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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

Java Code: Its written in Visual Studio Code

// EcommerceSearchExample.java

import java.util.Arrays;

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 + ")";

    }

}

public class EcommerceSearchExample {

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

        for (Product product : products) {

            if (product.productId == targetId) {

                return product;

            }

        }

        return null;

    }

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

        int left = 0;

        int right = products.length - 1;

        while (left <= right) {

            int mid = left + (right - left) / 2;

            if (products[mid].productId == targetId) {

                return products[mid];

            } else if (products[mid].productId < targetId) {

                left = mid + 1;

            } else {

                right = mid - 1;

            }

        }

        return null;

    }

    public static void sortProducts(Product[] products) {

        Arrays.sort(products, (p1, p2) -> Integer.compare(p1.productId, p2.productId));

    }

    public static void main(String[] args) {

        Product[] products = {

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

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

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

            new Product(102, "Book", "Education")

        };

        System.out.println("Linear Search for productId 104:");

        Product result1 = linearSearch(products, 104);

        System.out.println(result1 != null ? "Found: " + result1 : "Product not found");

        sortProducts(products);

        System.out.println("Binary Search for productId 104:");

        Product result2 = binarySearch(products, 104);

        System.out.println(result2 != null ? "Found: " + result2 : "Product not found");

        System.out.println("Time Complexity Analysis:");

        System.out.println("Linear Search: O(n) :Suitable for small or unsorted data.");

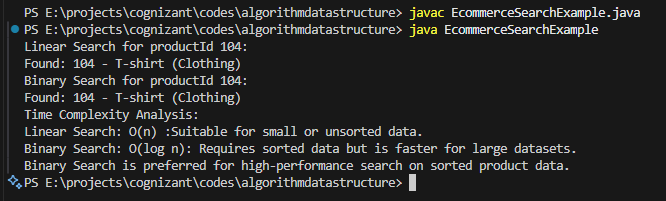
        System.out.println("Binary Search: O(log n): Requires sorted data but is faster for large datasets.");

        System.out.println("Binary Search is preferred for high-performance search on sorted product data.");

    }

}

Output Screenshot:



(Next Exercise is on the next Page)

**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

Java Code: It’s written in Visual Studio Code

//FinancialForecasting.java

public class FinancialForecasting {

    public static double forecastRecursive(double currentValue, double growthRate, int years) {

        if (years == 0) {

            return currentValue;

        }

        return forecastRecursive(currentValue \* (1 + growthRate), growthRate, years - 1);

    }

    public static double forecastIterative(double currentValue, double growthRate, int years) {

        for (int i = 0; i < years; i++) {

            currentValue \*= (1 + growthRate);

        }

        return currentValue;

    }

    public static void main(String[] args) {

        double initialValue = 10000.0;

        double growthRate = 0.08;

        int years = 5;

        double futureValueRecursive = forecastRecursive(initialValue, growthRate, years);

        System.out.printf("Future Value (Recursive): Rs.%.2f%n", futureValueRecursive);

        double futureValueIterative = forecastIterative(initialValue, growthRate, years);

        System.out.printf("Future Value (Iterative): Rs.%.2f%n", futureValueIterative);

        System.out.println();

        System.out.println("Time Complexity Analysis:");

        System.out.println("Recursive Forecast: O(n)");

        System.out.println("Iterative Forecast: O(n)");

        System.out.println("Iterative approach is more memory efficient");

    }

}

Output Screenshot:

