Week 1 (Algorithms and Data Structure)

Exercise 2: E-commerce Platform Search Function:

import java.util.\*;

// Step 1: Product class

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 + ")";

}

}

// Step 2: Linear Search

class LinearSearch {

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

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

}

// Step 3: Binary Search

class BinarySearch {

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

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

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

while (low <= high) {

int mid = (low + high) / 2;

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

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

else if (compare < 0) low = mid + 1;

else high = mid - 1;

}

return null;

}

}

// Step 4: Main class to run everything

public class SearchDemo {

public static void main(String[] args) {

Product[] products = {

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

new Product(2, "T-shirt", "Clothing"),

new Product(3, "Shoes", "Footwear"),

new Product(4, "Laptop", "Electronics"),

new Product(5, "Book", "Stationery")

};

Scanner sc = new Scanner(System.in);

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

String target = sc.nextLine();

// Linear Search

System.out.println("\n-- Linear Search --");

Product result1 = LinearSearch.search(products, target);

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

// Binary Search

System.out.println("\n-- Binary Search --");

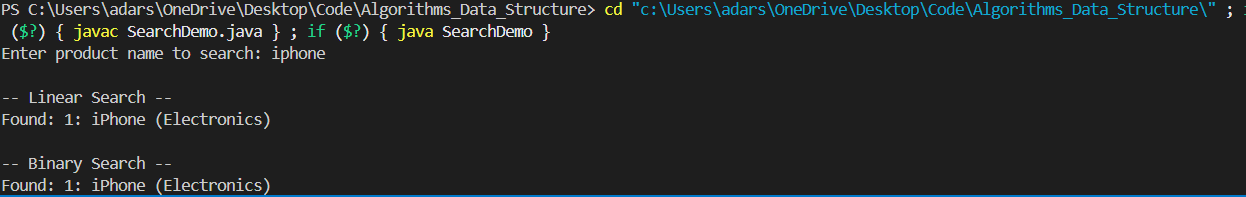
Product result2 = BinarySearch.search(products, target);

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

}

}

OUTPUT:



Exercise 7: Financial Forecasting:

import java.util.Scanner;

public class FinancialForecast {

// Recursive method to calculate future value

public static double futureValueRecursive(double initialValue, double growthRate, int years) {

if (years == 0) {

return initialValue;

}

return futureValueRecursive(initialValue, growthRate, years - 1) \* (1 + growthRate);

}

// Iterative method to calculate future value

public static double futureValueIterative(double initialValue, double growthRate, int years) {

double result = initialValue;

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

result \*= (1 + growthRate);

}

return result;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input

System.out.print("Enter the initial investment amount: ₹");

double initialValue = scanner.nextDouble();

System.out.print("Enter the annual growth rate (as percentage, e.g. 10 for 10%): ");

double ratePercent = scanner.nextDouble();

double growthRate = ratePercent / 100.0;

System.out.print("Enter the number of years to forecast: ");

int years = scanner.nextInt();

// Recursive calculation

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

// Iterative calculation

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

// Output

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

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

}

}

OUTPUT:

