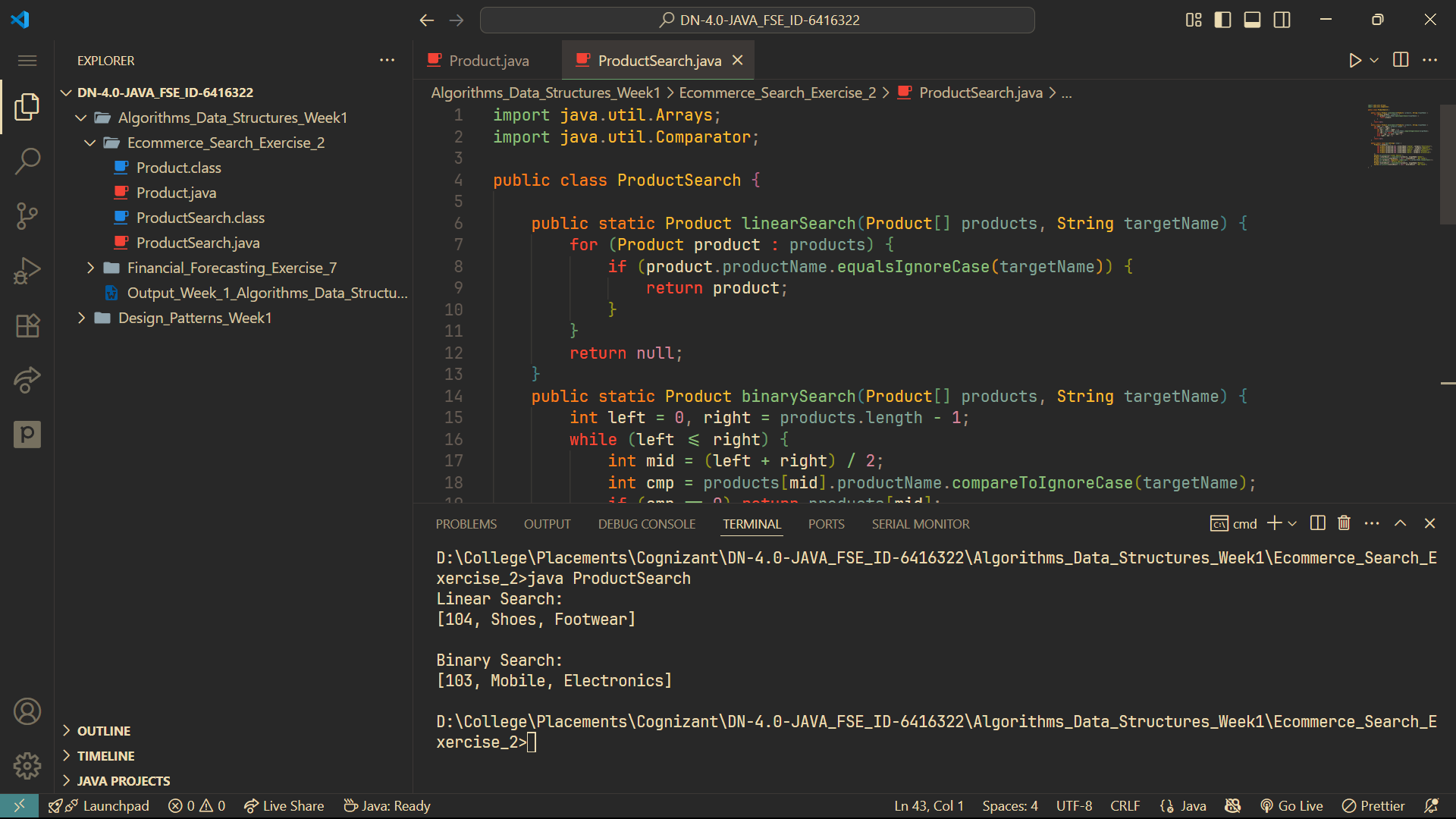
* + - 1. Does not require any sorting .
      2. Simpler to Implement.
      3. Suitable for small and unordered datasets.

🡪Binary Search O(logn) time complexity:

1. Sorting of data is needed.
2. Much faster for large datasets.
   1. Discuss which algorithm is more suitable for your platform and why.

🡪 As the product list of an E-commerce platform is constantly growing, thus a scalable fast search is important , thus binary search (with a sorted structure or index) is a better choice than linear search.

Output:



**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.

🡪 Recursion is a programming technique where a **function calls itself** to solve smaller instances of a problem **until it reaches a base case**. **It is especially useful for problems that have a natural hierarchical or repetitive structure, like computing factorials, Fibonacci numbers, or traversing tree structures**. In the context of financial forecasting, recursion can be used to calculate future values based on previous ones.

1. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.

🡪For the recursive algorithm futValRecursive(method), the recursion depth is the number of years (n) so it makes n recursive calls.

🡪Therefore, the Time Complexity: O(n).

* + Explain how to optimize the recursive solution to avoid excessive computation.

🡪 To optimize it so as to reduce the number of computations, I have memorized the solution which stores the previously computed values in the memo array, thus it avoids recomputing values for the same year multiple times.

Output:

