Exercise 2: E-commerce Platform Search Function

Code:

import java.util.Arrays;

import java.util.Comparator;

public class SearchOptimization {

static 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;

}

public String toString() {

return "[" + productId + ", " + productName + ", " + category + "]";

}

}

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, right = products.length - 1;

while (left <= right) {

int mid = (left + right) / 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 main(String[] args) {

Product[] products = {

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

new Product(205, "Shoes", "Footwear"),

new Product(303, "Book", "Stationery"),

new Product(404, "Mobile", "Electronics"),

new Product(109, "Chair", "Furniture")

};

int searchId = 303;

System.out.println("Linear Search:");

Product linearResult = linearSearch(products, searchId);

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

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

Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

System.out.println("Binary Search (on sorted data):");

Product binaryResult = binarySearch(products, searchId);

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

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

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

System.out.println("- Linear search is simple but slower for large datasets.");

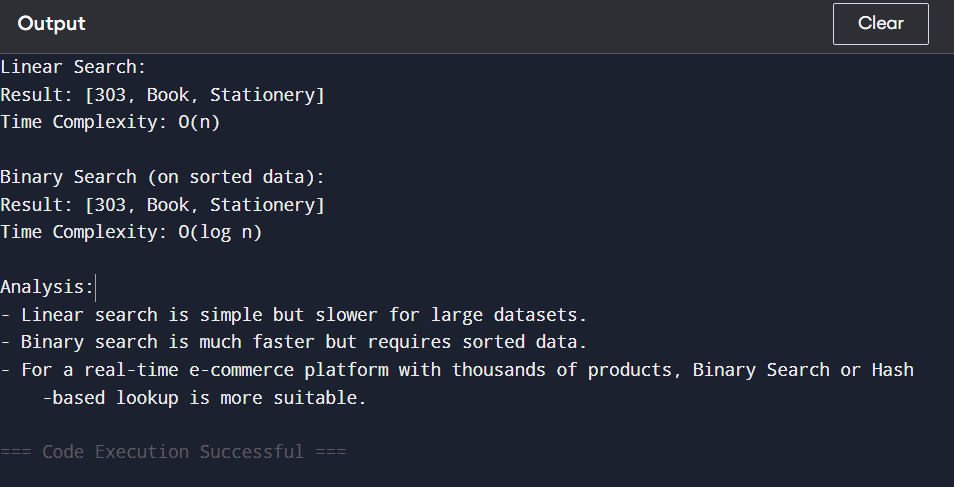
System.out.println("- Binary search is much faster but requires sorted data.");

System.out.println("- For a real-time e-commerce platform with thousands of products, Binary Search or Hash-based lookup is more suitable.");

}

}

Output:



Exercise 7: Financial Forecasting

Code:

public class FinancialForecast {

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

if (years == 0) {

return principal;

}

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

}

public static void main(String[] args) {

double principal = 10000.0;

double rate = 0.08;

int years = 5;

double futureValue = calculateFutureValue(principal, rate, years);

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

System.out.println("\nTime Complexity: O(n) because the function recurses once per year.");

System.out.println("For large 'n', recursion can lead to stack overflow or performance issues.");

System.out.println("optimization Tip: Use iteration or memoization for better performance.");

}

}

Output:

