supersetID-6362856

**Algorithms\_Data Structures**

**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 productId, String productName, String category) {  this.productId = productId;  this.productName = productName;  this.category = category;  }   @Override  public String toString() {  return productId + " - " + productName + " (" + category + ")";  }  }   // Linear search: O(n)  public static Product linearSearch(Product[] products, int targetId) {  for (Product product : products) {  if (product.productId == targetId) {  return product;  }  }  return null;  }   // Binary search: O(log n)  public static Product binarySearch(Product[] products, int targetId) {  int left = 0;  int right = products.length - 1;   while (left <= right) {  int mid = left + (right - left) / 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(105, "Keyboard", "Electronics"),  new Product(101, "Shoes", "Fashion"),  new Product(104, "Watch", "Accessories"),  new Product(103, "Smartphone", "Electronics"),  new Product(102, "Laptop", "Electronics")  };   int targetId = 103;   // Linear Search  System.out.println("=== Linear Search ===");  Product result1 = linearSearch(products, targetId);  System.out.println(result1 != null ? result1 : "Product not found");   // Sort array by productId before binary search  Arrays.sort(products, Comparator.comparingInt(p -> p.productId));   // Binary Search  System.out.println("=== Binary Search ===");  Product result2 = binarySearch(products, targetId);  System.out.println(result2 != null ? result2 : "Product not found");  } } |

**Output-**

=== Linear Search ===

103 - Smartphone (Electronics)

=== Binary Search ===

103 - Smartphone (Electronics)

**Exercise 7: Financial Forecasting**

**Code-**

|  |
| --- |
| public class FinancialForecasting {      // Recursive method  public static double futureValue(double presentValue, double growthRate, int periods) {  if (periods == 0) {  return presentValue; // Base case: no periods left  }  return futureValue(presentValue, growthRate, periods - 1) \* (1 + growthRate);  }   // Iterative method   public static double futureValueIterative(double presentValue, double growthRate, int periods) {  double result = presentValue;  for (int i = 0; i < periods; i++) {  result \*= (1 + growthRate);  }  return result;  }   public static void main(String[] args) {  double presentValue = 1000.0;   double growthRate = 0.05;   int years = 10;   // recursive method  double recursiveResult = futureValue(presentValue, growthRate, years);  System.out.printf("Future value (recursive) after %d years: %.2f%n", years, recursiveResult);   // iterative method  double iterativeResult = futureValueIterative(presentValue, growthRate, years);  System.out.printf("Future value (iterative) after %d years: %.2f%n", years, iterativeResult);   } } |

**Output-**

Future value (recursive) after 10 years: 1628.89

Future value (iterative) after 10 years: 1628.89