**WEEK-1**

**Solutions of Data Structures and Algorithm**

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

**#Understand Asymptotic Notation:**

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

->**Big O** measures how an algorithm scales with input size.

Eg: Linear Search - O(n)

Binary Search – O(n) (only for sorted arrays)

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

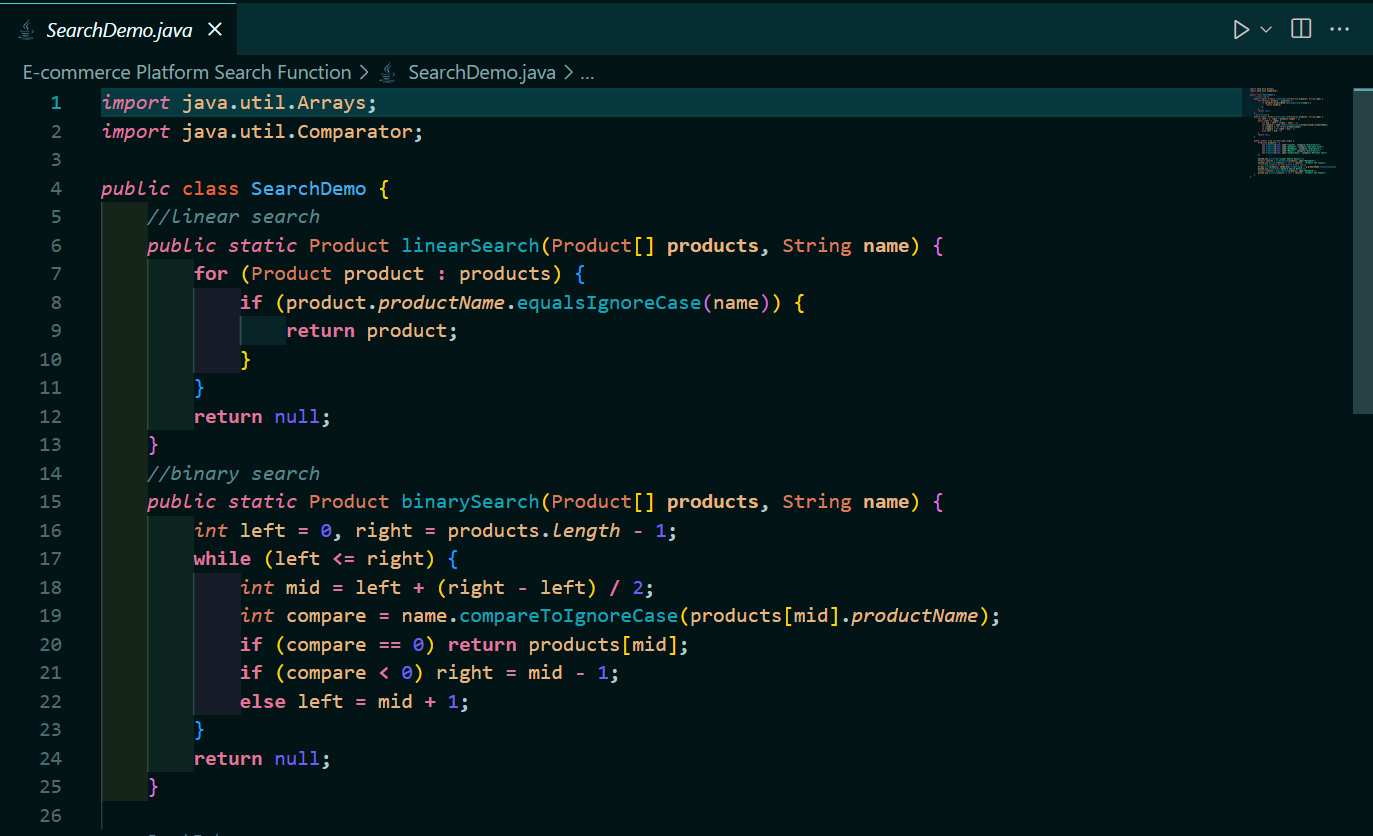
->

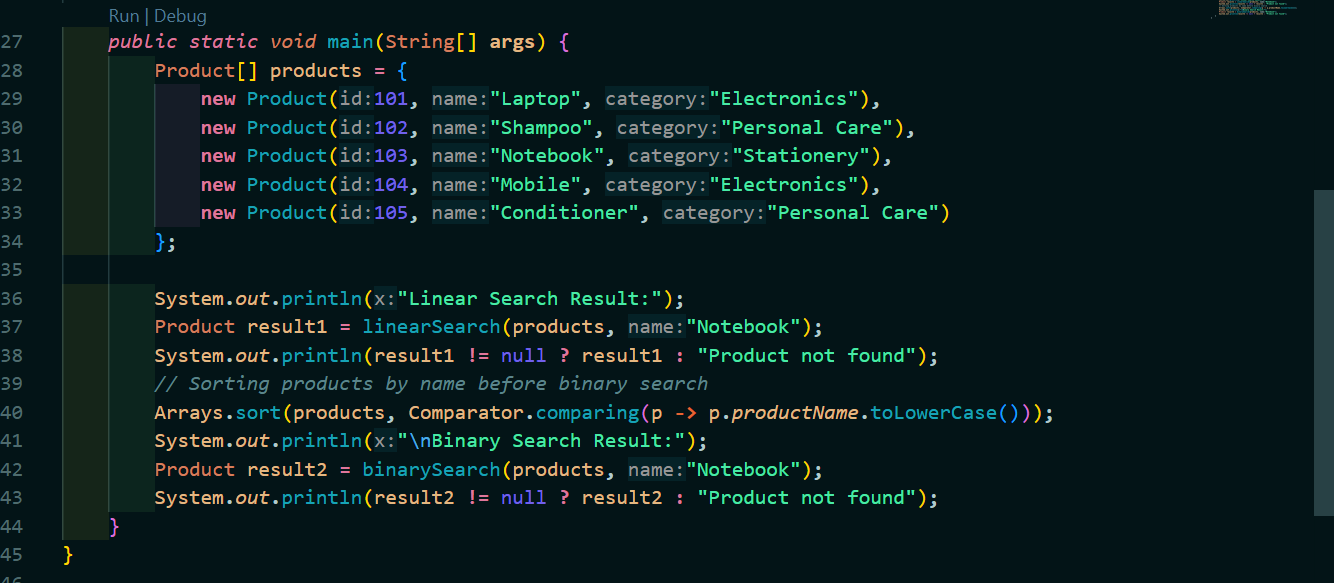
**#CODE**

Product Class

****

Implementation

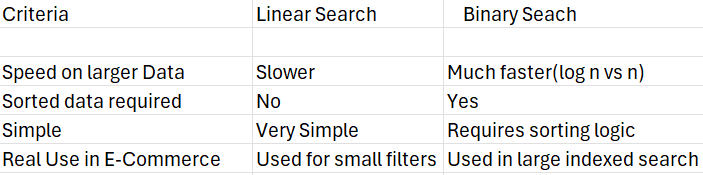




Output



**#Analysis**

****

* We can use **Linear Search** for small unsorted lists (e.g., filters).
* We can use **Binary Search** for large sorted product arrays (e.g., search bar).
* Big O helps you choose the right algorithm for performance-critical tasks

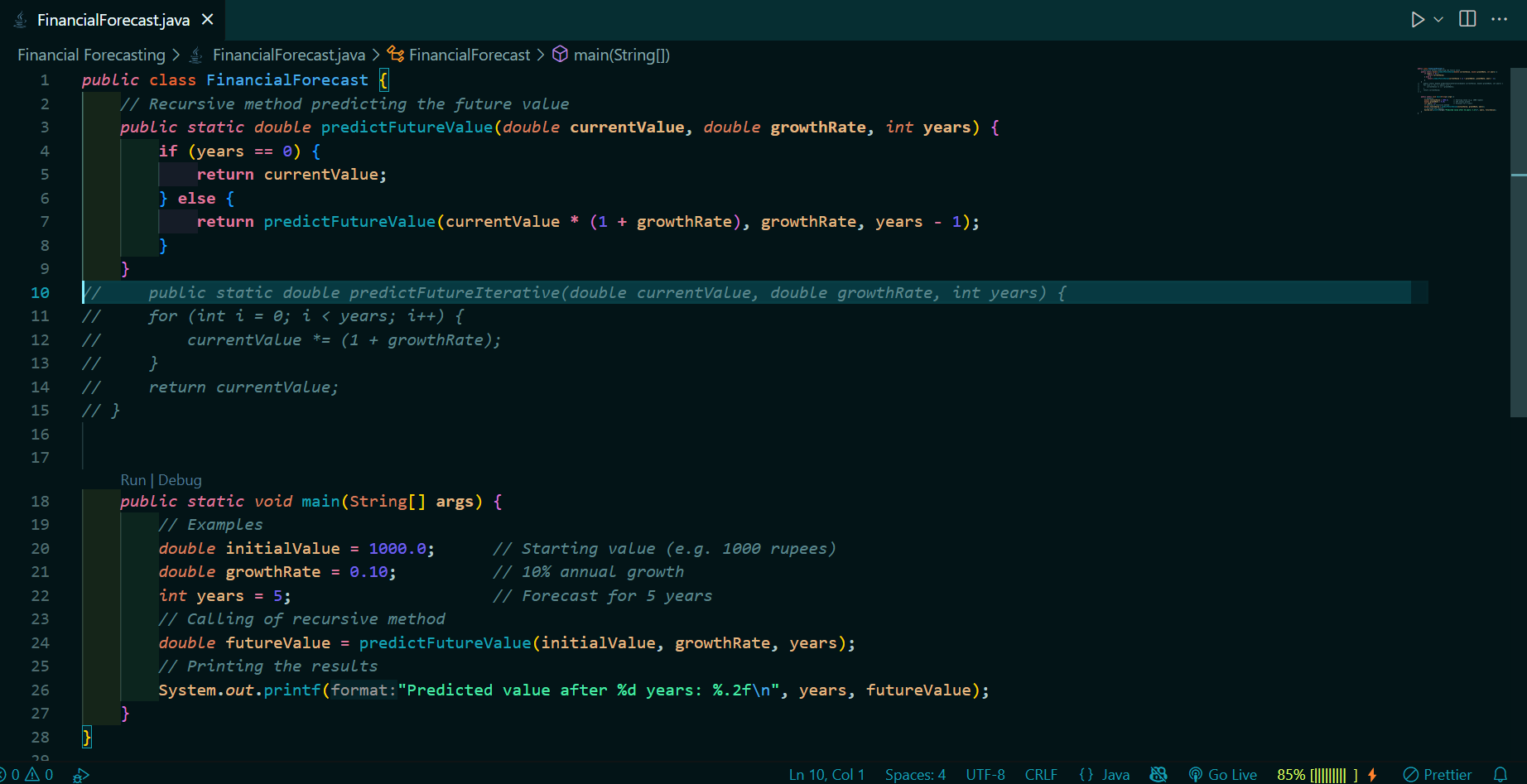
**Exercise 7: Financial Forecasting**

**#Understand Recursive Algorithms:**

**1.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 the same problem. It's often used for problems that can be broken down into **simpler subproblems**, which follow the same pattern.

#CODE



**#OUTPUT**

**Explanation:**

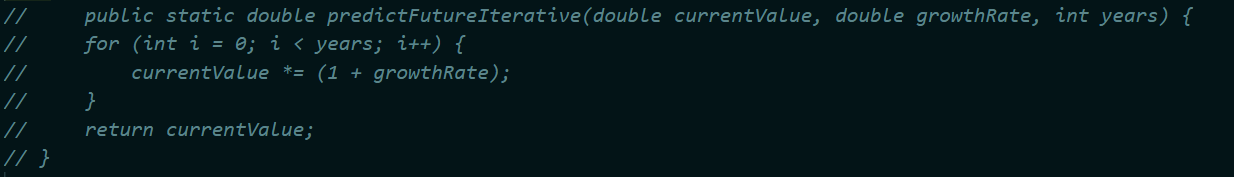
This uses compound interest-like recursion:

* Year 0: 1000.00
* Year 1: 1000 × 1.10 = 1100.00
* Year 2: 1100 × 1.10 = 1210.00
* Year 3: 1210 × 1.10 = 1331.00
* Year 4: 1331 × 1.10 = 1464.10
* Year 5: 1464.10 × 1.10 = 1610.51

**#Time complexity**

The time complexity for this recursive function is O(n) where n=number of years, as this method runs once per year.

**#Optimization**

****

We can use Iterative version faster if the n is large.