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

**Program:**

**1. ECommerceSearch.java**

import java.util.Arrays;

import java.util.Comparator;

class Product {

    int productId;

    String productName;

    String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

}

public class ECommerceSearch {

    public static void main(String[] args) {

        Product[] products = {

            new Product(201, "Television", "Electronics"),

            new Product(202, "Fruits", "Grocery"),

            new Product(203, "Watch", "Accessories"),

            new Product(204, "Water", "Beverages"),

            new Product(205, "Shoes", "Footwear")

        };

        String searchTerm = "Watch";

        long start=System.nanoTime();

        Product result1 = linearSearch(products, searchTerm);

        if (result1 != null)

            System.out.println("Linear Search: Found " + result1.productName + " in " + result1.category);

        else

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

        long end=System.nanoTime();

        System.out.println("Time take by Linear Search: "+(end-start)+" nano-seconds");

        System.out.println();

        start=System.nanoTime();

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

        Product result2 = binarySearch(products, searchTerm);

        if (result2 != null)

            System.out.println("Binary Search: Found " + result2.productName + " in " + result2.category);

        else

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

        end=System.nanoTime();

        System.out.println("Time take by Binary Search: "+(end-start)+" nano-seconds");

        System.out.println();

    }

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

        for (Product product : products) {

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

                return product;

            }

        }

        return null;

    }

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

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

        while (left <= right) {

            int mid = (left + right) / 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;

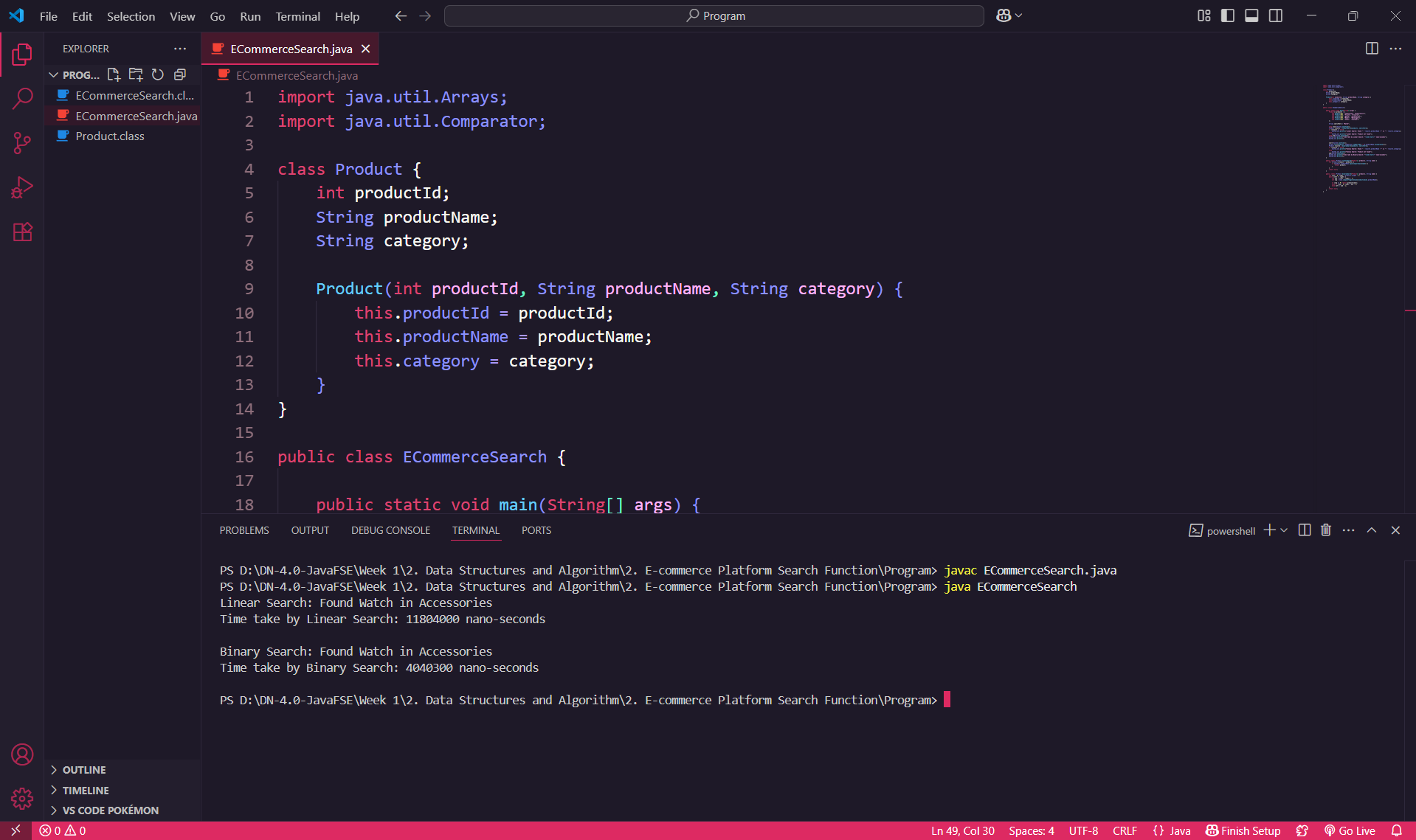
        }

        return null;

    }

}

**Output:**



**Questions to Answer:**

**1) Explain Big O notation and how it helps in analyzing algorithms.** Big O notation describes the performance or complexity of an algorithm in terms of input size. It tells us how the time or space used by an algorithm grows as the input grows. It helps compare algorithms and predict how they will scale in real-world scenarios.

**2) Describe the best, average, and worst-case scenarios for search operations.**

**I) Linear Search:**

Best case: O(1) → when the element is at the beginning.

Average case: O(n/2) ≈ O(n)

Worst case: O(n) → when the element is at the end or not present.

**II) Binary Search:**

Best case: O(1) → if the middle element is the target.

Average case: O(log n)

Worst case: O(log n)

**3) Compare the time complexity of linear and binary search algorithms.**

**I) Linear Search:** O(n) — It checks every element one by one.

**II) Binary Search:** O(log n) — It divides the array in half each time, making it much faster on large sorted data.

**4) Discuss which algorithm is more suitable for your platform and why.** For an e-commerce platform, binary search is more suitable because the product list can be very large. As long as the data is sorted, binary search offers faster results. However, linear search is simpler and works on unsorted data, which can be useful for small or dynamically changing product lists.