**EXERCISE 2: E-COMMERCE PLATFORM SEARCH FUNCTION**

**SETUP CODE:**

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 "Product{" +

"ID=" + productId +

", Name='" + productName + '\'' +

", Category='" + category + '\'' +

'}';

}

}

**IMPLEMENTATION CODE:**

import java.util.Arrays;

import java.util.Comparator;

public class ECommerceSearch {

// Linear Search - O(n)

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;

}

// Binary Search - O(log n)

public static int 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 mid;

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

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1;

}

// Main method

public static void main(String[] args) {

Product[] products = {

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

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

new Product(110, "Mobile", "Electronics"),

new Product(101, "Shirt", "Clothing"),

new Product(108, "Watch", "Accessories")

};

int targetId = 110;

// Linear Search

int linearResult = linearSearch(products, targetId);

if (linearResult != -1)

System.out.println("Linear Search Found: " + products[linearResult]);

else

System.out.println("Product not found using Linear Search.");

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

int binaryResult = binarySearch(products, targetId);

if (binaryResult != -1)

System.out.println("Binary Search Found: " + products[binaryResult]);

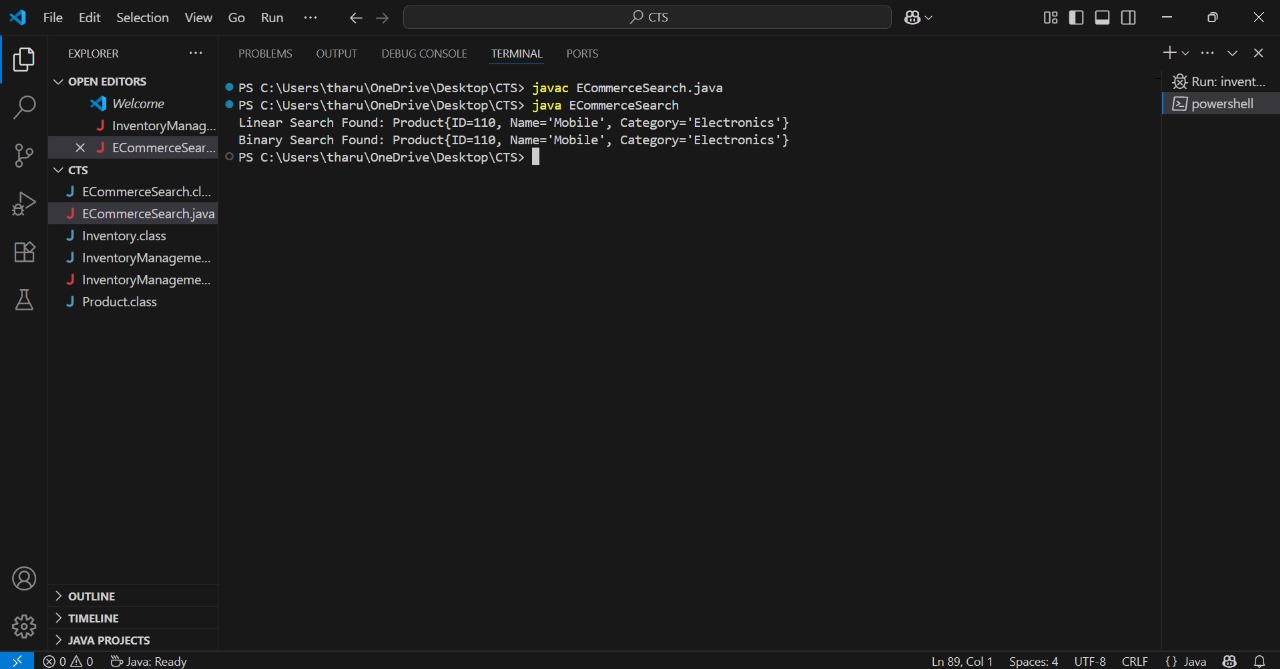
else

System.out.println("Product not found using Binary Search.");

}

}

**OUTPUT :**



**Analysis**

Linear Search:Use when the list is unsorted or small.

Binary Search:Best for sorted, large lists with frequent search operations.

**EXERCISE 7: FINANCIAL FORECASTING**

**CODE:**

public class FinancialForecast {

// Recursive method to calculate future value

public static double forecast(double amount, double rate, int years) {

if (years == 0) {

return amount;

}

return forecast(amount, rate, years - 1) \* (1 + rate);

}

public static double forecastMemo(double amount, double rate, int years, double[] memo) {

if (years == 0) {

return amount;

}

if (memo[years] != 0) {

return memo[years];

}

memo[years] = forecastMemo(amount, rate, years - 1, memo) \* (1 + rate);

return memo[years];

}

public static void main(String[] args) {

double initialAmount = 1000.0;

double growthRate = 0.05;

int years = 10;

double result = forecast(initialAmount, growthRate, years);

System.out.printf("Future value after %d years (recursive): ₹%.2f%n", years, result);

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

double memoResult = forecastMemo(initialAmount, growthRate, years, memo);

System.out.printf("Future value after %d years (memoized): ₹%.2f%n", years, memoResult);

}

}

**OUTPUT :**

