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

import java.util.Arrays;

import java.util.Comparator;

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 productId + ": " + productName + " (" + category + ")";

}

}

class SearchEngine {

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

for (Product product : products) {

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

return product;

}

}

return null;

}

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

int low = 0;

int high = products.length - 1;

while (low <= high) {

int mid = (low + high) / 2;

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

if (cmp == 0) {

return products[mid];

} else if (cmp < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return null;

}

}

public class Main {

public static void main(String[] args) {

Product[] products = {

new Product(1, "Laptop", "Electronics"),

new Product(2, "Shoes", "Fashion"),

new Product(3, "Watch", "Accessories"),

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

new Product(5, "Tablet", "Electronics")

};

String searchItem1 = "Watch";

Product result1 = SearchEngine.linearSearch(products, searchItem1);

System.out.println("Linear Search Result: " + (result1 != null ? result1 : "Product not found"));

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

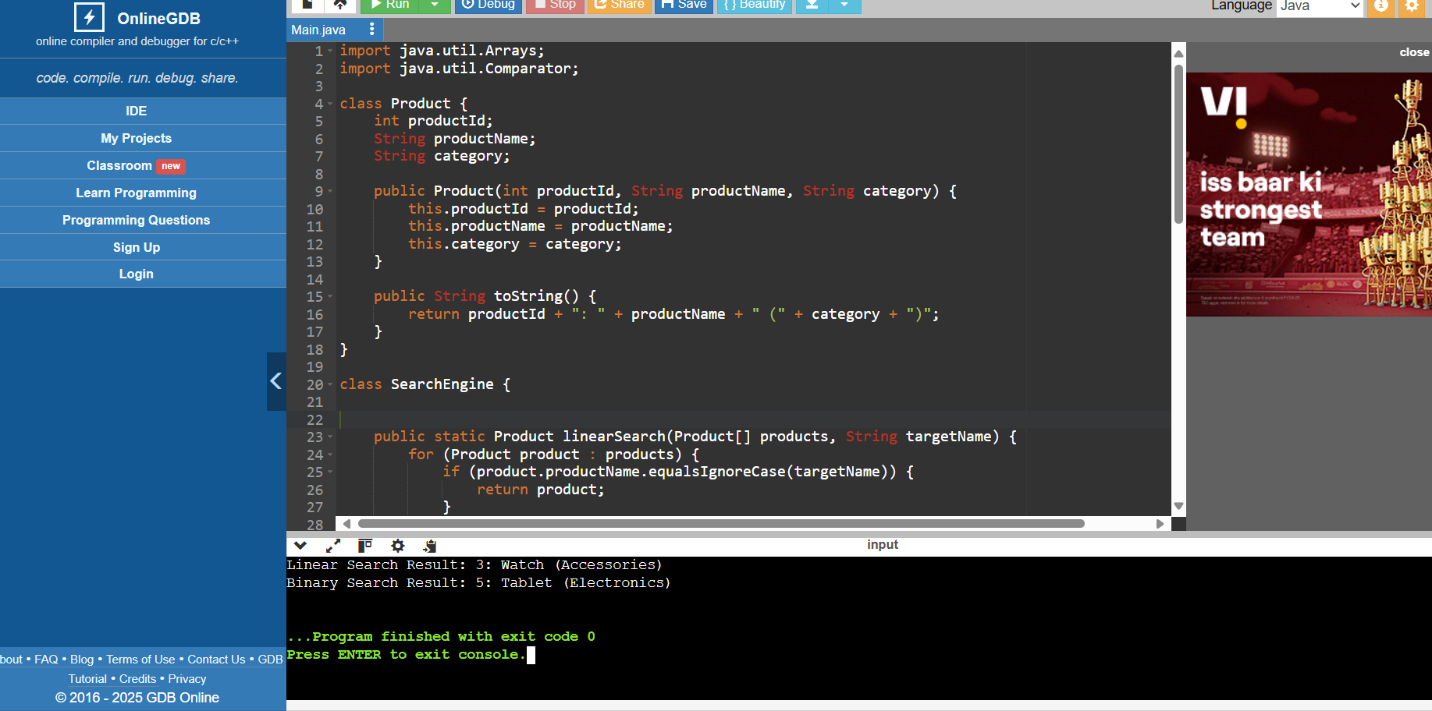
String searchItem2 = "Tablet";

Product result2 = SearchEngine.binarySearch(products, searchItem2);

System.out.println("Binary Search Result: " + (result2 != null ? result2 : "Product not found"));

}

}



Exercise 7: Financial Forecasting

import java.util.\*;

public class FinancialForecast {

public static double futureValueRecursive(double principal, double rate, int years) {

if (years == 0) {

return principal;

}

return futureValueRecursive(principal, rate, years - 1) \* (1 + rate);

}

public static double futureValueIterative(double principal, double rate, int years) {

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

principal \*= (1 + rate);

}

return principal;

}

public static void main(String[] args) {

double principal = 100000;

double growthRate = 0.10;

int years = 5;

double recursiveResult = futureValueRecursive(principal, growthRate, years);

double iterativeResult = futureValueIterative(principal, growthRate, years);

System.out.printf("Recursive Result after %d years: ₹%.2f\n", years, recursiveResult);

System.out.printf("Iterative Result after %d years: ₹%.2f\n", years, iterativeResult);

}

}