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

Code :

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

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;

}

@Override

public String toString() {

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

}

}

class SearchUtils {

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

for (Product product : products) {

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

return product;

}

}

return null;

}

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

for (Product product : products) {

if (product.productId == targetId) {

return product;

}

}

return null;

}

public static Product binarySearchByName(Product[] sortedProducts, String targetName) {

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

while (low <= high) {

int mid = (low + high) / 2;

String midName = sortedProducts[mid].productName.toLowerCase();

int cmp = targetName.toLowerCase().compareTo(midName);

if (cmp == 0)

return sortedProducts[mid];

else if (cmp < 0)

high = mid - 1;

else

low = mid + 1;

}

return null;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Product[] products = {

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

new Product(102, "Tablets", "Health"),

new Product(103, "Headphones", "Electronics"),

new Product(104, "Chair", "Furniture")

};

Product[] sortedProducts = Arrays.copyOf(products, products.length);

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

System.out.println("E-Commerce Search Platform");

System.out.println("1. Linear Search by Name");

System.out.println("2. Linear Search by ID");

System.out.println("3. Binary Search by Name");

System.out.print("Enter choice: ");

int choice = scanner.nextInt();

scanner.nextLine();

Product result = null;

switch (choice) {

case 1:

System.out.print("Enter product name: ");

String name1 = scanner.nextLine();

result = SearchUtils.linearSearchByName(products, name1);

break;

case 2:

System.out.print("Enter product ID: ");

int id = scanner.nextInt();

result = SearchUtils.linearSearchById(products, id);

break;

case 3:

System.out.print("Enter product name: ");

String name2 = scanner.nextLine();

result = SearchUtils.binarySearchByName(sortedProducts, name2);

break;

default:

System.out.println("Invalid choice.");

return;

}

if (result != null)

System.out.println("product Found: " + result);

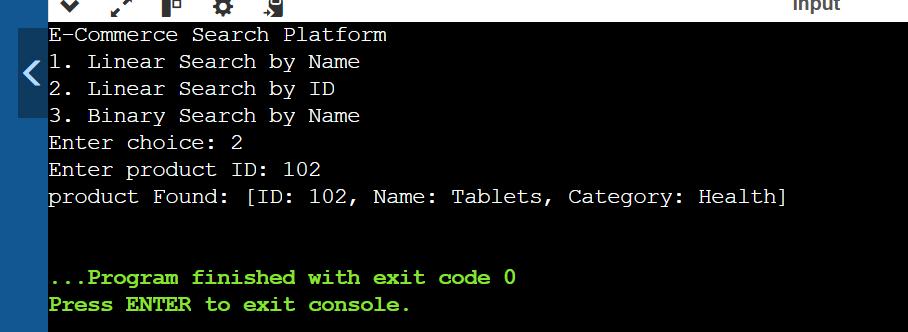
else

System.out.println("product Not Found.");

}

}

Output :



**Exercise 7: Financial Forecasting**

Code :

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter present amount: ");

double amount = sc.nextDouble();

System.out.print("Enter annual growth rate (%): ");

double rate = sc.nextDouble() / 100.0;

System.out.print("Enter number of years: ");

int years = sc.nextInt();

double futureValueRecursive = FinancialForecast.calculateFutureValue(amount, rate, years);

double futureValueIterative = FinancialForecast.calculateFutureValueIterative(amount, rate, years);

System.out.printf("Recursive Future Value: %.2f%n", futureValueRecursive);

System.out.printf("Iterative Future Value: %.2f%n", futureValueIterative);

}

}

class FinancialForecast {

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

if (years == 0) return amount;

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

}

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

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

amount \*= (1 + rate);

}

return amount;

}

}

Output :

