WEEK1 – DATA STRUCTURES AND ALGORITHMS

EXERCISE2: **E-commerce Platform Search Function**

Main.java

package com.ecommerce.search;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(101, "Laptop", "Electronics"),

new Product(102, "Shoes", "Footwear"),

new Product(103, "Mobile", "Electronics"),

new Product(104, "T-Shirt", "Apparel"),

new Product(105, "Camera", "Electronics")

};

System.out.println("Linear Search Result:");

Product result1 = SearchUtil.linearSearch(products, "Mobile");

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

SearchUtil.sortProductsByName(products);

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

Product result2 = SearchUtil.binarySearch(products, "Mobile");

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

}

}

**Product.java**

package com.ecommerce.search;

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;

}

@Override

public String toString() {

return "Product ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

**SearchUtil.java**

package com.ecommerce.search;

import java.util.Arrays;

import java.util.Comparator;

public class SearchUtil {

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

for (Product p : products) {

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

return p;

}

}

return null;

}

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

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

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

else if (compare < 0) low = mid + 1;

else high = mid - 1;

}

return null;

}

public static void sortProductsByName(Product[] products) {

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

}

}

OUTPUT:

Linear Search Result:

Product ID: 103, Name: Mobile, Category: Electronics

Binary Search Result:

Product ID: 103, Name: Mobile, Category: Electronics

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

In the context of an e-commerce platform, the goal is to enable fast and efficient product searches as the number of products grows.

Algorithm Options:

**Linear Search:**

Time complexity: O(n)

Scans each product one by one.

Suitable for small or unsorted lists.

**Binary Search:**

Time complexity: O(log n)

Requires the product list to be sorted.

Cuts the search range in half at each step.

Binary Search is More Suitable than the linear search .It is much faster for large product lists especially when data is sorted by productId.

As the platform grows to thousands or millions of products, binary search will quickly find the target with very few comparisons. This improves user experience with faster search results.

**Binary search is more suitable for an e-commerce platform due to its logarithmic time complexity and scalability, ensuring fast and efficient search as the number of products increases.**