Exercise 2: E-commerce Platform Search Function

Code:

import java.util.Arrays;

import java.util.Comparator;

public class EComSearch {

static class Product {

int productId;

String productName;

String category;

public Product(int id, String name, String category) {

this.productId = id;

this.productName = name;

this.category = category;

}

public String toString() {

return "[" + productId + ", " + productName + ", " + category + "]";

}

}

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

int k=0;

for (Product product : products) {

k++;

if (product.productName.equalsIgnoreCase(targetName)) {

System.out.println("No of checks made:"+k);

return product;

}

}

System.out.println("No of checks made:"+k);

return null;

}

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

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

int k=0;

while (left <= right) {

k++;

int mid = (left + right) / 2;

int comparison = products[mid].productName.compareToIgnoreCase(targetName);

if (comparison == 0){

System.out.println("No of checks made:"+k);

return products[mid];

}

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

else right = mid - 1;

}

System.out.println("No of checks made:"+k);

return null;

}

public static void main(String[] args) {

Product[] products = {

new Product(1, "Shoes", "Footwear"),

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

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

new Product(4, "Chargers", "Electronics"),

new Product(5, "Book", "Stationery")

};

System.out.println("Linear Search for 'Chargers':");

Product res = linearSearch(products, "Chargers");

if(res!=null) System.out.println("Details of searched product:"+res);

else System.out.println("Sorry we are unable to find such product.");

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

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

Product result2 = binarySearch(products, "Chargers");

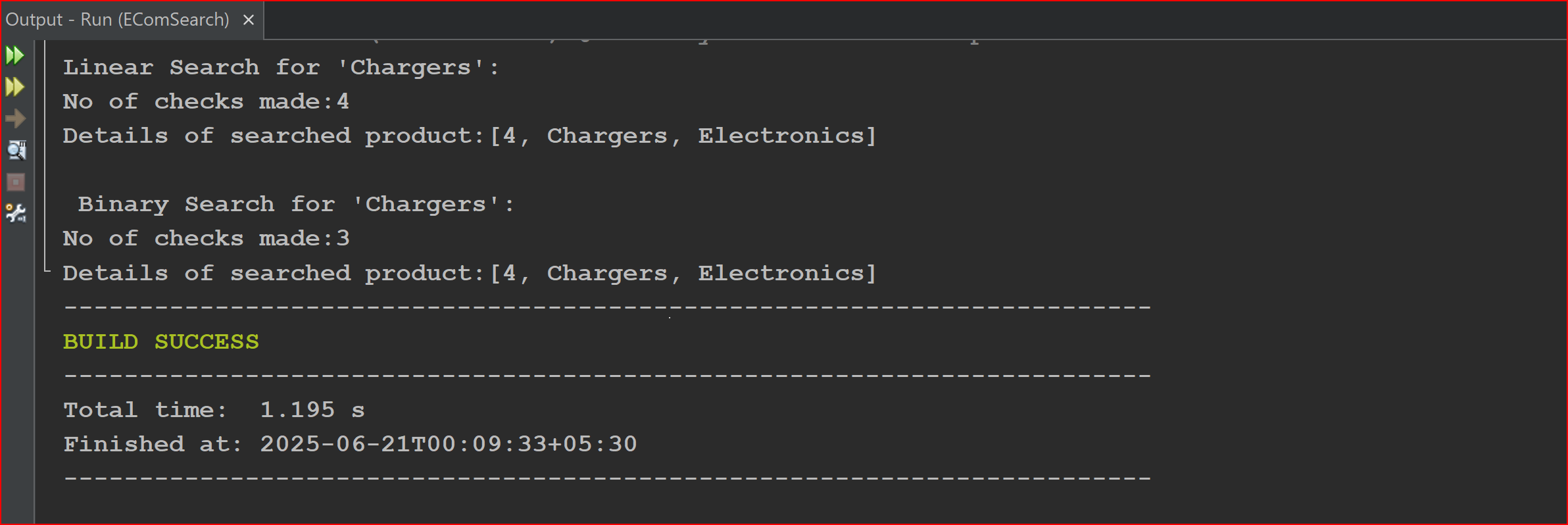
if(res!=null) System.out.println("Details of searched product:"+res);

else System.out.println("Sorry we are unable to find such product.");

}

}

Output:



**Analysis of searching algorithms used:**

It is found that searching for an element in an array using linear search algorithm would take O(n) time in the worst and average case scenarios and would take O(1) time in the best case where the first element is the one to be searched for. Using Binary Search we can bring down the worst and average case time complexity to O(log n) but it demands for a sorted list of elements.

Exercise 7: Financial Forecasting

Code:

public class FinancialForecasting {

public static double compute(double principle,double roi,int years){

if(years==0) return principle;

return (1+roi)\*compute(principle,roi,years-1);

}

public static void main(String[] args) {

double principle=10000;

double roi=0.07;

int years=7;

double amount=compute(principle,roi,years);

System.out.printf("The amount you receive after completion of the term is %2f", amount);

}

}

**Analysis:** we can use recursion when we want to compute results of a similar sub problem. In the cases where the same problems may repeat at a later time we can use memoization technique and store the answer to each sub problem and if the sub problem is repeated just fetch the stored value to avoid recomputation.

Output:

