**Week-1**

**Data Structures and Algorithms**

**HandsOn**

**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.

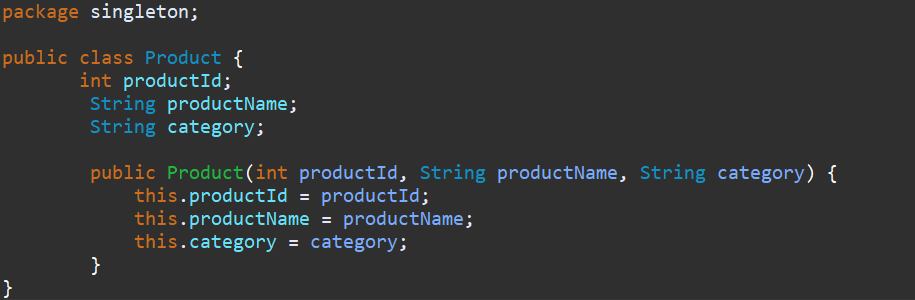
1. **Understand Asymptotic Notation:**

**Big O notation**

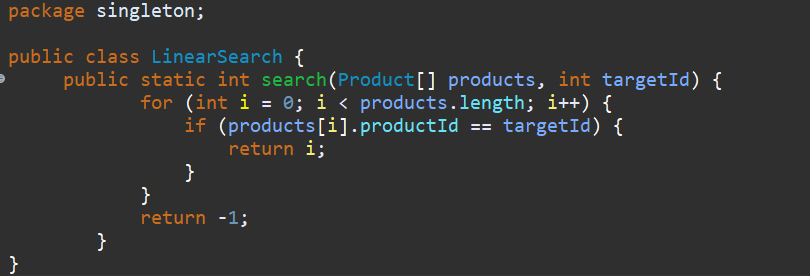
**Big O notation** is a mathematical way to describe the **efficiency of an algorithm** as the input size increases. It focuses on the **growth rate** of an algorithm rather than exact execution time, helping developers understand how scalable or efficient their code is.

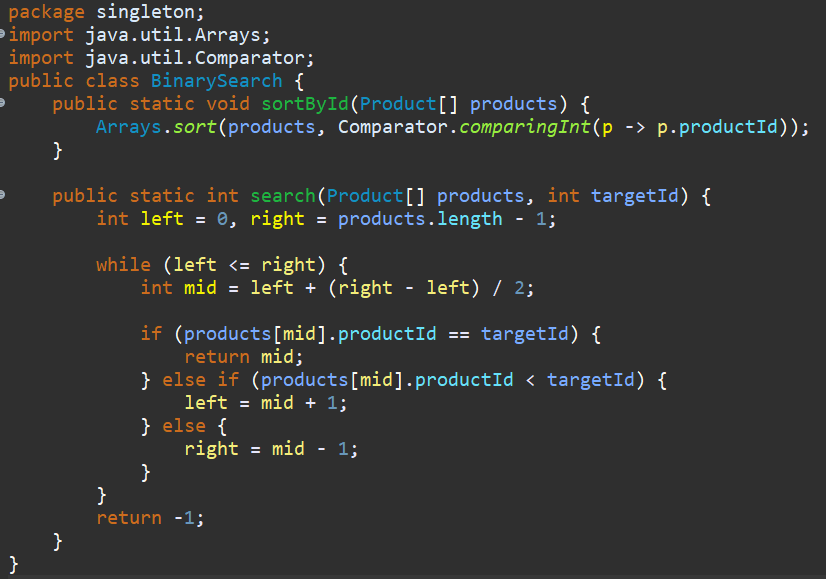
* It **abstracts away constants and lower-order terms**, giving a high-level view of performance.
* Helps in **comparing algorithms**.
* Useful in predicting **how the program behaves with large inputs**.

1. **Setup:**



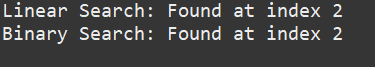
1. **Implementation:**

****

****

****

**Output:**



1. **Analysis :**

**Linear Search**

* **Description**: Go through each element one by one until the target is found.

| **Case** | **Time Complexity** | **When It Happens** |
| --- | --- | --- |
| **Best** | O(1) | Target is the **first** element. |
| **Average** | O(n) | Target is somewhere in the **middle**. |
| **Worst** | O(n) | Target is at the **end** or **not present**. |

**Binary Search (works only on sorted data)**

* **Description**: Repeatedly divide the search space in half.

| **Case** | **Time Complexity** | **When It Happens** |
| --- | --- | --- |
| **Best** | O(1) | Target is at the **middle** on the first try. |
| **Average** | O(log n) | Target is found after several divisions. |
| **Worst** | O(log n) | Target is **not present** or found last. |

**Exercise 7: Financial Forecasting**

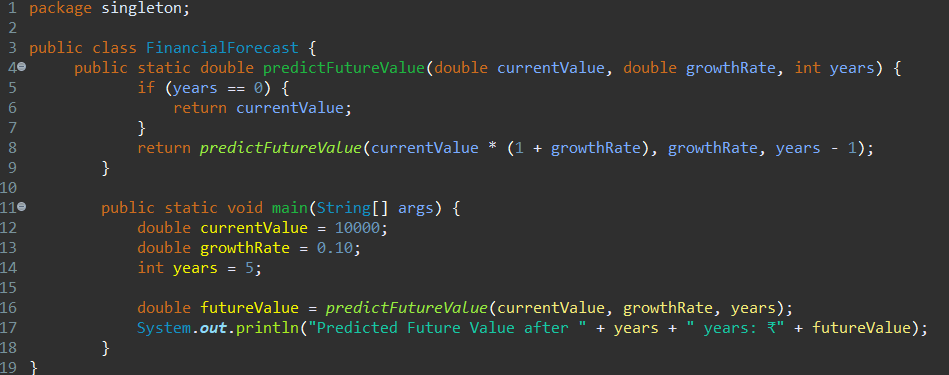
**Scenario:**

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

1. **Understanding Recursive Algorithms:**

Recursion is a programming technique where a method calls itself to solve a smaller subproblem of the original task. It helps simplify problems that can be broken down into smaller, repeatable steps. In financial forecasting, recursion can be used to repeatedly apply growth over a time period to estimate future values.

1. **Setup and Implementation**

****

**Output:**

****

**4. Analysis**

**Time Complexity:**

The time complexity of this recursive approach is O(n), where n is the number of years. Each recursive call processes one year.

**Optimization Suggestion:**

For large values of n, the recursive approach can lead to deep call stacks. To optimize:

* Convert to an iterative approach to avoid recursion depth.
* Use memoization if overlapping subproblems exist .