**Cognizant Deep Skilling - Digital Nurture 4.0**

**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.

**Java Code**

**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;

}

public String toString() {

return productId + " - " + productName + " (" + category + ")";

}

}

**Main.java**

import java.util.Arrays;

import java.util.Comparator;

public class Main {

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

for (Product p : products) {

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

return p;

}

}

return null;

}

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

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

while (left <= right) {

int mid = (left + right) / 2;

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

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

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

else right = mid - 1;

}

return null;

}

public static void main(String[] args) {

Product[] products = {

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

new Product(2, "TShirt", "Clothing"),

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

new Product(4, "Shoes", "Footwear")

};

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

String searchFor = "TV";

Product linearResult = linearSearch(products, searchFor);

System.out.println("Linear Search: " + (linearResult != null ? linearResult : "Not found"));

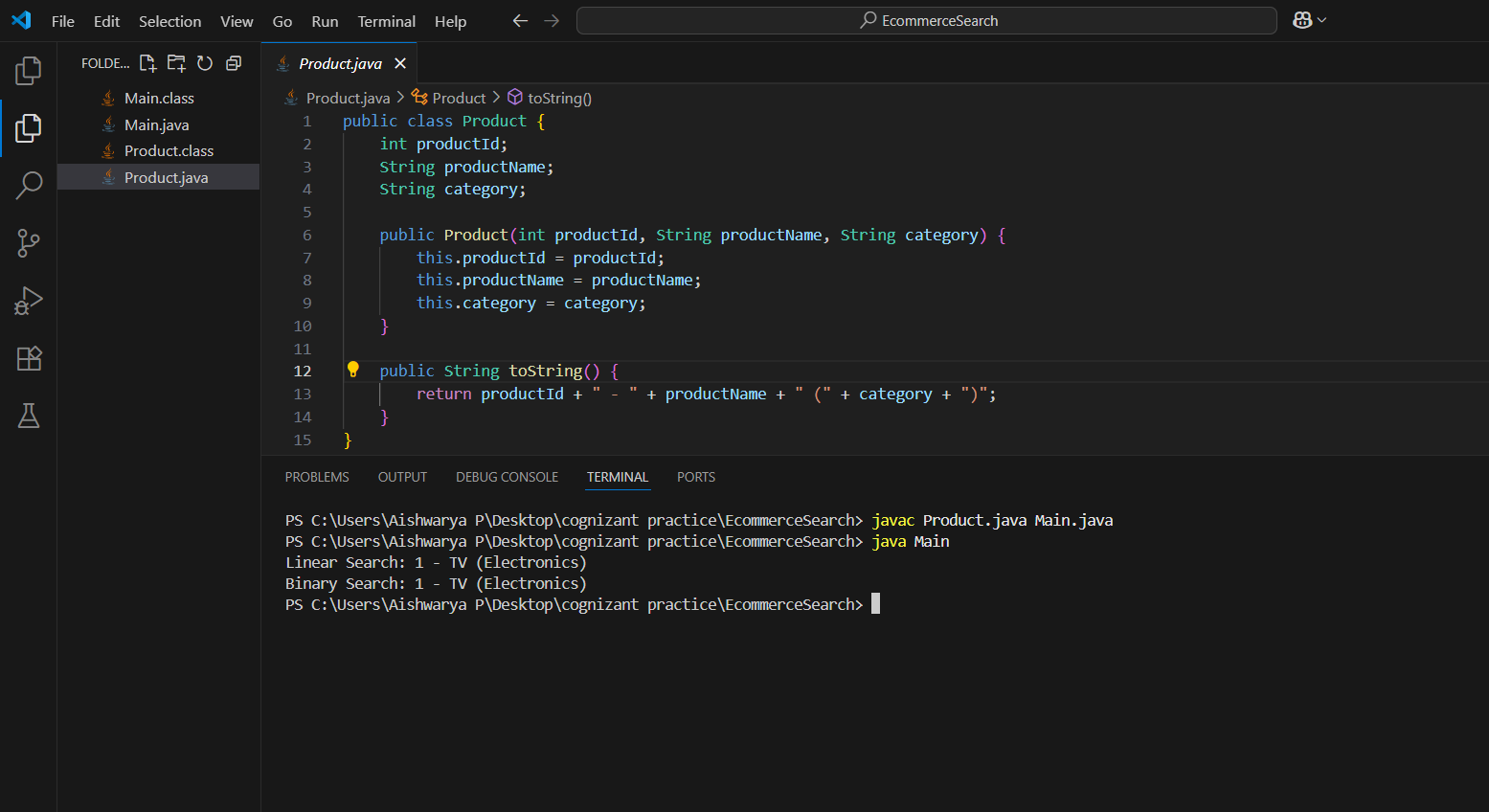
Product binaryResult = binarySearch(products, searchFor);

System.out.println("Binary Search: " + (binaryResult != null ? binaryResult : "Not found"));

}

}

**Output**



**Exercise 7: Financial Forecasting Tool**

**Scenario:**

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

**Java Code**

**Forecast.java**

public class Forecast {

public static double futureValueRecursive(double principal, double rate, int years) {

if (years == 0)

return principal;

return futureValueRecursive(principal, rate, years - 1) \* (1 + rate);

}

public static double futureValueMemo(double principal, double rate, int years, double[] memo) {

if (years == 0)

return principal;

if (memo[years] != 0)

return memo[years];

memo[years] = futureValueMemo(principal, rate, years - 1, memo) \* (1 + rate);

return memo[years];

}

}

**Main.java**

public class Main {

    public static void main(String[] args) {

        double principal = 1000;

        double rate = 0.05;

        int years = 7;

        double fvRecursive = Forecast.futureValueRecursive(principal, rate, years);

        System.out.printf("Future Value (Recursive): %.2f%n", fvRecursive);

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

        double fvMemo = Forecast.futureValueMemo(principal, rate, years, memo);

        System.out.printf("Future Value (Memoized): %.2f%n", fvMemo);

    }

}

**Output**

