**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.

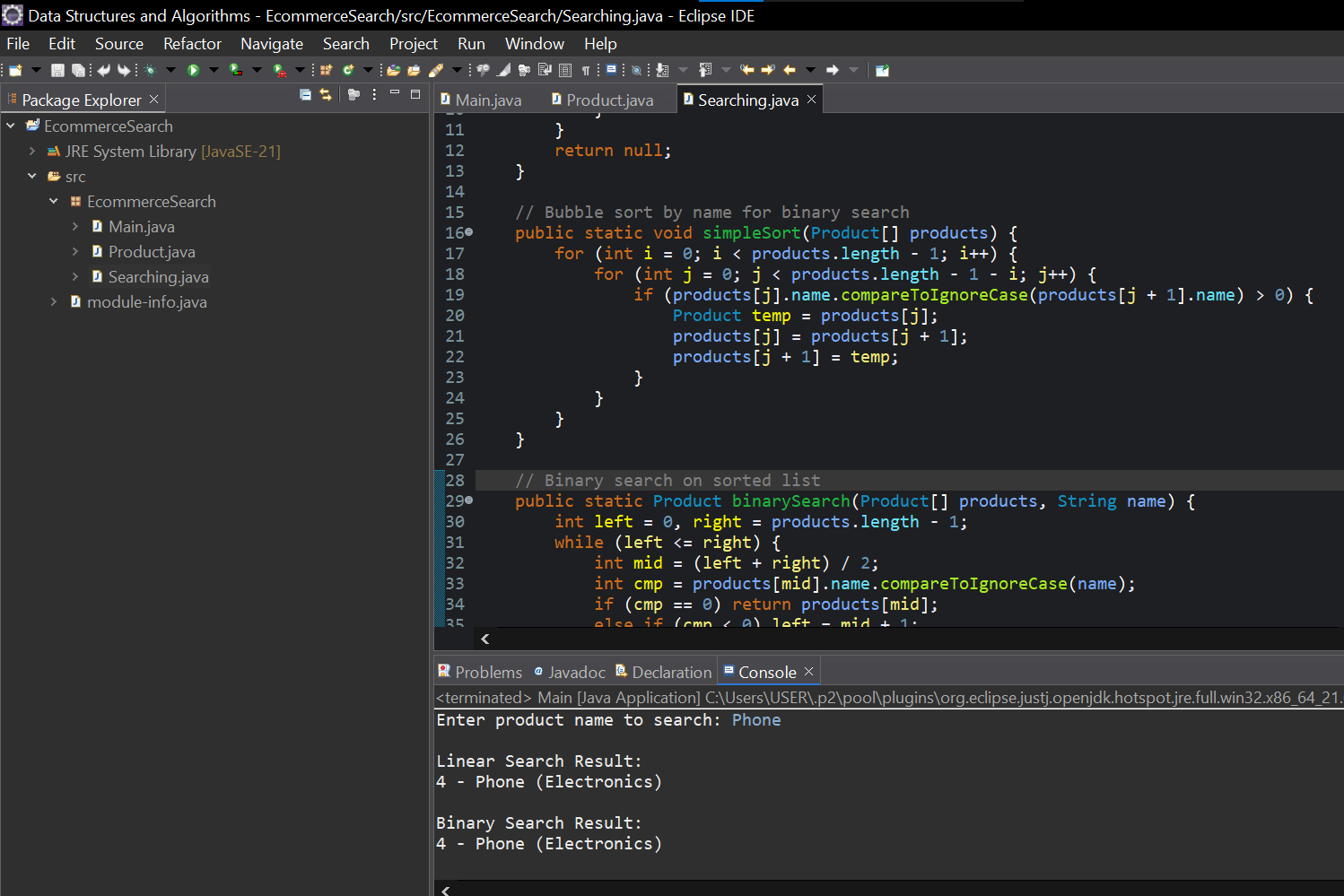
**Steps:**

1. **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.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

Solutions:

1. The efficiency of an algorithm is measured using Big O notation, particularly as the size of the input increases. It helps us compare various solutions by displaying how quickly or slowly an algorithm executes. There are three possible outcomes for search operations: the best case, which is when the item is found right away (like O(1)), the average case, which is when the item is in the middle, and the worst case, which is when the item is last or not found (like O(n) for linear search). Binary search splits the list and is significantly faster (O(log n)), but it requires sorted data, whereas linear search checks each item individually (O(n)). Big O assists us in selecting the optimal algorithm for increased efficiency.

2&3.

Source code for the **E-commerce Platform Search Function**

Screenshot of the Application

Product.java

package EcommerceSearch;

public class Product {

int id;

String name;

String category;

public Product(int id, String name, String category) {

this.id = id;

this.name = name;

this.category = category;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String toString() {

return id + " - " + name + " (" + category + ")";

}

}

Searching.java

package EcommerceSearch;

public class Searching {

// Linear search

public static Product linearSearch(Product[] products, String name) {

for (Product p : products) {

if (p.name.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

// Bubble sort by name for binary search

public static void simpleSort(Product[] products) {

for (int i = 0; i < products.length - 1; i++) {

for (int j = 0; j < products.length - 1 - i; j++) {

if (products[j].name.compareToIgnoreCase(products[j + 1].name) > 0) {

Product temp = products[j];

products[j] = products[j + 1];

products[j + 1] = temp;

}

}

}

}

// Binary search on sorted list

public static Product binarySearch(Product[] products, String name) {

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

while (left <= right) {

int mid = (left + right) / 2;

int cmp = products[mid].name.compareToIgnoreCase(name);

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

else if (cmp < 0) left = mid + 1;

else right = mid - 1;

}

return null;

}

}

Main.java

package EcommerceSearch;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Product[] products = {

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

new Product(2, "Book", "Education"),

new Product(3, "Shoes", "Fashion"),

new Product(4, "Phone", "Electronics"),

new Product(5, "Watch", "Fashion")

};

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter product name to search: ");

String searchName = scanner.nextLine();

//linear search

System.***out***.println("\nLinear Search Result:");

Product result1 = Searching.*linearSearch*(products, searchName);

System.***out***.println(result1 != null ? result1 : "Product not found");

//binary search

Searching.*simpleSort*(products); //prerequisite to Sort array before binary search

System.***out***.println("\nBinary Search Result:");

Product result2 = Searching.*binarySearch*(products, searchName);

System.***out***.println(result2 != null ? result2 : "Product not found");

scanner.close();

}

}

Output:

Enter product name to search: Phone

Linear Search Result:

4 - Phone (Electronics)

Binary Search Result:

4 - Phone (Electronics)

4.

Linear search is O(n), which means it goes through each element in turn until a match is found or the end is reached. Binary search, on the other hand, is O(log n), which is considerably quicker as it continuously halves the sorted list to locate the target. For a large e-commerce website with enormous product data, binary search is preferable as it provides quicker performance, particularly when products are already sorted based on name or ID. When dealing with small or unsorted data, linear search is the option due to its ease of use. In general, for improved speed and scalability, binary search is the optimal choice.