**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.

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.

1. **Analysis:**

○ Compare the time complexity of linear and binary search algorithms.

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

**SOURCE CODE:**

import java.util.Arrays;

import java.util.Comparator;

public class ECommerceSearch {

public static void main(String[] args) {

Product[] itemCatalog = {

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

new Product(105, "Novel Book", "Stationery"),

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

new Product(102, "Denim Shirt", "Clothing"),

new Product(104, "Running Shoes", "Footwear")

};

int targetProductIdValue = 103;

int linearResultIndex = SearchOperations.*linearSearch*(itemCatalog, targetProductIdValue);

System.*out*.println("Linear Search Index: " + linearResultIndex);

if (linearResultIndex != -1) {

System.*out*.println("Product found : " + itemCatalog[linearResultIndex].toString());

} else {

System.*out*.println("Product not found using Linear Search.");

}

Arrays.*sort*(itemCatalog, Comparator.*comparingInt*(Product::getProductIdentifier));

int binaryResultIndex = SearchOperations.*binarySearch*(itemCatalog, targetProductIdValue);

System.*out*.println("Binary Search Index: " + binaryResultIndex);

if (binaryResultIndex != -1) {

System.*out*.println("Product found : " + itemCatalog[binaryResultIndex].toString());

} else {

System.*out*.println("Product not found using Binary Search.");

}

}

}

class Product {

private int productIdentifier;

private String productDisplayName;

private String productCategory;

public Product(int productIdentifier, String productDisplayName, String productCategory) {

this.productIdentifier = productIdentifier;

this.productDisplayName = productDisplayName;

this.productCategory = productCategory;

}

public int getProductIdentifier() {

return productIdentifier;

}

@Override

public String toString() {

return productIdentifier + " - " + productDisplayName + " - " + productCategory;

}

}

class SearchOperations {

public static int linearSearch(Product[] catalogItems, int searchId) {

for (int i = 0; i < catalogItems.length; i++) {

if (catalogItems[i].getProductIdentifier() == searchId) {

return i;

}

}

return -1;

}

public static int binarySearch(Product[] catalogItems, int searchId) {

int startIdx = 0;

int endIdx = catalogItems.length - 1;

while (startIdx <= endIdx) {

int midIdx = startIdx + (endIdx - startIdx) / 2;

if (catalogItems[midIdx].getProductIdentifier() == searchId) {

return midIdx;

} else if (catalogItems[midIdx].getProductIdentifier() < searchId) {

startIdx = midIdx + 1;

} else {

endIdx = midIdx - 1;

}

}

return -1;

}

}

**OUTPUT :**   
