**Algorithms\_Data Structures Hands-on**

**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.

**Steps:**

1. **Understand Asymptotic Notation:**
   * Explain Big O notation and how it helps in analyzing algorithms.
   * Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
   * Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
   * Implement linear search and binary search algorithms.
   * Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
   * Compare the time complexity of linear and binary search algorithms.
   * Discuss which algorithm is more suitable for your platform and why.

import java.util.*\**;

public class Product {

    int productId;

    String productName;

    String category;

    public Product(int id, String name, String category) {

*this*.productId = id;

*this*.productName = name;

*this*.category = category;

    }

    public String toString() {

        return productId + " - " + productName + " (" + category + ")";

    }

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

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

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

                return products[i];

            }

        }

        return null;

    }

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

        int low = 0, high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

            int result = products[mid].productName.compareToIgnoreCase(targetName);

            if (result == 0)

                return products[mid];

            else if (result < 0)

                low = mid + 1;

            else

                high = mid - 1;

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shoes", "Fashion"),

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

            new Product(104, "Book", "Education")

        };

        Product result1 = linearSearch(products, "Shoes");

        System.out.println("Linear Search: " + (result1 != null ? result1 : "Not Found"));

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

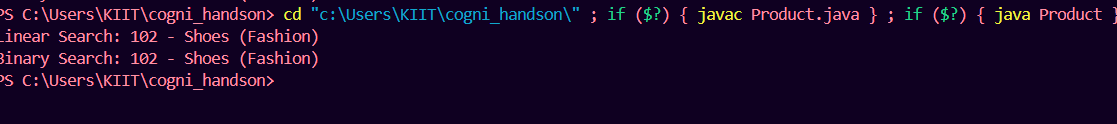
        Product result2 = binarySearch(products, "Shoes");

        System.out.println("Binary Search: " + (result2 != null ? result2 : "Not Found"));

    }

}

Output:



**Exercise 7: Financial Forecasting**

**Scenario:**

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

**Steps:**

1. **Understand Recursive Algorithms:**
   * Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   * Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   * Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   * Discuss the time complexity of your recursive algorithm.
   * Explain how to optimize the recursive solution to avoid excessive computation.

public class FinancialForecast {

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

        if (years == 0) {

            return currentValue;

        }

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

    }

    public static void main(String[] args) {

        double initialValue = 10000;

        double growthRate = 0.05;

        int years = 5;

        double futureValue = forecastValue(initialValue, growthRate, years);

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

    }

}

OUTPUT:

