**Week 1 : Algorithms Data Structures**

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

import java.util.Arrays;

import java.util.Scanner;

// Define the Product class with attributes productId, productName, and category

class Product {

    int productId;

    String productName;

    String category;

    // Constructor

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    // Getters

    public int getProductId() {

        return productId;

    }

    public String getProductName() {

        return productName;

    }

    public String getCategory() {

        return category;

    }

    @Override

    public String toString() {

        return "Product{" +

                "productId=" + productId +

                ", productName='" + productName + '\'' +

                ", category='" + category + '\'' +

                '}';

    }

}

public class EcommercePlatform {

    // Linear Search algorithm to find product by productName

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

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

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

                return i; // Return the index of the found product

            }

        }

        return -1; // Product not found

    }

    // Binary Search algorithm to find product by productName (requires sorted array)

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

        int low = 0;

        int high = products.length - 1;

        while (low <= high) {

            int mid = low + (high - low) / 2;

            int comparison = products[mid].getProductName().compareToIgnoreCase(productName);

            if (comparison == 0) {

                return mid; // Return the index of the found product

            } else if (comparison < 0) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return -1; // Product not found

    }

    public static void main(String[] args) {

        // Create Scanner object for user input

        Scanner scanner = new Scanner(System.in);

        // Sample product list (predefined for demonstration)

        Product[] products = {

            new Product(1, "Laptop", "Electronics"),

            new Product(2, "Smartphone", "Electronics"),

            new Product(3, "Tablet", "Electronics"),

            new Product(4, "Headphones", "Accessories"),

            new Product(5, "Smartwatch", "Accessories")

        };

        // Sort the products by productName for binary search

        Arrays.sort(products, (p1, p2) -> p1.getProductName().compareToIgnoreCase(p2.getProductName()));

        // Display the product list

        System.out.println("Product List:");

        for (Product product : products) {

            System.out.println(product);

        }

        // Get user input for the product name to search

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

        String productName = scanner.nextLine();

        // Perform linear search

        int linearIndex = linearSearch(products, productName);

        if (linearIndex != -1) {

            System.out.println("Linear Search: Found " + products[linearIndex]);

        } else {

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

        }

        // Perform binary search

        int binaryIndex = binarySearch(products, productName);

        if (binaryIndex != -1) {

            System.out.println("Binary Search: Found " + products[binaryIndex]);

        } else {

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

        }

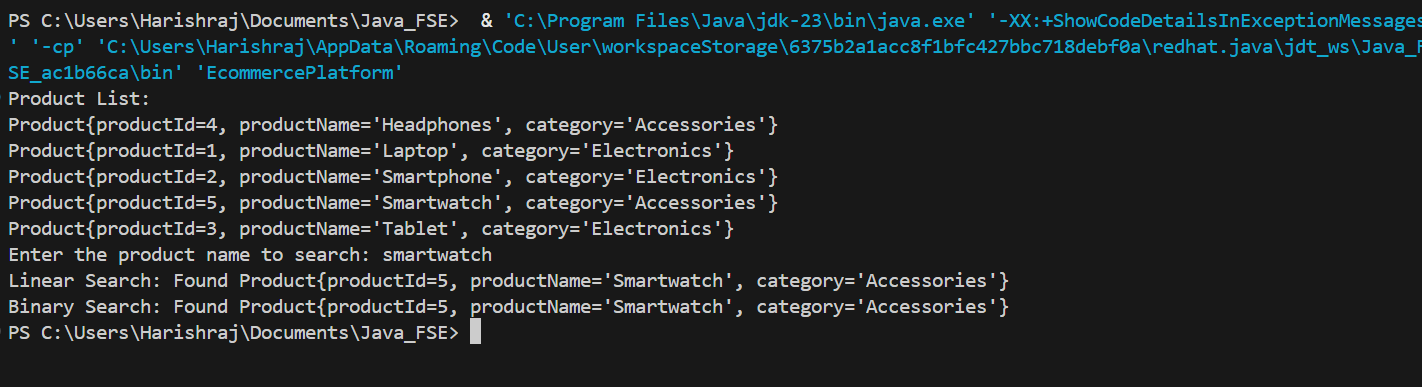
        // Close the scanner

        scanner.close();

    }

}

Output:



**Exercise 7: Financial Forecasting**

public class FinancialForecast {

    // Recursive method to calculate future value

    public static double futureValue(double principal, double rate, int years) {

        if (years == 0) {

            return principal; // Base case

        }

        return futureValue(principal, rate, years - 1) \* (1 + rate);

    }

    public static void main(String[] args) {

        double principal = 10000.0; // Initial investment

        double annualRate = 0.05;   // 5% annual growth

        int years = 5;

        double forecastedValue = futureValue(principal, annualRate, years);

        System.out.printf("Forecasted value after %d years: ₹%.2f\n", years, forecastedValue);

    }

}

Output:  
