WEEK 1:

**Exercise 2: E-commerce Platform Search Function**

**Product.java**

package ecommerce;

public class Product {

private int productId;

private String productName;

private String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public String getProductName() {

return productName;

}

public String getCategory() {

return category;

}

*@Override*

public String toString() {

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

}

}

**SearchUtil.java**

package ecommerce;

import java.util.Arrays;

import java.util.Comparator;

public class SearchUtil {

// Linear Search by Product Name

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

for (Product product : products) {

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

return product;

}

}

return null;

}

// Binary Search by Product Name (Array must be sorted)

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

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

while (left <= right) {

int mid = left + (right - left) / 2;

int compare = products[mid].getProductName().compareToIgnoreCase(targetName);

if (compare == 0) {

return products[mid];

} else if (compare < 0) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return null;

}

// Helper to sort products by name

public static void sortProductsByName(Product[] products) {

Arrays.*sort*(products, Comparator.*comparing*(Product::getProductName, String.***CASE\_INSENSITIVE\_ORDER***));

}

}

**Main.java**

package ecommerce;

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Fashion"),

new Product(3, "Camera", "Electronics"),

new Product(4, "Book", "Education"),

new Product(5, "Smartphone", "Electronics")

};

String searchTarget = "Shoes";

// Linear Search

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

Product foundLinear = SearchUtil.*linearSearch*(products, searchTarget);

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

// Binary Search

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

SearchUtil.*sortProductsByName*(products); // Must sort before binary search

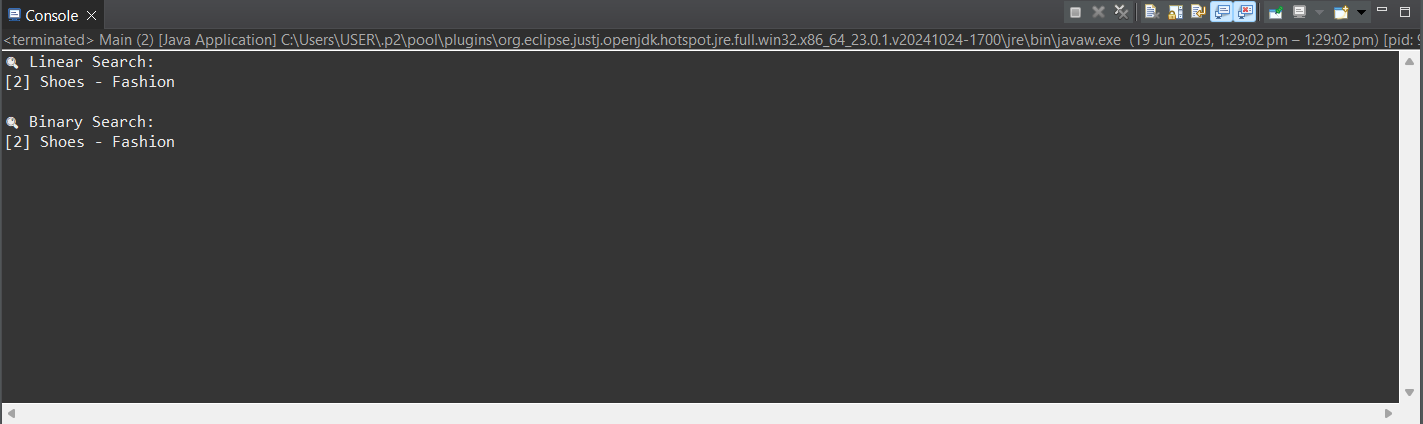
Product foundBinary = SearchUtil.*binarySearch*(products, searchTarget);

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

}

}

**Output:**

****

**Exercise 7: Financial Forecasting**

**FinancialForecast.java**

package finance;

public class FinancialForecast {

// Recursive method to calculate future value

public static double forecastRecursive(double currentValue, double growthRate, int years) {

if (years == 0) {

return currentValue; // base case

}

return *forecastRecursive*(currentValue, growthRate, years - 1) \* (1 + growthRate);

}

// Optimized version using tail recursion (or even iteration)

public static double forecastIterative(double currentValue, double growthRate, int years) {

for (int i = 0; i < years; i++) {

currentValue \*= (1 + growthRate);

}

return currentValue;

}

}

**Main.java**

package finance;

public class Main {

public static void main(String[] args) {

double initialValue = 10000;

double growthRate = 0.08; // 8% annual growth

int years = 5;

System.***out***.println(" Forecast using recursion:");

double resultRecursive = FinancialForecast.*forecastRecursive*(initialValue, growthRate, years);

System.***out***.printf("Value after %d years: ₹%.2f%n", years, resultRecursive);

System.***out***.println("\n Forecast using iteration:");

double resultIterative = FinancialForecast.*forecastIterative*(initialValue, growthRate, years);

System.***out***.printf("Value after %d years: ₹%.2f%n", years, resultIterative);

}

}

**Output:**

****