**Exercise -2 :**

**E-commerce Platform Search Function**

**Setup :**

public class Product {

//these are the instance variables that are used to give the overall structure of the program based on the SOLID principles int productId; String productName; String category;

public Product(int productId, String productName, String category) {

//here we have declared the constructors. the variables will be provided with the initial values for the code to begin with.

this.productId = productId;

this.productName = productName;

this.category = category;

}

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

**Implementing linear search and binary search algorithms and**

**storing the products in an array for linear search and a sorted array for binary search :**

public class Main {

// Linear Search this type of search can be carried on both (sorted as well as unsorted array) static Product linearSearch(Product products[], String name) {

//this is used to check for the element end it ignores upper and the lower cases .

for (Product p : products) {

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

}

return null;

}

// Binary Search and this type of search is to be conducted on a sorted array

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

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

while (low <= high) {

int mid = (low + high) / 2;

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

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

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

else high = mid - 1;

} return null;

}

//the final main function

public static void main(String args[]) {

//this is dataset that has been provided beforehand  
Product[] unsorted = {

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

new Product(2, "Shirt", "Clothing"),

new Product(3, "Laptop", "Electronics")

};

//this is used to sort the elements and store the values in the array

Product[] sorted = Arrays.copyOf(unsorted, unsorted.length);

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

Product result1 = linearSearch(unsorted, "Laptop");

Product result2 = binarySearch(sorted, "Laptop");

// this is used to print the overall value

System.out.println("Linear Search: " + (result1 != null ? result1 : "Not Found")); System.out.println("Binary Search: " + (result2 != null ? result2 : "Not Found")); }

}

**Compare the time complexity of linear and binary search algorithms. Discuss which algorithm is more suitable for your platform and why.**

**Linear Search :**

Linear search checks **each element one by one** until it finds the target or reaches the end.

* Best Case O(1): When the target is the first element itself.
* Average Case 0(n/2) or O(n): When the target is present somewhere in the middle.
* Worst Case O(n): When the target is placed in the end or is not present in the list.

**Binary Search :**

It is used for **sorted arrays and** it keeps dividing the array into halves till the final element is found.

* Best Case O(1): When the target is the middle element.
* Average Case 0(n/2) or O(n): When you have to split the list multiple times to find the element.
* Worst Case O(n): When the element is not present or is present at one of the extremes.

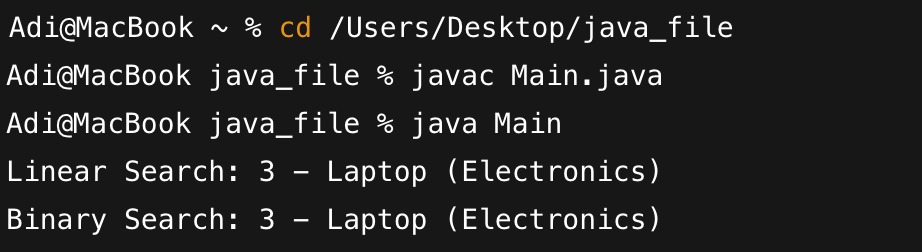
**Binary Search should be used when:**

* When the product list is large
* when we can keep the list sorted ie. sorted baesed on product name or id.
* and when we need **fast and repeated searches.**

**Linear Search should be used when:**

* When the overall product list is **small**
* when the data changes **frequently** and sorting each time is costly
* When we are searching for a product by **partial matches** or in **unsorted data**

**OUTPUTS –**

****