**ALGORITHMS AND DATA STRUCTURES**

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

**Scenario:**

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

**Product.java**

**package** ecommerce;

**public** **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** "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

**SearchFunctions.java**

**package** ecommerce;

**public** **class** SearchFunctions {

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

**for** (Product p : products) {

**if** (p.productName.equalsIgnoreCase(targetName)) {

**return** p;

}

}

**return** **null**;

}

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

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

**while** (left <= right) {

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

**int** result = products[mid].productName.compareToIgnoreCase(targetName);

**if** (result == 0) **return** products[mid];

**else** **if** (result < 0) left = mid + 1;

**else** right = mid - 1;

}

**return** **null**;

}

}

**Main.java**

package ecommerce;

import java.util.Arrays;

import java.util.Comparator;

public class Main {

public static void main(String[] args) {

Product[] products = {

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

new Product(102, "Shirt", "Clothing"),

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

new Product(104, "Book", "Stationery"),

new Product(105, "Tablet", "Electronics")

};

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

Product result1 = SearchFunctions.linearSearch(products, "Mobile");

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

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

Arrays.sort(products, Comparator.comparing(p -> p.productName));

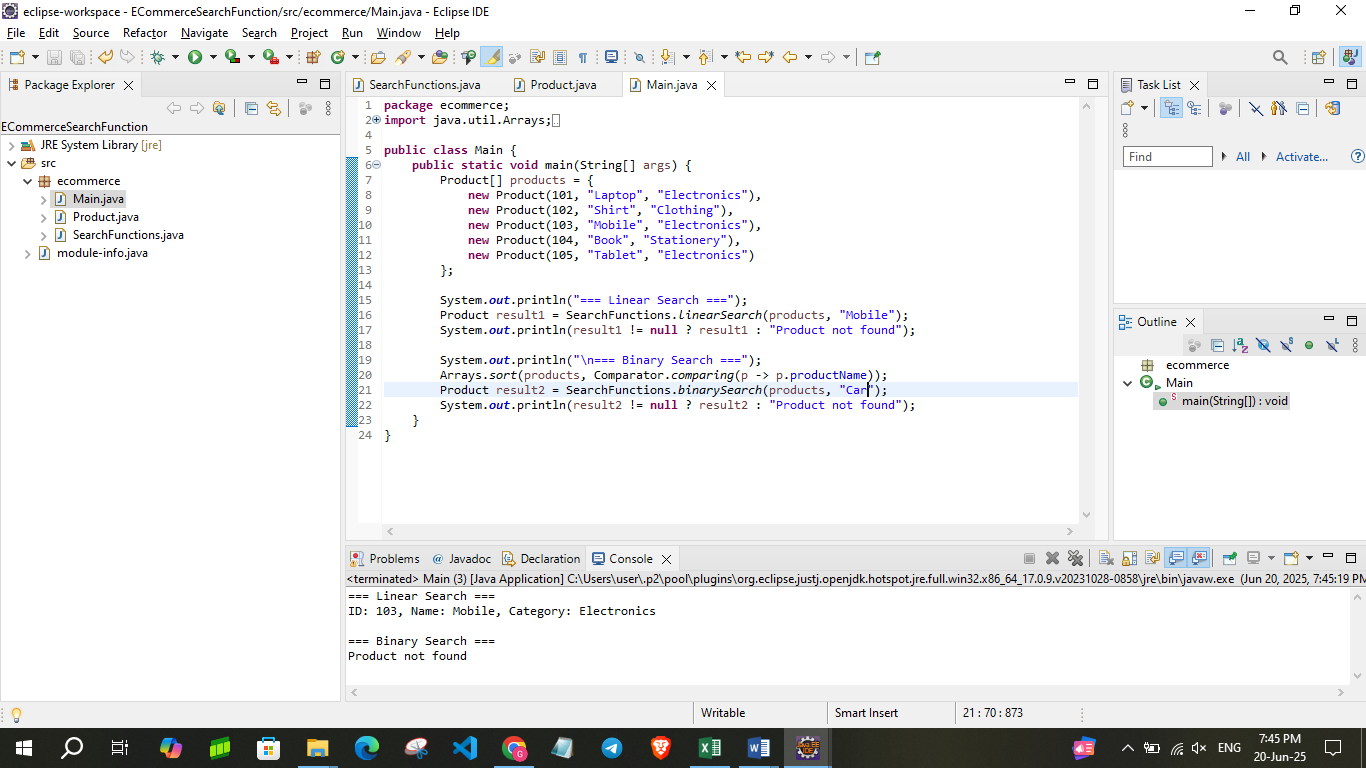
Product result2 = SearchFunctions.binarySearch(products, "Mobile");

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

}

}

**OUTPUT:**



**Exercise 7: Financial Forecasting**

**Scenario:**

You are developing a financial forecasting tool that predicts future values based on past data.

**Forecast.java**

**package** finance;

**public** **class** Forecast {

**public** **static** **double** futureValue(**double** presentValue, **double** growthRate, **int** years) {

**if** (years == 0) {

**return** presentValue;

}

**return** *futureValue*(presentValue, growthRate, years - 1) \* (1 + growthRate);

}

**public** **static** **double** futureValueMemo(**double** presentValue, **double** growthRate, **int** years, **double**[] memo) {

**if** (years == 0) **return** presentValue;

**if** (memo[years] != 0) **return** memo[years];

memo[years] = *futureValueMemo*(presentValue, growthRate, years - 1, memo) \* (1 + growthRate);

**return** memo[years];

}

}

**Main.java**

**package** finance;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**double** presentValue = 10000;

**double** rate = 0.08;

**int** years = 5;

System.***out***.println("Recursive Forecast (Simple):");

**double** result = Forecast.*futureValue*(presentValue, rate, years);

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

System.***out***.println("\nRecursive Forecast (With Memoization):");

**double**[] memo = **new** **double**[years + 1];

**double** resultMemo = Forecast.*futureValueMemo*(presentValue, rate, years, memo);

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

}

}

**OUTPUT:**

