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

**1.Product.java**

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;

    }

}

**2. SearchFunction.java**

import java.util.Arrays;

public class SearchFunction

 {

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

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

            if (products[i].productName.equalsIgnoreCase(name)) {

                return i;

            }

        }

        return -1;

    }

    public static int 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 mid;

            else if (cmp < 0)

                low = mid + 1;

            else

                high = mid - 1;

        }

        return -1;

    }

    public static void main(String[] args) {

        Product[] products = {

            new Product(101, "Shoes", "Footwear"),

            new Product(102, "Phone", "Electronics"),

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

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

        };

        // Sort array by productName for binary search

Arrays.sort(products, (a, b) -> a.productName.compareToIgnoreCase(b.productName));

String searchKey = "Phone";

int linIndex = linearSearch(products, searchKey);

        int binIndex = binarySearch(products, searchKey);

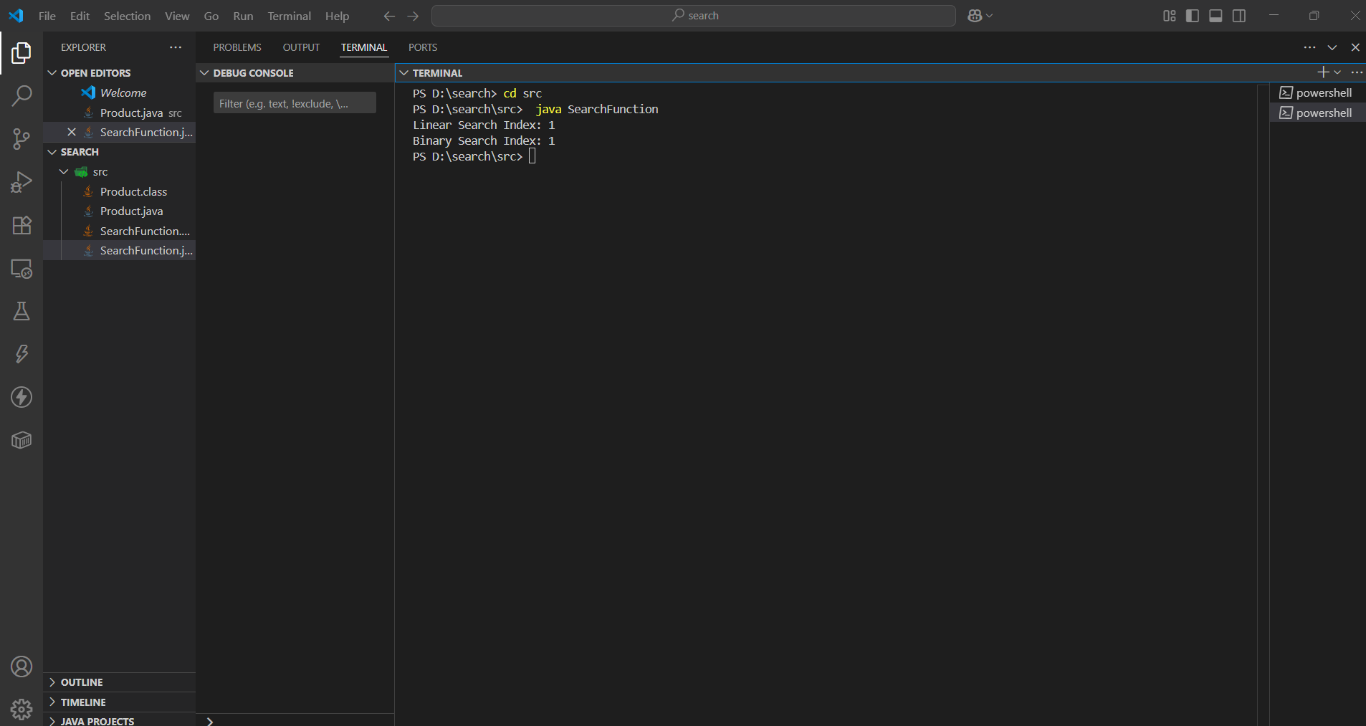
        System.out.println("Linear Search Index: " + linIndex);

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

    }

}

**Output**

****

**Conclusion:**

1. Binary Search is the better choice for an e-commerce platform because it offers much faster performance on large datasets.
2. Although it requires the product list to be sorted, this one-time sorting effort is worth it for the gain in speed during search operations.
3. Thus, for optimizing search functionality in an e-commerce system, Binary Search should be preferred, provided the products are stored in a sorted array or data structure

**Exercise 7: Financial Forecasting**

**FinancialForcast.java:**

public class FinancialForecast {

// Recursive method to compute future value

public static double calculateFutureValue(double presentValue, double rate, int years) {

// Base Case: if no years left, return present value

if (years == 0) {

return presentValue;

}

// Recursive Case: apply growth for one year and call for remaining years

return calculateFutureValue(presentValue \* (1 + rate), rate, years - 1);

}

public static void main(String[] args) {

double presentValue = 10000; // Initial investment

double annualRate = 0.08; // 8% annual growth rate

int years = 5; // Forecast for 5 years

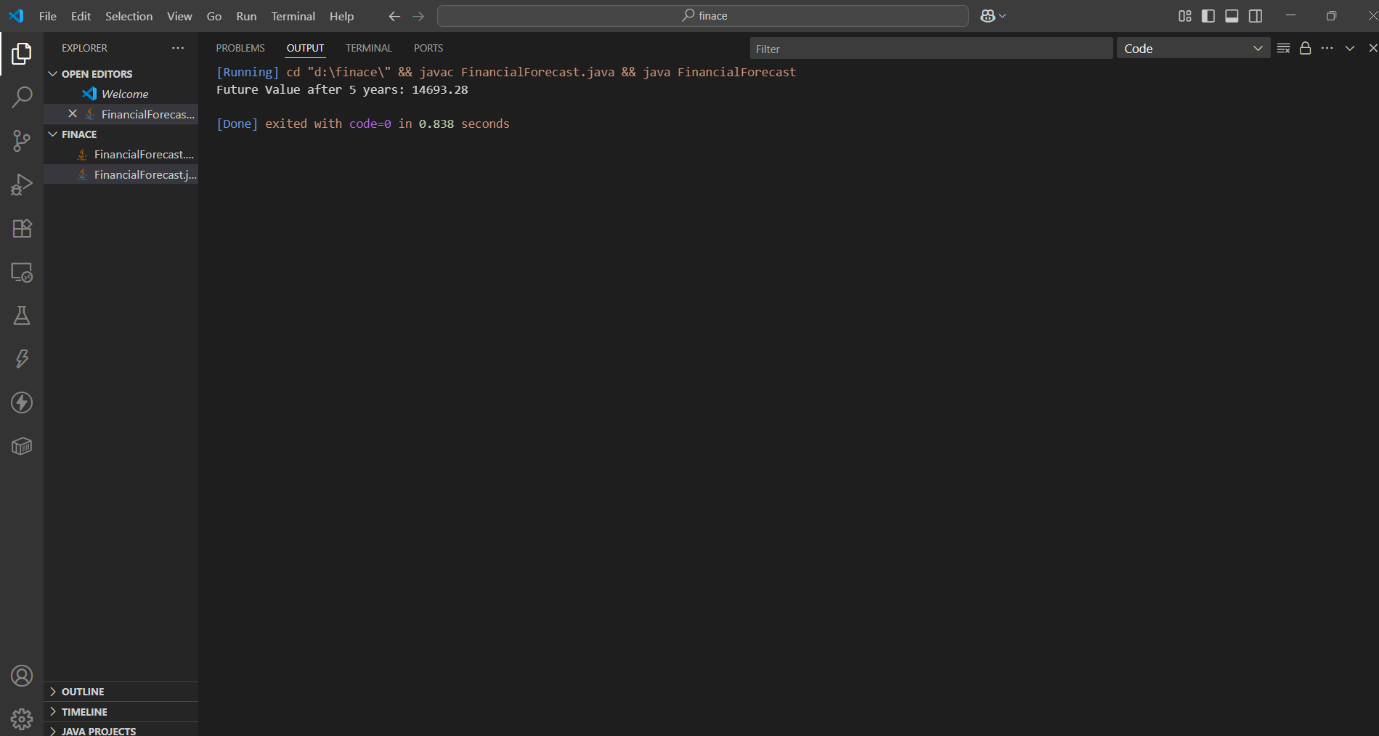
double futureValue = calculateFutureValue(presentValue, annualRate, years);

System.out.printf("Future Value after %d years: %.2f\n", years, futureValue);

}

}

**Output:**

****

**Analysis:**

1. The time complexity of this recursive function is **O(n)** because it performs **one recursive call per year** until the base case (year = 0) is reached.
2. Each call performs a **simple multiplication**, so it's efficient for small values of n