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

public class BinarySearch {

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

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

        while (left <= right) {

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

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

                return mid;

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

                left = mid + 1;

            } else {

                right = mid - 1;

            }

        }

        return -1; // Not found

    }

}

public class LinearSearch {

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

        for (int i = 0; i < products.length; i++) {

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

                return i;

            }

        }

        return -1; // Not found

    }

}

import java.util.Arrays;

import java.util.Comparator;

public class Main {

    public static void main(String[] args) {

        Product[] products = {

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

            new Product(1, "Shirt", "Clothing"),

            new Product(2, "Book", "Education")

        };

        int index1 = LinearSearch.linearSearch(products, 2);

        System.out.println("Linear Search: Found at index " + index1);

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

        int index2 = BinarySearch.binarySearch(products, 2);

        System.out.println("Binary Search: Found at index " + index2);

    }

}

public class Product {

    int productId;

    String productName;

    String category;

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

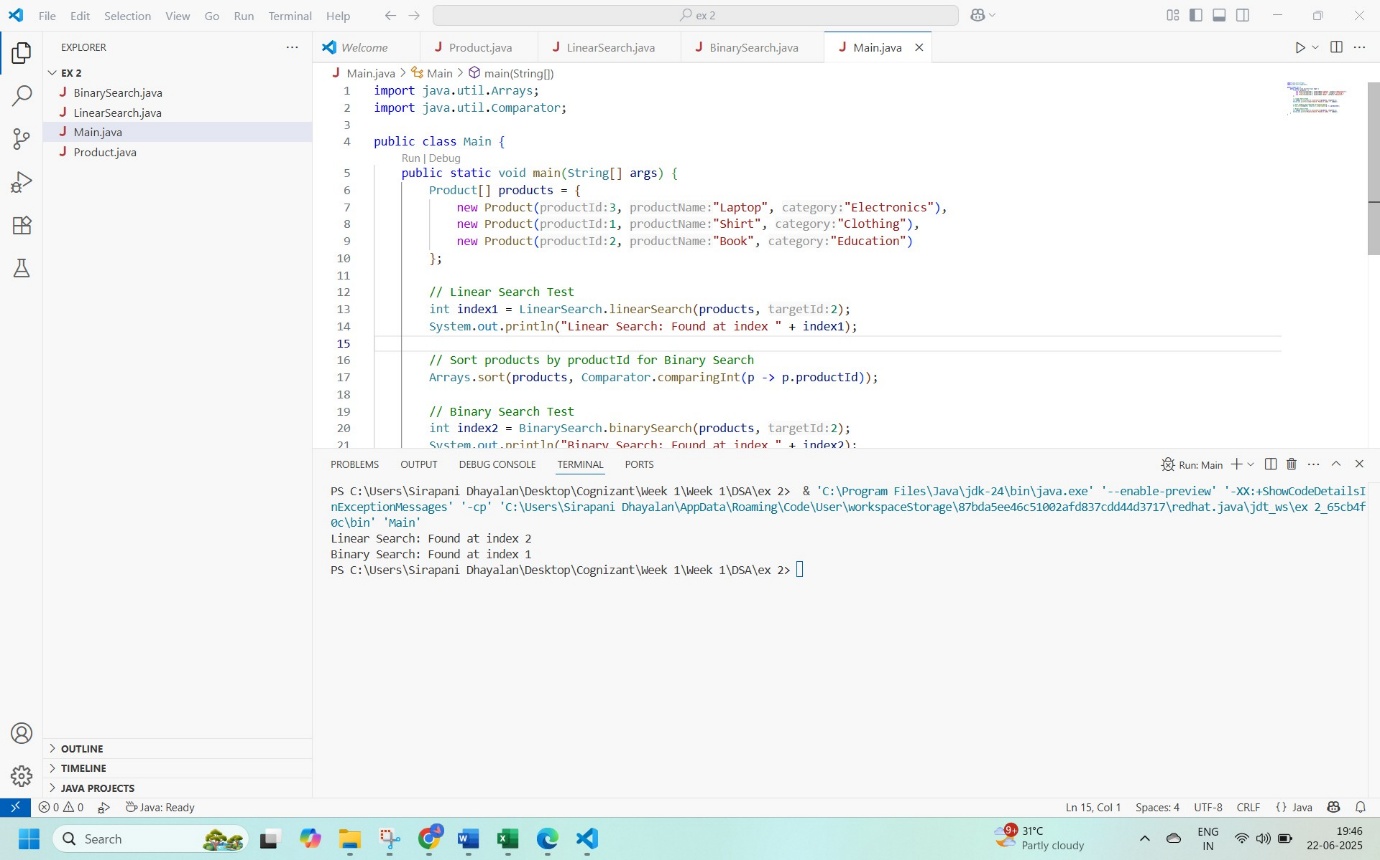
        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

}



**Exercise 7: Financial Forecasting**

public class FinancialForecast {

    // Recursive Method

    public static double futureValueRecursive(double presentValue, double growthRate, int periods) {

        if (periods == 0) {

            return presentValue; // Base Case

        } else {

            return futureValueRecursive(presentValue, growthRate, periods - 1) \* (1 + growthRate);

        }

    }

    // Iterative (Optimized) Method

    public static double futureValueIterative(double presentValue, double growthRate, int periods) {

        double result = presentValue;

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

            result \*= (1 + growthRate);

        }

        return result;

    }

}

public class Main {

    public static void main(String[] args) {

        double presentValue = 1000.0; // Initial amount

        double growthRate = 0.05;     // 5% growth per period

        int periods = 3;              // Forecast for 3 periods

        // Recursive calculation

        double futureValueRec = FinancialForecast.futureValueRecursive(presentValue, growthRate, periods);

        System.out.println("Future Value (Recursive) after " + periods + " periods: " + futureValueRec);

        // Iterative calculation

        double futureValueIter = FinancialForecast.futureValueIterative(presentValue, growthRate, periods);

        System.out.println("Future Value (Iterative) after " + periods + " periods: " + futureValueIter);

    }

}

A screenshot of a computer

AI-generated content may be incorrect.