**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:**

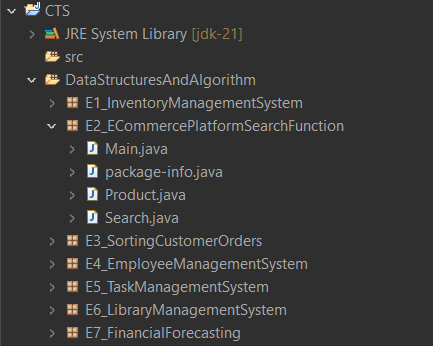
1. **Understand Asymptotic Notation:**
   * **Explain Big O notation and how it helps in analyzing algorithms.**

Big O describes the upper bound of time or space complexity of an algorithm in terms of input size n.  
It helps us:

* Predict performance as data grows
* Choose efficient algorithms for large datasets
  + **Describe the best, average, and worst-case scenarios for search operations.**

|  |  |  |
| --- | --- | --- |
| Case | Linear Search | Binary Search |
| Best | O(1) (first item) | O(1) (middle element) |
| Average | O(n) | O(log n) |
| Worst | O(n) | O(log n) |

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

****

1. **Implementation:**
   * **Implement linear search and binary search algorithms.**
   * **Store products in an array for linear search and a sorted array for binary search.**

**Product.java**

package E2\_ECommercePlatformSearchFunction;

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 String toString() {

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

}

}

**Search.java**

package E2\_ECommercePlatformSearchFunction;

public class Search {

// Linear Search

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

for (Product p : list) {

if (p.name.equalsIgnoreCase(name)) return p;

}

return null;

}

// Binary Search (array must be sorted)

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

int low = 0, high = list.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

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

else if (cmp > 0) low = mid + 1;

else high = mid - 1;

}

return null;

}

}

**Main.java**

package E2\_ECommercePlatformSearchFunction;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Product[] products = {

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

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

new Product(3, "T-shirt", "Clothing"),

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

};

// Sort for binary search

Arrays.*sort*(products, Comparator.*comparing*(p -> p.name));

// Try linear search

Product found1 = Search.*linearSearch*(products, "Phone");

System.*out*.println("Linear Search: " + (found1 != null ? found1 : "Not found"));

// Try binary search

Product found2 = Search.*binarySearch*(products, "Phone");

System.*out*.println("Binary Search: " + (found2 != null ? found2 : "Not found"));

//Another Example

Product found3 = Search.*linearSearch*(products, "Bag");

System.*out*.println("Linear Search: " + (found3 != null ? found3 : "Not found"));

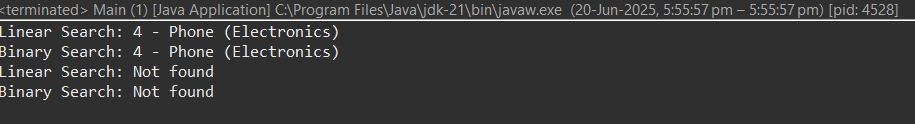
Product found4 = Search.*binarySearch*(products, "bag");

System.*out*.println("Binary Search: " + (found4 != null ? found4 : "Not found"));

}

}

**Output:**

****

1. **Analysis:**
   * **Compare the time complexity of linear and binary search algorithms.**

|  |  |  |
| --- | --- | --- |
| **Search Type** | **Time Complexity** | **When to Use** |
| **Linear Search** | O(n) | Small datasets or unsorted data |
| **Binary Search** | O(log n) | Large datasets that are sorted |

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

**Binary search is more suitable** when:

* Data is sorted (or can be indexed)
* Performance matters (e.g., thousands of products)