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

**Time Complexity :**

Big O notation helps us understand how efficient an algorithm is, especially when the number of items grows. It describes how the time taken by an algorithm increases as the input size increases, without getting into specific hardware or exact timing.

For example:

* **O(1)**: Constant time.
* **O(n)**: Linear time.
* **O(log n)**: Logarithmic time.

**Source Code :**

import java.util.\*;

class Product {

int productId;

String productName;

String category;

Product(int id, String name, String cat) {

productId = id;

productName = name;

category = cat;

}

public String toString() {

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

}

}

public class EcommerceSearchComparison {

public static void main(String[] args) {

Product[] products = {

new Product(1001, "Laptop", "Electronics"),

new Product(1002, "Shoes", "Footwear"),

new Product(1003, "Keyboard", "Electronics"),

new Product(1004, "Notebook", "Stationery"),

new Product(1005, "Jeans", "Clothing")

};

Scanner sc = new Scanner(System.in);

System.out.print("Enter product name to search: ");

String search = sc.nextLine();

System.out.println("\n--- Linear Search ---");

long startTime = System.nanoTime();

boolean foundLinear = false;

for (Product p : products) {

if (equalsIgnoreCase(p.productName, search)) {

System.out.println("Found: " + p);

foundLinear = true;

break;

}

}

if (!foundLinear) {

System.out.println("Product not found.");

}

long endTime = System.nanoTime();

System.out.println("Linear Search Time: " + (endTime - startTime) + " ns");

sortProductsByName(products);

System.out.println("\n--- Binary Search ---");

startTime = System.nanoTime();

boolean foundBinary = false;

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

while (low <= high) {

int mid = (low + high) / 2;

int cmp = compareIgnoreCase(search, products[mid].productName);

if (cmp == 0) {

System.out.println("Found: " + products[mid]);

foundBinary = true;

break;

} else if (cmp < 0) {

high = mid - 1;

} else {

low = mid + 1;

}

}

if (!foundBinary) {

System.out.println("Product not found.");

}

endTime = System.nanoTime();

System.out.println("Binary Search Time: " + (endTime - startTime) + " ns");

}

public static boolean equalsIgnoreCase(String a, String b) {

return toLower(a).equals(toLower(b));

}

public static int compareIgnoreCase(String a, String b) {

return toLower(a).compareTo(toLower(b));

}

public static String toLower(String s) {

char[] arr = s.toCharArray();

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

if (arr[i] >= 'A' && arr[i] <= 'Z') {

arr[i] = (char) (arr[i] + 32);

}

}

return new String(arr);

}

public static void sortProductsByName(Product[] arr) {

int n = arr.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - 1 - i; j++) {

if (compareIgnoreCase(arr[j].productName, arr[j + 1].productName) > 0) {

Product temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

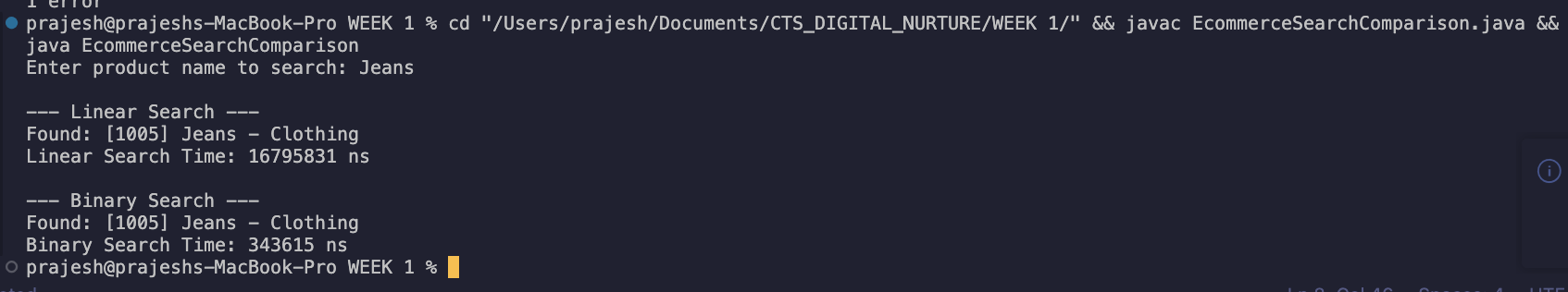
}

}

}

}

**Output :**

****