**Week 1**

**Hands-on :-**

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

1. **What is Big O Notation?**

Big O notation describes the upper bound of an algorithm's running time in terms of input size n. It helps compare the efficiency of algorithms.

**Describe the best, average, and worst-case scenarios for search operations?**

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| **Linear Search** | **O(1)** | **O(n)** | **O(n)** |
| **Binary Search** | **O(1)** | **O(log n)** | **O(log n)** |

1. **SETUP**

Code: -

1. 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;

}

@Override

public String toString() {

return "[" + productId + "] " + productName + " - " + category;

}

}

1. LinearSearch.java

public class LinearSearch {

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

for (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

}

1. BinarySearch

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

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

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

int left = 0, right = products.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

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

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

else if (cmp < 0) right = mid - 1;

else left = mid + 1;

}

return null;

}

}

1. Test.java

public class test {

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Mouse", "Accessories"),

new Product(103, "Shirt", "Apparel"),

new Product(104, "Camera", "Electronics"),

new Product(105, "Shoes", "Apparel")

};

String target = "Shirt";

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

Product found1 = LinearSearch.linearSearch(products, target);

System.out.println(found1 != null ? found1 : "Not Found");

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

Product found2 = BinarySearch.binarySearch(products, target);

System.out.println(found2 != null ? found2 : "Not Found");

}

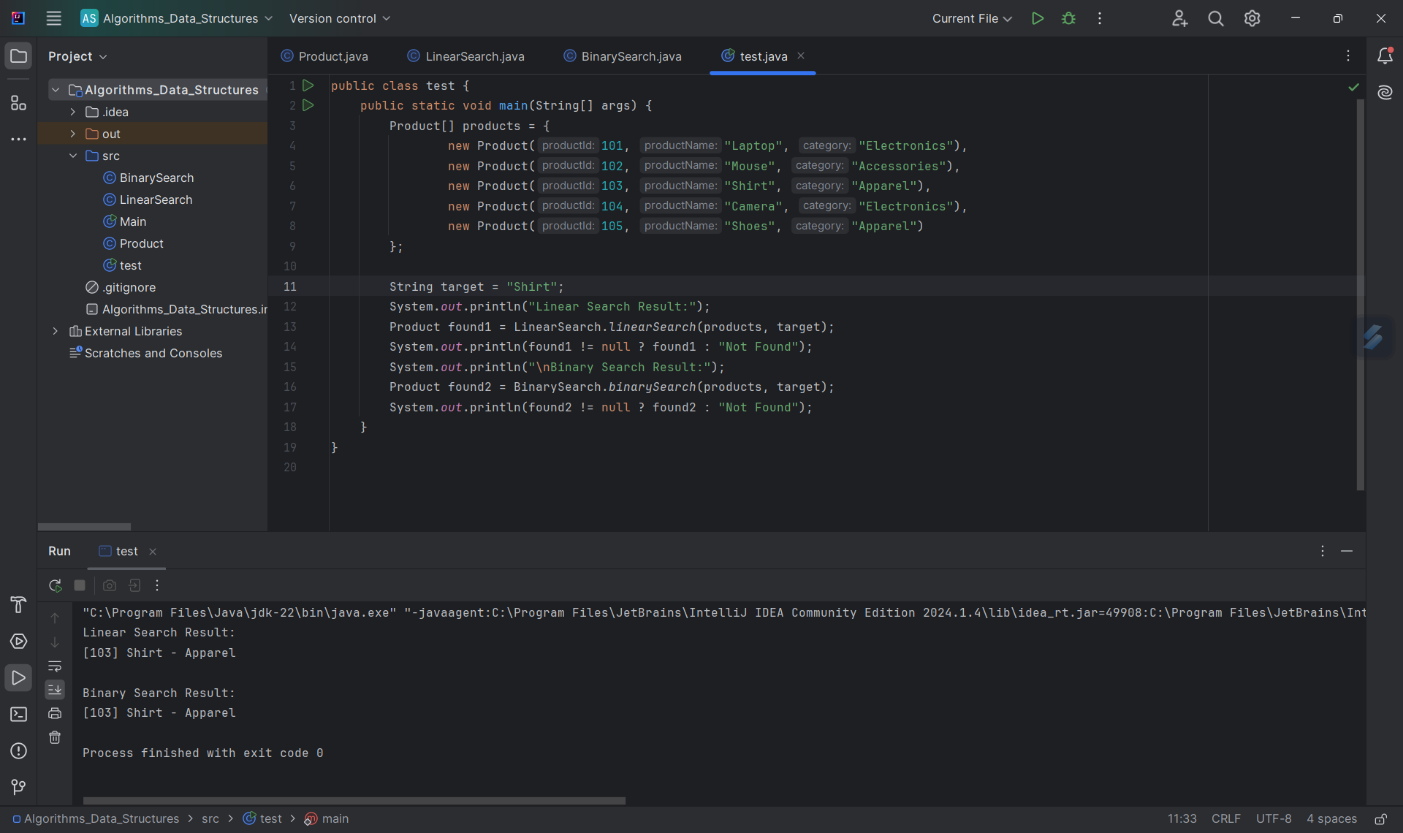
}

**Best algorithm suited for the e-commerce platform is Binary Search: -**

**Reason:**

1. **It offers faster performance (O(log n)) on large, sorted datasets.**
2. **Product listings can be easily sorted (e.g., by product name or ID).**
3. **E-commerce platforms usually contain thousands of products, so efficient search is critical for good user experience.**

**OUTPUT FOR THE CODE**

****