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

import java.util.Arrays;

import java.util.Comparator;

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;

    }

}

//Linearsearch.java

class LinearSearch {

    public static Product search(Product[] products, String key, String value) {

        for (Product product : products) {

            switch (key.toLowerCase()) {

                case "productid":

                    if (Integer.toString(product.productId).equals(value)) return product;

                    break;

                case "productname":

                    if (product.productName.equalsIgnoreCase(value)) return product;

                    break;

                case "category":

                    if (product.category.equalsIgnoreCase(value)) return product;

                    break;

            }

        }

        return null;

    }

}

//BinarySearch.java

class BinarySearch {

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

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

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

        while (left <= right) {

            int mid = (left + right) / 2;

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

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

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

            else right = mid - 1;

        }

        return null;

    }

}

//Main class

public class CommercePlatformSearchFunction {

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shampoo", "Personal Care"),

            new Product(103, "MacBook", "Electronics"),

            new Product(104, "T-Shirt", "Clothing")

        };

        System.out.println("\nLinear Search by productId");

        System.out.println(LinearSearch.search(products, "productId", "102"));

        System.out.println("\nLinear Search by category");

        System.out.println(LinearSearch.search(products, "category", "Electronics"));

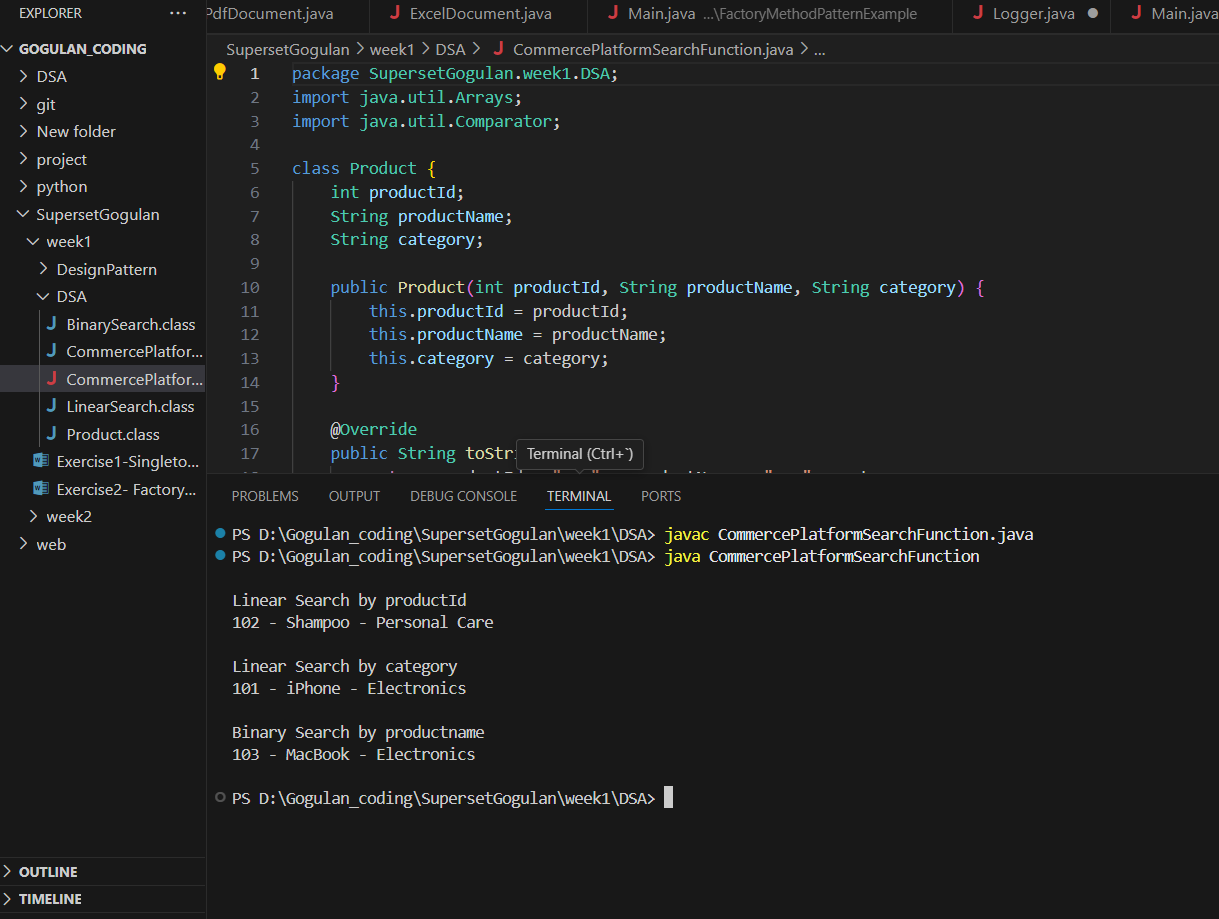
        System.out.println("\nBinary Search by productname");

        System.out.println(BinarySearch.searchByName(products, "MacBook") + "\n");

    }

}

**OUTPUT:**

****

**Exercise 7: Financial Forecasting**

import java.util.Scanner;

public class FinancialForecast {

    public static double forecastValue(double currentValue, double rate, int years) {

        if (years == 0) return currentValue;

        return forecastValue(currentValue, rate, years - 1) \* (1 + rate);

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("\nEnter Initial Amount : ");

        double initial = sc.nextDouble();

        System.out.println("Enter Growth Rate : ");

        double rate = sc.nextDouble();

        System.out.println("Enter No. of Years : ");

        int years = sc.nextInt();

        double result = forecastValue(initial, rate, years);

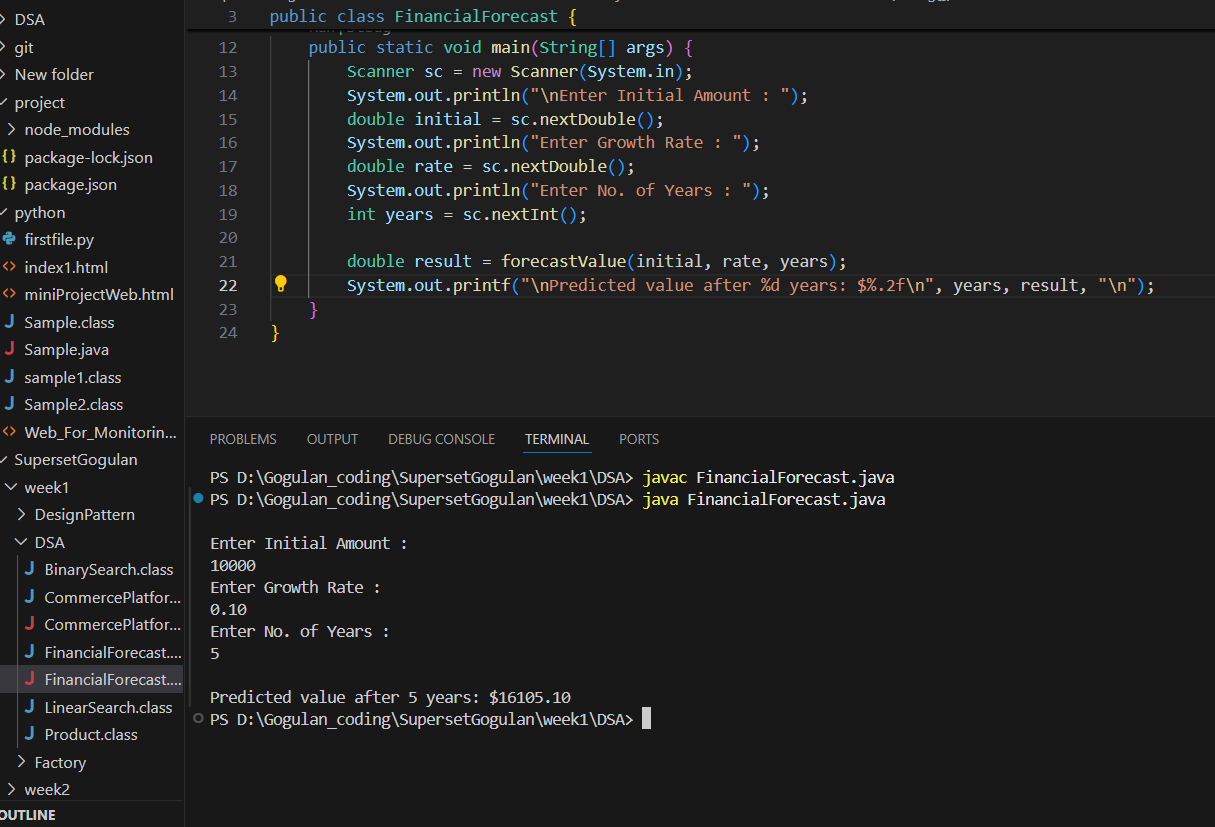
        System.out.printf("\nPredicted value after %d years: $%.2f\n", years, result, "\n");

    }

}

//Dynamic Programming is better efficient solution

**OUTPUT:**

****