**WEEK- 1**

**ALGORITHMS DATA STRUCTURES**

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

**SOLUTION: Step 1**

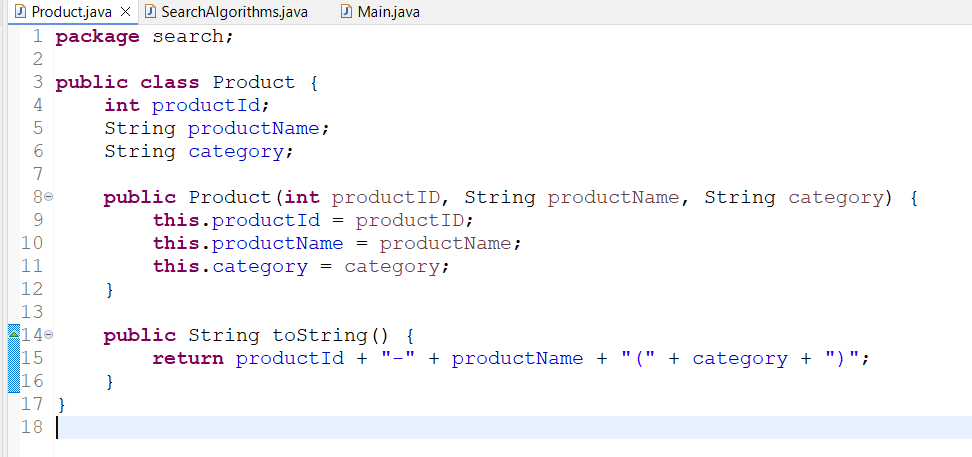
Big O Notation is used to describe how the runtime or space requirements of an algorithm scale with input size (n).

| Big O | Meaning | Example |
| --- | --- | --- |
| O(1) | Constant Time | Accessing an array element by index |
| O(n) | Linear Time | Linear Search |
| O(log n) | Logarithmic Time | Binary Search |
| O(n²) | Quadratic Time | Nested loops over n |

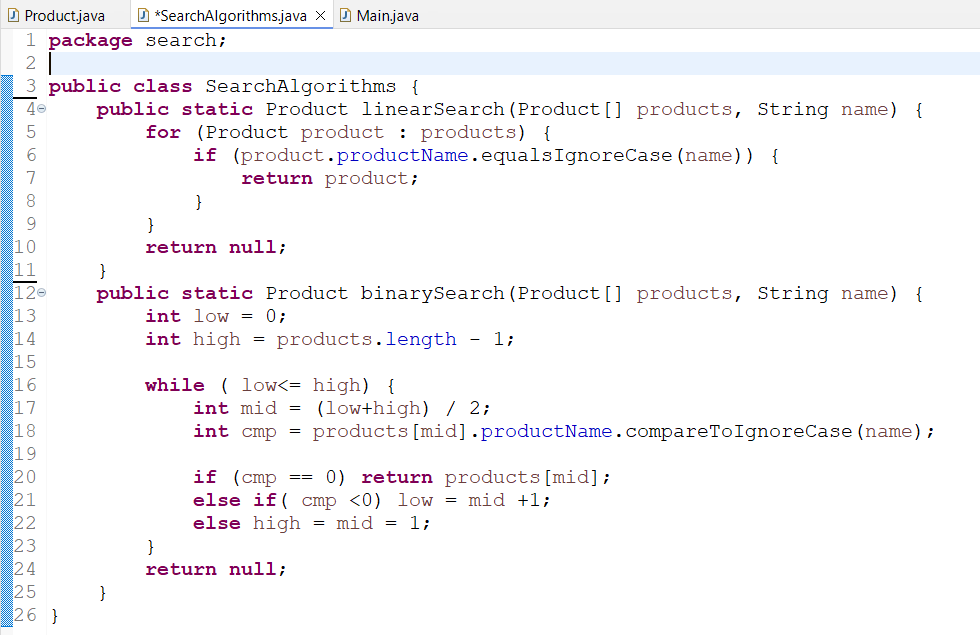
Search Operation Scenarios

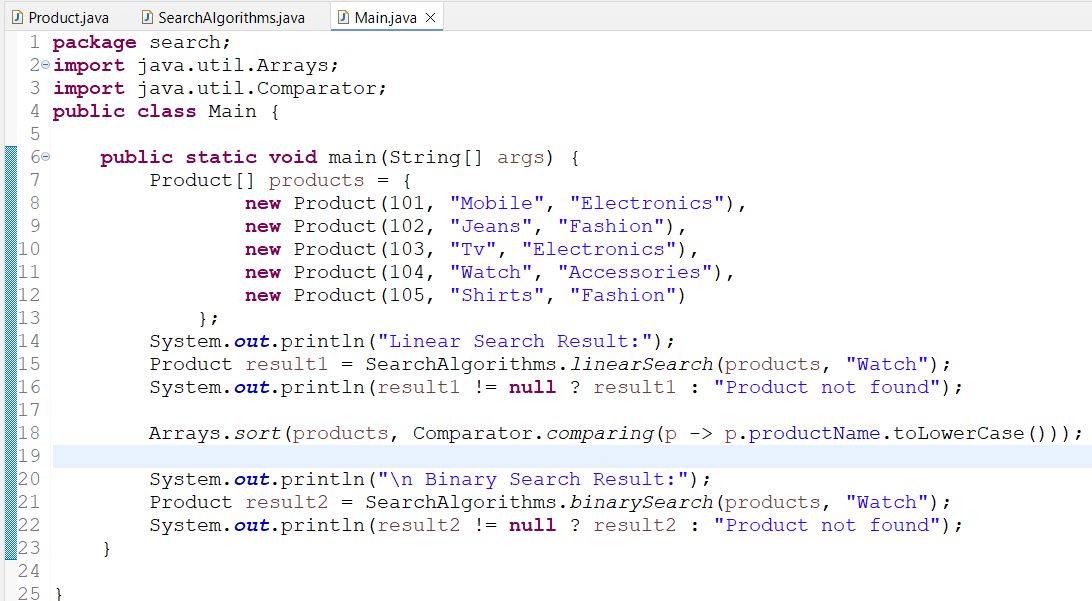
* Best Case: Target found immediately (e.g., first element).
* Average Case: Target found in the middle of data.
* Worst Case: Target is at the end or not present at all.

**CODE: Step 2:**

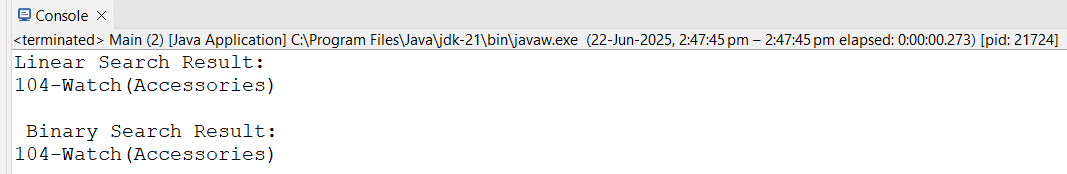
****

**Step 3:**

****

****

**Output:**

****

**Step 4: Analysis**

**Time Complexity**

| Search Type | Time Complexity | Best Case | Worst Case |
| --- | --- | --- | --- |
| Linear Search | O(n) | O(1) (first match) | O(n) (last or not found) |
| Binary Search | O(log n) | O(1) | O(log n) |

* We can use Linear Search when:
  + The list is unsorted.
  + Dataset is small or infrequently searched.
* And Use Binary Search when:
  + The list is sorted.
  + Search is frequent or dataset is large.

Here for an e-commerce platform:

* Products are often sorted
* Searches are frequent.  
   Therefore Binary Search is better if data is sorted .

**Exercise 7: Financial Forecasting**

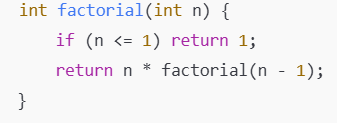
**SOLUTION:**

**Step 1:**

Recursion is a technique where a method calls itself to solve smaller instances of the same problem.

Instead of solving a problem in a single step, recursion breaks it down into simpler, smaller instances of the same problem. This process continues until it reaches a base case, which is a simple, easily solvable situation.

Like in calculating the factorial of a number :



Now how it can simplify certain problems:-

Recursion can turn complex problems into more manageable sub-problems. For example:

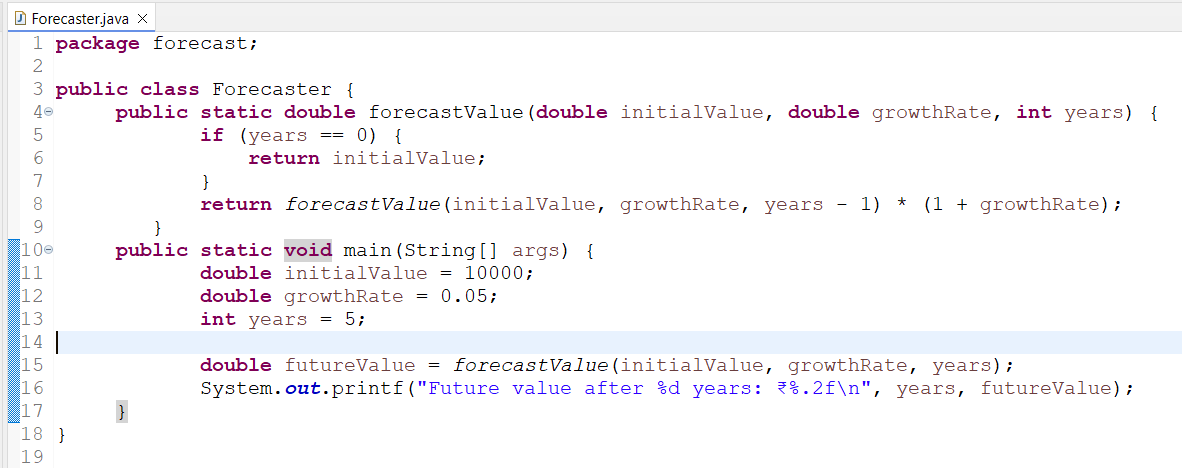
* Traversing data structures like trees and graphs
* Sorting algorithms like quicksort and mergesort
* Solving combinatorial problems

By breaking a problem down into smaller, similar problems, recursion makes code more elegant and easier to understand, especially for problems naturally defined in a recursive manner.

**CODE:**

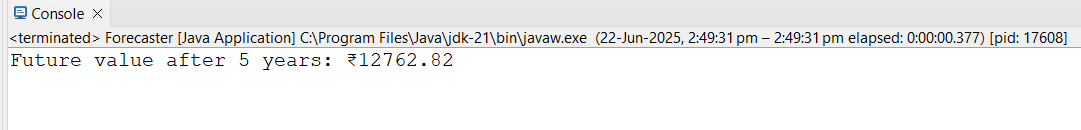
**Step 2:**

**Recursive Approach**

****

**Analysis**

**Output:**

****

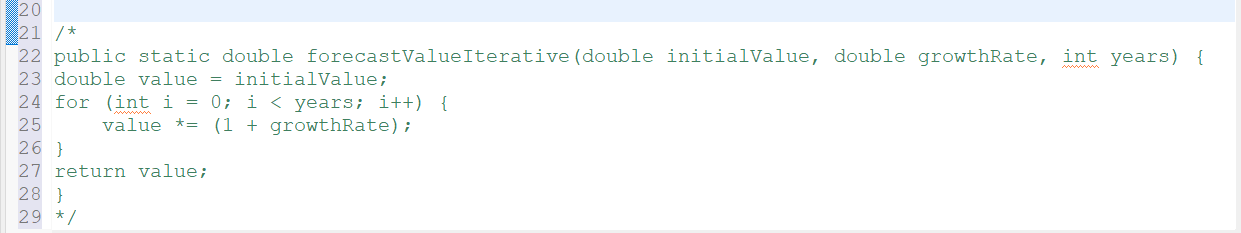
**Time Complexity**

Each recursive call reduces years by 1 until it reaches 0:

* T(n) = T(n-1) + 1
* So, Time Complexity: O(n)

**Optimization:**

we can also use iterative approach like this :



**SUBMITTED BY:**

Superset ID: 6393474

Name- Manjeet Swaroop

Email – [manjeet.swaroop@gmail.com](mailto:manjeet.swaroop@gmail.com)

Course: DIGITAL NURTURE 4.0 JAVA FSE