Java Code for E-commerce Platform Search & Financial Forecasting

# 1. E-commerce Platform Search Function

import java.util.Arrays;  
import java.util.Comparator;  
  
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: " + productId + ", Name: " + productName + ", Category: " + category;  
 }  
}  
  
class SearchEngine {  
 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) {  
 Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));  
 int left = 0, right = products.length - 1;  
 while (left <= right) {  
 int mid = left + (right - left) / 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 class ECommerceSearch {  
 public static void main(String[] args) {  
 Product[] products = {  
 new Product(101, "Laptop", "Electronics"),  
 new Product(102, "Shoes", "Footwear"),  
 new Product(103, "Watch", "Accessories"),  
 new Product(104, "Mobile", "Electronics"),  
 new Product(105, "Bag", "Luggage")  
 };  
  
 String searchTerm = "Mobile";  
  
 System.out.println("Performing Linear Search for: " + searchTerm);  
 Product result1 = SearchEngine.linearSearch(products, searchTerm);  
 System.out.println(result1 != null ? result1 : "Product not found.");  
  
 System.out.println("\nPerforming Binary Search for: " + searchTerm);  
 Product result2 = SearchEngine.binarySearch(products, searchTerm);  
 System.out.println(result2 != null ? result2 : "Product not found.");  
 }  
}

# 2. Financial Forecasting using Recursion

public class FinancialForecasting {  
  
 // Recursive method to calculate future value  
 public static double predictFutureValue(double currentValue, double growthRate, int years) {  
 if (years == 0) {  
 return currentValue;  
 }  
 return predictFutureValue(currentValue \* (1 + growthRate), growthRate, years - 1);  
 }  
  
 // Optimized version using iteration (optional)  
 public static double predictFutureValueIterative(double currentValue, double growthRate, int years) {  
 for (int i = 0; i < years; i++) {  
 currentValue \*= (1 + growthRate);  
 }  
 return currentValue;  
 }  
  
 public static void main(String[] args) {  
 double currentValue = 1000.0;  
 double growthRate = 0.10; // 10%  
 int years = 5;  
  
 double futureValueRecursive = predictFutureValue(currentValue, growthRate, years);  
 double futureValueIterative = predictFutureValueIterative(currentValue, growthRate, years);  
  
 System.out.println("Future value (recursive): $" + futureValueRecursive);  
 System.out.println("Future value (iterative): $" + futureValueIterative);  
 }  
}