Data structures and Algorithms

Exercise 2: E-commerce Platform Search Function

Program code:

package mypack;

import java.util.Arrays;

import java.util.Comparator;

public class EcommerceSearchExample {

    static class Product {

        int productId;

        String productName;

        String category;

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

            this.productId = productId;

            this.productName = productName;

            this.category = category;

        }

public String toString() {

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

        }

    }

    static Product linearSearch(Product[] products, int targetId) {

        for (Product product : products) {

            if (product.productId == targetId) {

                return product;

            }

        }

        return null;

    }

    static Product binarySearch(Product[] products, int targetId) {

        int low = 0, high = products.length - 1;

while (low <= high) {

            int mid = (low + high) / 2;

            if (products[mid].productId == targetId) {

                return products[mid];

            } else if (products[mid].productId < targetId) {

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(102, "Shoes", "Footwear"),

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

            new Product(104, "Watch", "Accessories"),

            new Product(103, "Book", "Education")

        };

System.out.println("\n Linear Search :");

        Product linearResult = linearSearch(products, 104);

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

        Arrays.sort(products, Comparator.comparingInt(p -> p.productId));

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

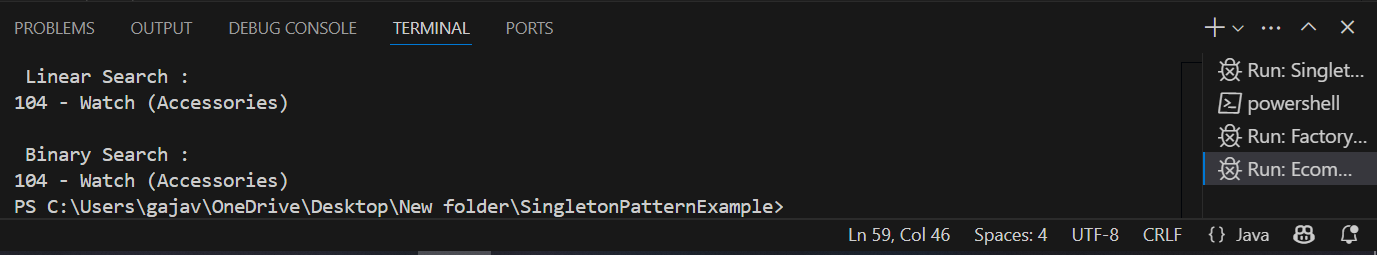
        Product binaryResult = binarySearch(products, 104);

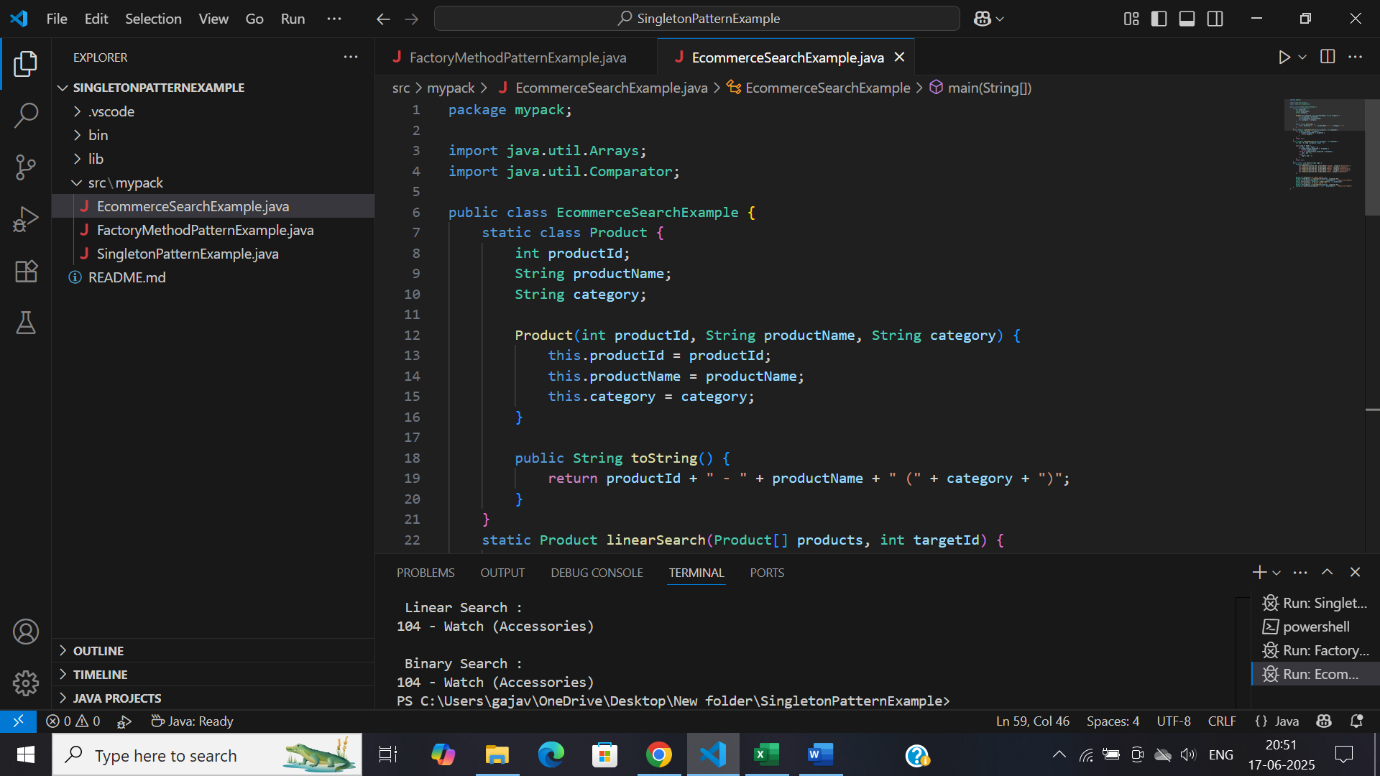
        System.out.println(binaryResult != null ? binaryResult : "Product not found");

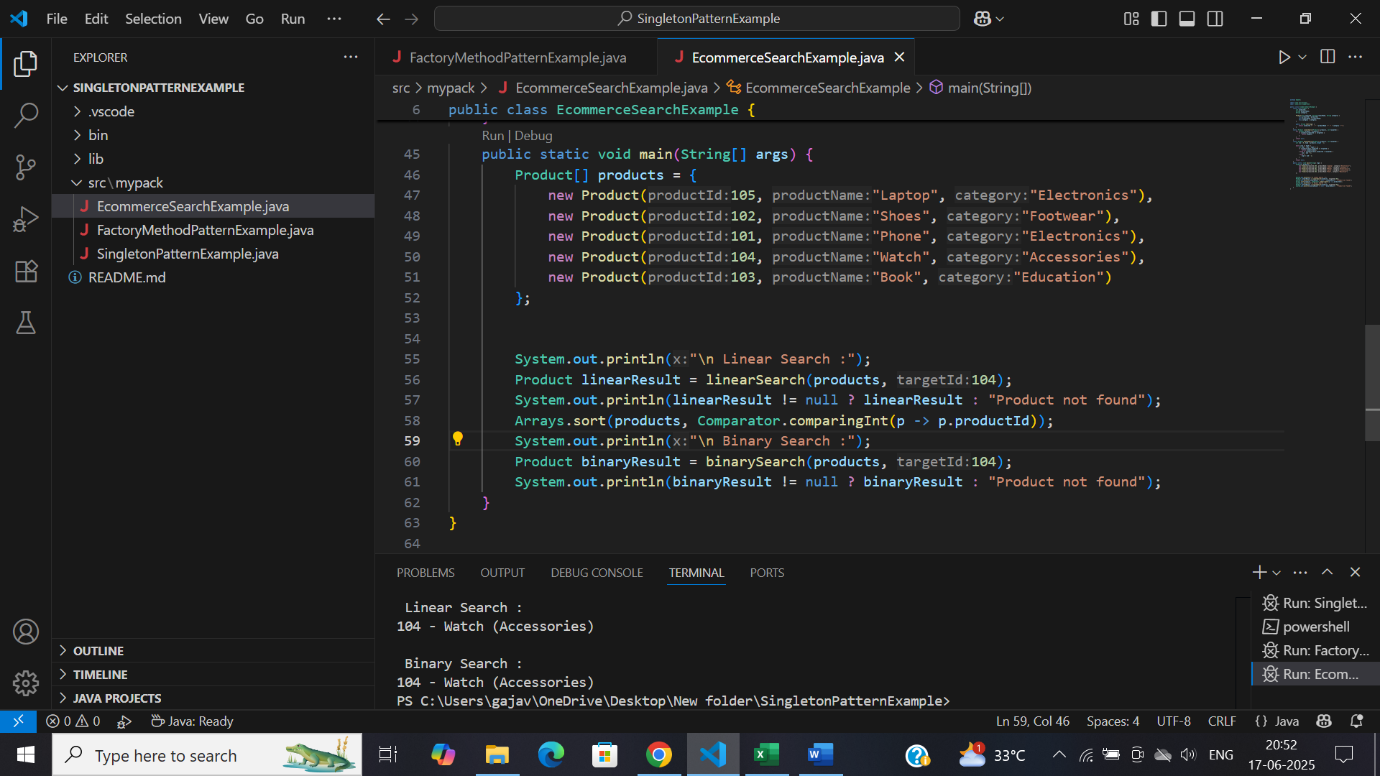
    }

}

OUTPUT:







Exercise 7: Financial Forecasting

Program code:

package mypack;

public class FinancialForecastTool {

    public static double findFutureAmount(double now, double rate, int years) {

        if (years == 0) {

            return now;

        }

        return (1 + rate) \* findFutureAmount(now, rate, years - 1);

    }

    public static double findFutureAmountMemo(double now, double rate, int years, double[] memo) {

        if (years == 0) return now;

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

        memo[years] = (1 + rate) \* findFutureAmountMemo(now, rate, years - 1, memo);

        return memo[years];

    }

public static void main(String[] args) {

        double now = 1000.0;

        double rate = 0.07;

        int years = 10;

System.out.println("Financial Forecast Tool\n");

double result1 = findFutureAmount(now, rate, years);

        System.out.printf("Recursive Forecast (Year %d): %.2f\n", years, result1);

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

        double result2 = findFutureAmountMemo(now, rate, years, memo);

        System.out.printf("Optimized Forecast with Memoization (Year %d): %.2f\n", years, result2);

    }

}

OUTPUT:

