Exercise 2: E-commerce Platform Search Function

Code:

import java.util.Arrays;

import java.util.Comparator;

public class ECommerceSearchExample {

static 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 (Product product : products) {

if (product.productName.equalsIgnoreCase(targetName)) {

return product;

}

}

return null;

}

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "T-Shirt", "Apparel"),

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

new Product(104, "Shoes", "Footwear"),

new Product(105, "Book", "Stationery")

};

System.out.println("Linear Search for 'Smartphone':");

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

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

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

System.out.println("\n Binary Search for 'Smartphone':");

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

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

System.out.println("\n Analysis:");

System.out.println("Linear Search - Time Complexity: O(n)");

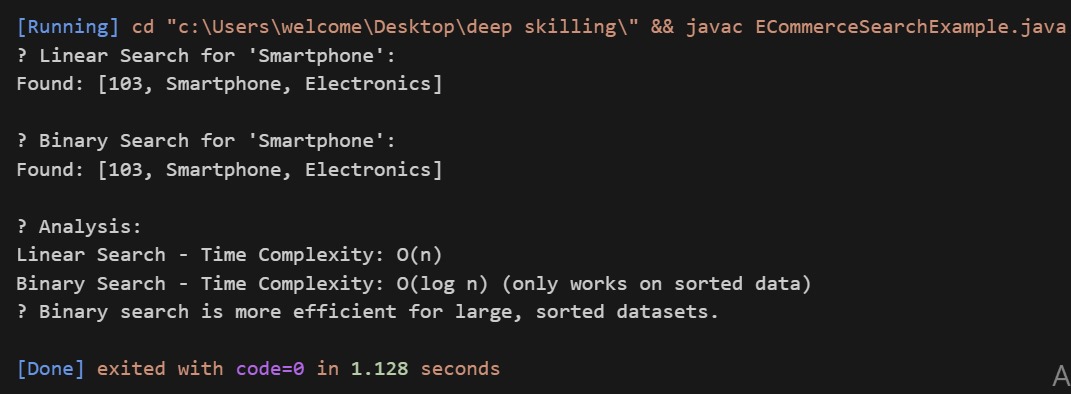
System.out.println("Binary Search - Time Complexity: O(log n) (only works on sorted data)");

System.out.println(" Binary search is more efficient for large, sorted datasets.");

}

}

Output:



Exercise 7: Financial Forecasting

Code:

public class FinancialForecastingExample {

public static double forecastValue(double presentValue, double growthRate, int years) {

if (years == 0) {

return presentValue;

}

return (1 + growthRate) \* forecastValue(presentValue, growthRate, years - 1);

}

public static void main(String[] args) {

double presentValue = 10000;

double annualGrowthRate = 0.08;

int forecastYears = 5;

double futureValue = forecastValue(presentValue, annualGrowthRate, forecastYears);

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

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

System.out.println("- Time Complexity: O(n) where n is the number of years");

System.out.println("- Each call recurses once until base case is hit");

System.out.println("\n Optimization Tip:");

System.out.println("- Use memoization or convert to an iterative approach to avoid deep recursion.");

}

}

Output:

