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

// Product.java

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;

}

@Override

public String toString() {

return "Product ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

//Main.java

import java.util.Arrays;

import java.util.Comparator;

public class Main {

// Linear search by productName

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

for (Product p : products) {

if (p.productName.equalsIgnoreCase(name)) {

return p;

}

}

return null;

}

// Binary search (products must be sorted by productName)

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

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

while (left <= right) {

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

int cmp = products[mid].productName.compareToIgnoreCase(name);

if (cmp == 0) return products[mid];

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

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

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

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

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

};

String searchName = "Smartphone";

// Linear Search

Product foundLinear = linearSearch(products, searchName);

System.out.println("Linear Search result: " + (foundLinear != null ? foundLinear : "Not Found"));

// Sort array for binary search

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

// Binary Search

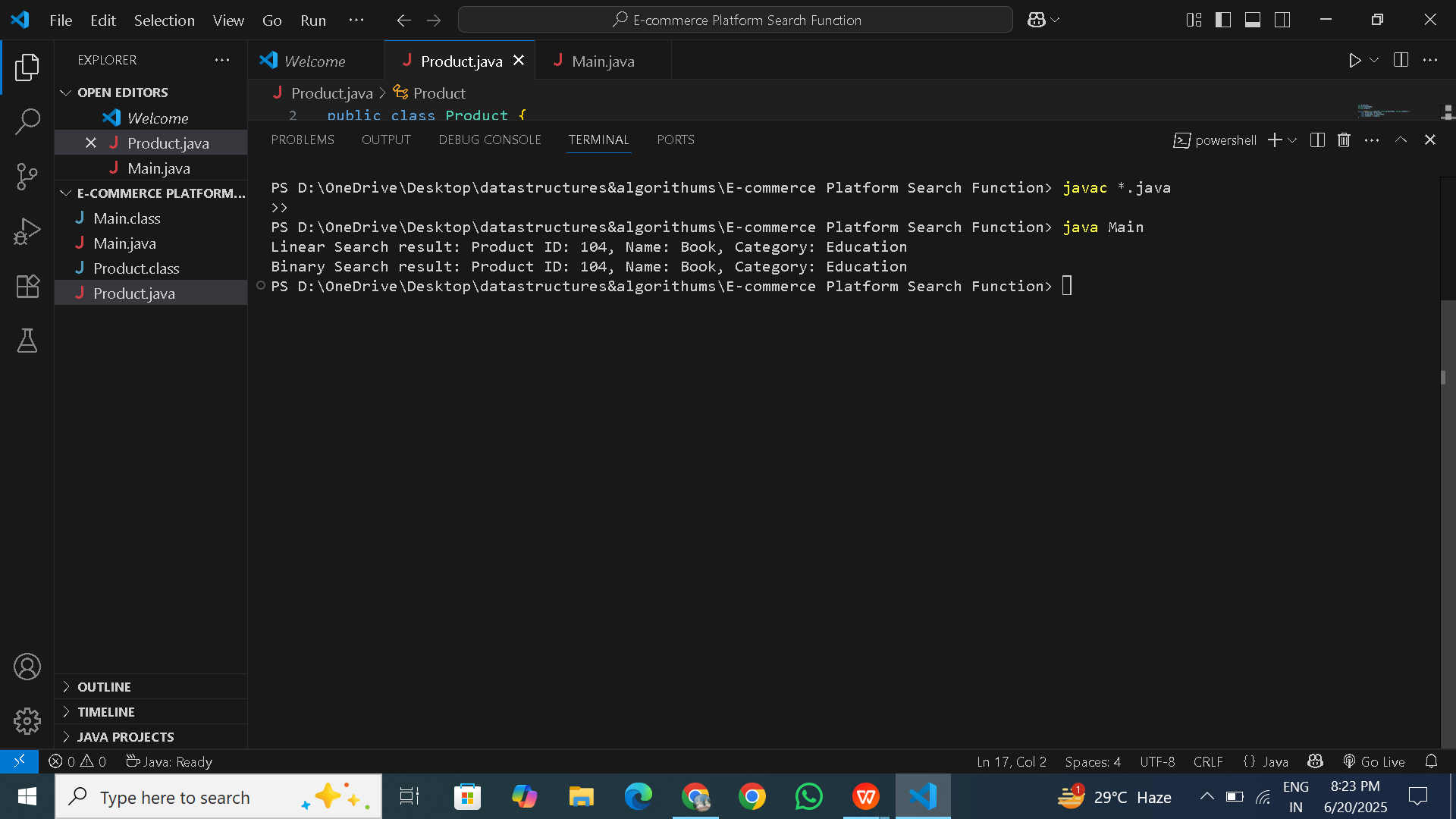
Product foundBinary = binarySearch(products, searchName);

System.out.println("Binary Search result: " + (foundBinary != null ? foundBinary : "Not Found"));

}

}

Output:



**Exercise 3: Sorting Customer Orders**

//Order.java

public class Order {

int orderId;

String customerName;

double totalPrice;

public Order(int orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

@Override

public String toString() {

return "Order ID: " + orderId + ", Customer: " + customerName + ", Total Price: ₹" + totalPrice;

}

}

//SortOrders.java

public class SortOrders {

// Bubble Sort by totalPrice

public static void bubbleSort(Order[] orders) {

int n = orders.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (orders[j].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

}

}

}

}

// Quick Sort by totalPrice

public static void quickSort(Order[] orders, int low, int high) {

if (low < high) {

int pi = partition(orders, low, high);

quickSort(orders, low, pi - 1);

quickSort(orders, pi + 1, high);

}

}

private static int partition(Order[] orders, int low, int high) {

double pivot = orders[high].totalPrice;

int i = low - 1;

for (int j = low; j < high; j++) {

if (orders[j].totalPrice <= pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

orders[i + 1] = orders[high];

orders[high] = temp;

return i + 1;

}

// Print utility

public static void printOrders(Order[] orders) {

for (Order o : orders) {

System.out.println(o);

}

}

}

//Main.java

public class Main {

public static void main(String[] args) {

Order[] orders = {

new Order(201, "Alice", 2500.00),

new Order(202, "Bob", 1200.50),

new Order(203, "Charlie", 4200.75),

new Order(204, "Daisy", 3100.00),

new Order(205, "Ethan", 1500.00)

};

// Bubble Sort

System.out.println("🔵 Bubble Sort:");

Order[] bubbleSorted = orders.clone();

SortOrders.bubbleSort(bubbleSorted);

SortOrders.printOrders(bubbleSorted);

// Quick Sort

System.out.println("\n🔴 Quick Sort:");

Order[] quickSorted = orders.clone();

SortOrders.quickSort(quickSorted, 0, quickSorted.length - 1);

SortOrders.printOrders(quickSorted);

}

}

**Output:**

