**Exercise 2 – E-commerce Platform Search Function**

# 1. Objective

To implement and compare linear and binary search algorithms for efficient product searching in an e-commerce platform.

# 2. Problem Statement / Scenario

You are developing a search feature for an e-commerce platform. The platform must efficiently find products based on user input. For smaller datasets, linear search may suffice, but for larger, sorted data, binary search offers better performance. The system should demonstrate both approaches using a Product class.

# 3. Approach / Steps

## 3.1 Understand Asymptotic Notation

• Big O notation describes the time complexity of algorithms as input size increases.

• Best Case: O(1) – Immediate match (e.g., first element).

• Average Case: O(n/2) – Match found mid-way (linear); O(log n) for binary.

• Worst Case: O(n) – Last element or not found (linear); O(log n) for binary.

## 3.2 Setup

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

## 3.3 Implementation

• Implement linearSearch and binarySearch methods in a SearchEngine class.

• Store products in an array for linear search and sort them for binary search.

# 4. Code

## 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;  
 }  
}

## SearchEngine.java

public class SearchEngine {  
 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 left = 0, right = products.length - 1;  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
 int cmp = name.compareToIgnoreCase(products[mid].productName);  
 if (cmp == 0) return products[mid];  
 else if (cmp < 0) right = mid - 1;  
 else left = mid + 1;  
 }  
 return null;  
 }  
}

## Main.java

import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(101, "Laptop", "Electronics"),  
 new Product(102, "Shoes", "Footwear"),  
 new Product(103, "Watch", "Accessories"),  
 new Product(104, "Mobile", "Electronics")  
 };  
  
 Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));  
  
 Product result1 = SearchEngine.linearSearch(products, "Watch");  
 System.out.println(result1 != null ? "Found (Linear): " + result1.productName : "Not found (Linear)");  
  
 Product result2 = SearchEngine.binarySearch(products, "Watch");  
 System.out.println(result2 != null ? "Found (Binary): " + result2.productName : "Not found (Binary)");  
 }  
}

# 5. Analysis

• Linear Search Time Complexity: O(n)

• Binary Search Time Complexity: O(log n) [on sorted data]

• Binary search is more efficient for larger datasets but requires the array to be sorted.

# 6. Conclusion

By implementing both linear and binary search, we can optimize product searching in an e-commerce platform. Binary search is preferred when working with sorted product data due to its logarithmic performance.

# 7. Output (Screenshot)

Below is the output of the program after performing both linear and binary searches:

