**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:

Understand Asymptotic Notation:

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

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

Setup:

Create a class Product with attributes for searching, such as productId, productName, and category.

Implementation:

Implement linear search and binary search algorithms.

Store products in an array for linear search and a sorted array for binary search.

Analysis:

Compare the time complexity of linear and binary search algorithms.

Discuss which algorithm is more suitable for your platform and why.

**Program files**

**Product.java**

public class Product {

int productId;

String productName;

String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

**LinearSearch.java**

public class LinearSearch {

public static Product search(Product[] products, String targetName) {

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

}

**BinarySearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

public static Product search(Product[] products, String targetName) {

Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 2;

int compare = products[mid].productName.compareToIgnoreCase(targetName);

if (compare == 0) return products[mid];

if (compare < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Fashion"),

new Product(3, "Phone", "Electronics"),

new Product(4, "Watch", "Accessories")

};

Product foundLinear = LinearSearch.search(products, "Phone"); //Time Complexity O(N)

System.out.println("Linear Search Result: " + (foundLinear != null ? foundLinear : "Not Found"));

Product foundBinary = BinarySearch.search(products, "Phone");

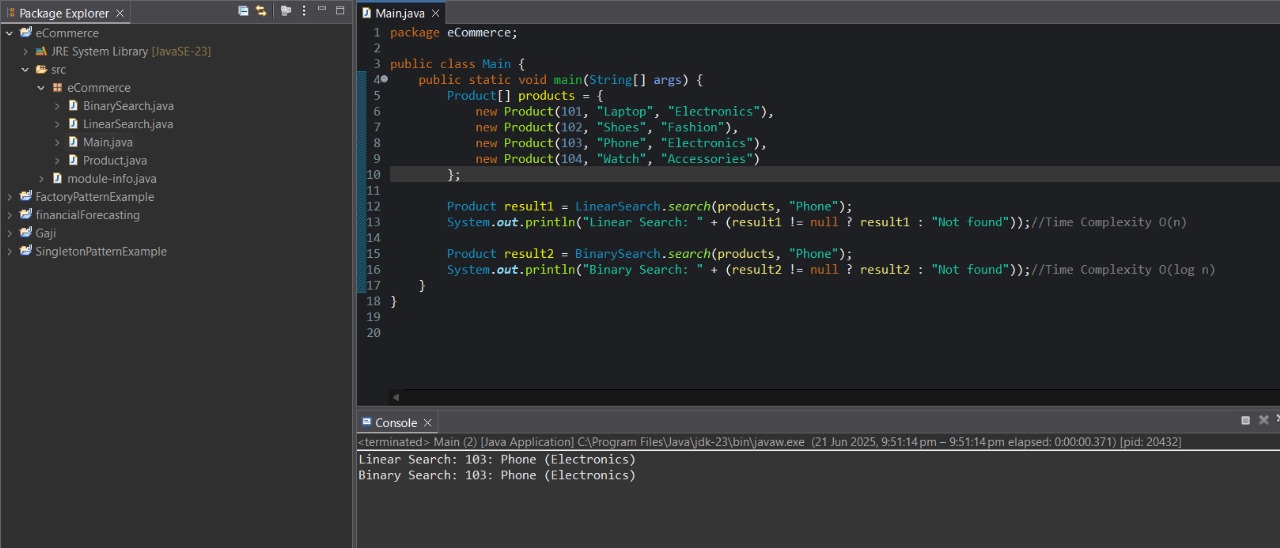
//Time Complexity O(log N) needs sorting

System.out.println("Binary Search Result: " + (foundBinary != null ? foundBinary : "Not Found"));

}

}

**Output:**

****