**Algorithms and Data structures**

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

**Class 1 : Product.java**

public class Product {

    private int productId;

    private String productName;

    private String category;

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

        this.productId = productId;

        this.productName = productName;

        this.category = category;

    }

    public int getProductId() {

        return productId;

    }

    public String getProductName() {

        return productName;

    }

    public String getCategory() {

        return category;

    }

    @Override

    public String toString() {

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

    }

}

**Class 2 : LinearSearch.java**

public class LinearSearch {

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

        for (Product product : products) {

            if (product.getProductId() == targetId) {

                return product;

            }

        }

        return null;

    }

}

**Class 3 : BinarySearch.java**

import java.util.Arrays;

import java.util.Comparator;

public class BinarySearch {

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

        int low = 0;

        int high = products.length - 1;

        while (low <= high) {

            int mid = low + (high - low) / 2;

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

                return products[mid];

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

                low = mid + 1;

            } else {

                high = mid - 1;

            }

        }

        return null;

    }

    public static void sortProductsById(Product[] products) {

        Arrays.sort(products, Comparator.comparingInt(Product::getProductId));

    }

}

**Class 4 : Main.java**

public class Main {

public static void main(String[] args) {

Product[] products = {

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

new Product(203, "Smartphone", "Electronics"),

new Product(305, "Headphones", "Accessories"),

new Product(410, "Keyboard", "Accessories"),

new Product(520, "Mouse", "Accessories")

};

// Linear Search Test

int targetId = 305;

Product resultLinear = LinearSearch.linearSearch(products, targetId);

System.out.println("Linear Search Result: " + resultLinear);

// Binary Search Test (requires sorting)

BinarySearch.sortProductsById(products);

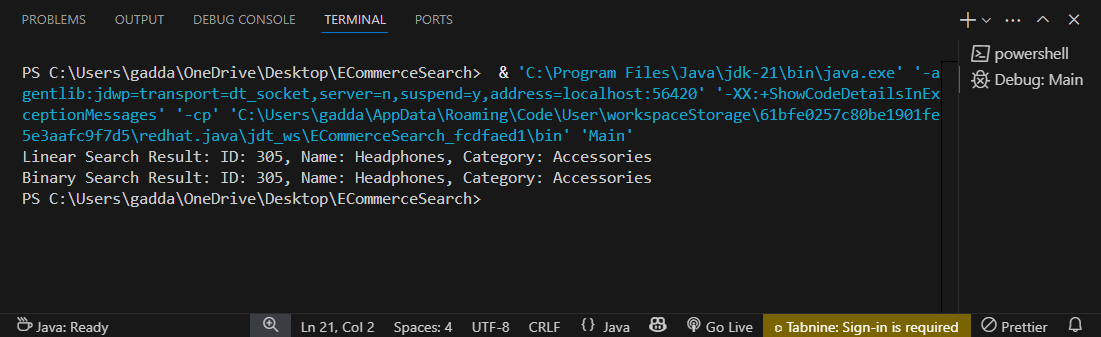
Product resultBinary = BinarySearch.binarySearch(products, targetId);

System.out.println("Binary Search Result: " + resultBinary);

}

}

**Output :**

****

**Exercise 7: Financial Forecasting**

**Class 1 : FinancialForecast.java**

public class FinancialForecast {

    // Recursive method to compute future value

    public static double futureValueRecursive(double currentValue, double growthRate, int yearsLeft) {

        if (yearsLeft == 0) {

            return currentValue;  // base case

        } else {

            return (1 + growthRate) \* futureValueRecursive(currentValue, growthRate, yearsLeft - 1);

        }

    }

    // Iterative method to compute future value (optimized)

    public static double futureValueIterative(double currentValue, double growthRate, int years) {

        double result = currentValue;

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

            result \*= (1 + growthRate);

        }

        return result;

    }

    public static void main(String[] args) {

        double currentValue = 1000.0;   // initial amount

        double growthRate = 0.05;       // 5% growth rate

        int years = 10;                 // number of years to forecast

        // Call recursive method

        double predictedValueRecursive = futureValueRecursive(currentValue, growthRate, years);

        System.out.println("Predicted value (recursive) after " + years + " years: " + predictedValueRecursive);

        // Call iterative method

        double predictedValueIterative = futureValueIterative(currentValue, growthRate, years);

        System.out.println("Predicted value (iterative) after " + years + " years: " + predictedValueIterative);

    }

}

**Output :**

