Exercise 2: E-commerce Platform Search Function

Code:

import java.util.Arrays;

public class ECommerceSearchDemo {

    static class Product implements Comparable<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 + ")";

        }

        @Override

        public int compareTo(Product other) {

            return Integer.compare(this.productId, other.productId);

        }

    }

    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 low = 0;

        int high = products.length - 1;

        while (low <= high) {

            int mid = (low + high) / 2;

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

                return products[mid];

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

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

            new Product(105, "Shoes", "Footwear"),

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

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

            new Product(104, "Bag", "Accessories"),

            new Product(102, "Watch", "Accessories")

        };

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

        Product result1 = linearSearch(products, 103);

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

        Arrays.sort(products);

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

        Product result2 = binarySearch(products, 103);

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

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

        System.out.println("Linear Search: O(n) - Scans each item one by one.");

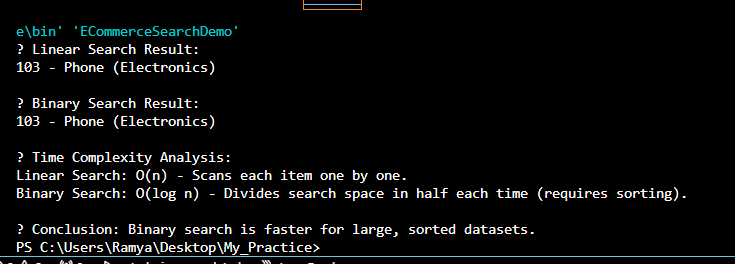
        System.out.println("Binary Search: O(log n) - Divides search space in half each time (requires sorting).");

        System.out.println("\n Conclusion: Binary search is faster for large, sorted datasets.");

    }

}

Output:



Exercise 7: Financial Forecasting

Code:

public class FinancialForecastingDemo {

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

        if (years == 0) {

            return initialValue;

        }

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

    }

    public static double calculateFutureValueIterative(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) {

        double initialValue = 10000;

        double growthRate = 0.08;

        int years = 5;

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

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

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

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

    }

}

Output:

