**Week - 1**

**Algorithms\_Data Structures**

**1. E-Commerce Platform Search Function – Java Implementation**

**Project Name** : E-Commerce Search

**Exercise**  : Implementing the E-Commerce Platform Search Function

**Name**  : Alladi Manasa

**Superset Id** : 6373907

Introduction :

This project implements a basic search functionality for an e-commerce platform. The goal is to allow searching products efficiently using both linear and binary search techniques, comparing their performance, and understanding when to use each.

**Objectives :**

* Create a Product class with relevant fields.
* Implement both linear and binary search algorithms.
* Demonstrate the working of both algorithms using a sample product array.
* Analyze time complexities and determine the most efficient search strategy.

**Technology Stack:**

| **Component** | **Technology Used** |
| --- | --- |
| Programming Language | Java |
| IDE Used | Eclipse IDE |
| Data Structure | Arrays |
| Algorithms Used | Linear Search, Binary Search |
| Execution | Console-based output |

**System Design :**

**Class : Product**

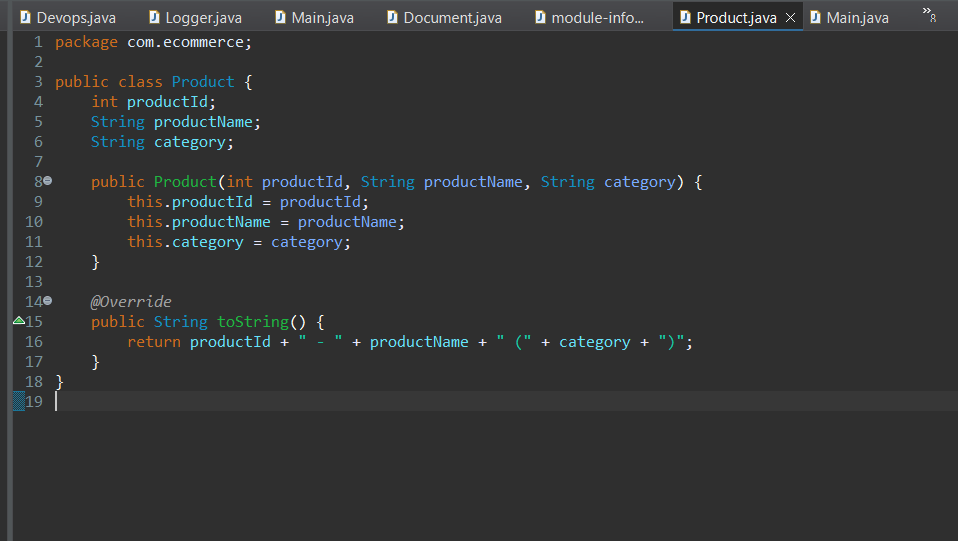
| **Attribute** | **Description** |
| --- | --- |
| productId | Unique identifier for product |
| productName | Name of the product |
| category | Type or category of the product |

**Class : Main**

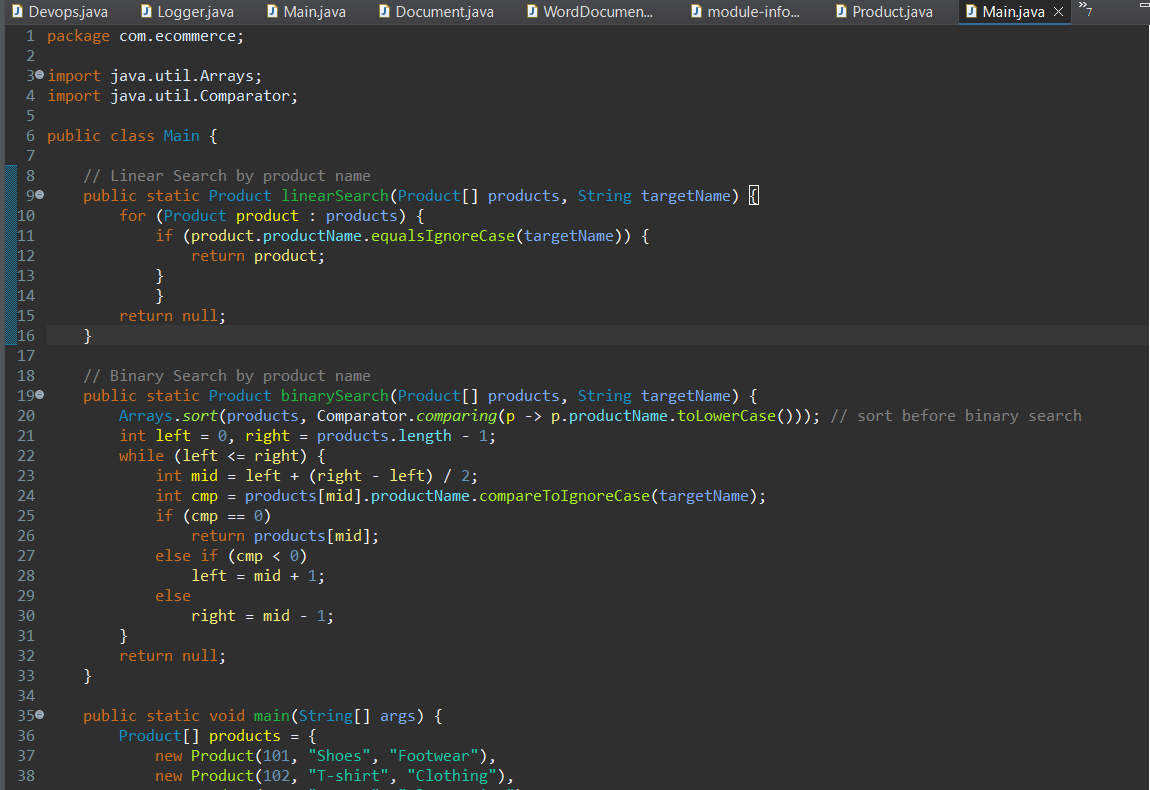
* Contains the main() method
* Creates an array of Product objects
* Implements linearSearch() method
* Implements binarySearch() method

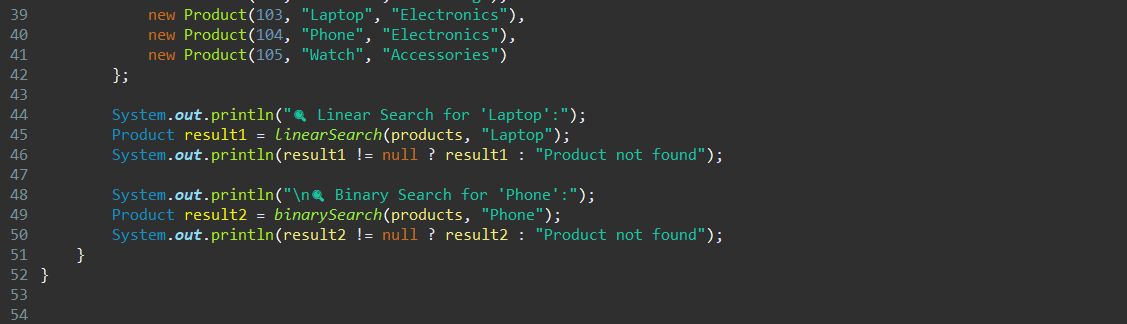
**Implementation :**

**Product.java**

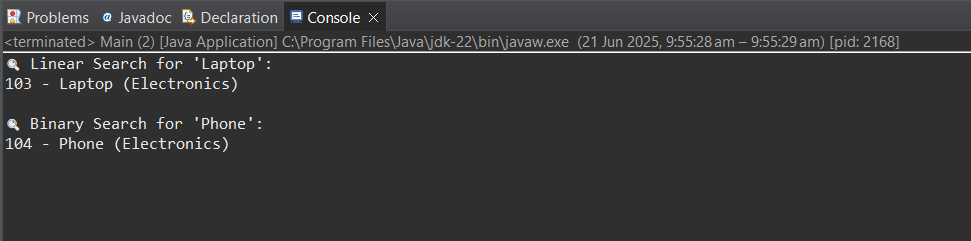


**Main.java**





**Output :**



### ****Analysis :****

#### Time Complexity Comparison

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| Linear Search | O(1) | O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

* Linear search is simple and works on unsorted data but becomes slow with large lists.
* Binary search is faster (logarithmic time) but only works on **sorted arrays.**
* **Binary search** is recommended for fast performance when data is sorted.

**Conclusion :**  
This project successfully demonstrates how linear and binary search algorithms work in Java. It compares their efficiencies and explains their best use cases. In an e-commerce platform where performance is key and large datasets are common, binary search is more efficient and preferred—given the data is sorted.

**2. Financial Forecasting using Recursive Algorithm (Java)**

**Project Name** : FinancialForecasting

**Exercise**  : Implementing Financial Forecasting using Recursion

**Name**  : Alladi Manasa

**Superset Id** : 6373907

**Introduction :**

This project implements a financial forecasting tool that calculates the future value of an investment or amount using a recursive algorithm. Recursion simplifies the logic by breaking down the future value calculation year by year based on a constant growth rate.

**Objectives :**

* To understand the concept of recursion and how it helps simplify calculations.
* To implement a recursive method for predicting financial values over time.
* To apply the method in a Java program using Eclipse IDE.
* To analyze time complexity and explore optimization strategies

**Technology Stack :**

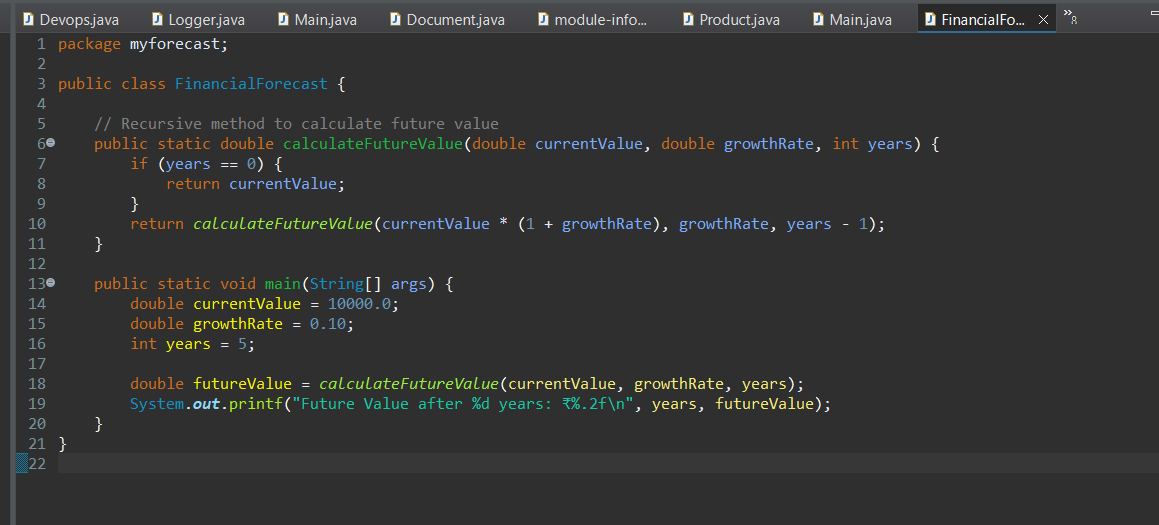
| **Component** | **Tool/Language Used** |
| --- | --- |
| Programming Language | Java |
| IDE | Eclipse IDE |
| Concept Used | Recursion |
| Output Format | Console-based |

**System Design :**

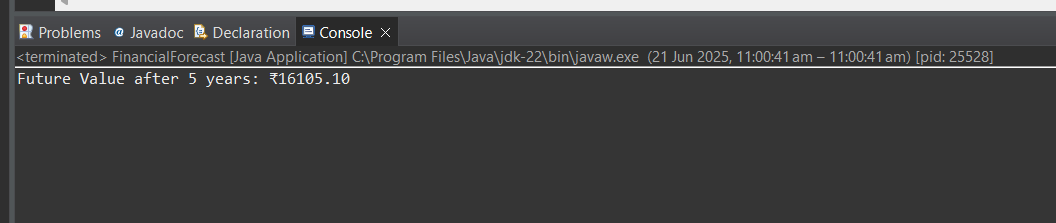
| **Input** | **Data Type** | **Example** | **Description** |
| --- | --- | --- | --- |
| currentValue | double | 10000.0 | Initial amount |
| growthRate | double | 0.10 | 10% annual growth (as decimal) |
| years | int | 5 | Forecast period in years |

**Implementation :**

**FinancialForecast.java**



**Output :**



**Time Complexity Analysis :**

Since the recursive function runs once for each year, the time complexity is **O(n)**, where n = years.

**Conclusion :**

This project shows how recursion can be used in real-world financial applications to predict future values. Recursive logic is simple and elegant but must be optimized for efficiency in large-scale scenarios.