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

**Code:**

import java.util.\*;

import java.util.stream.Collectors;

public class ECommerceSearchApp {

static class Product {

String id;

String name;

String description;

String category;

double price;

public Product(String id, String name, String description, String category, double price) {

this.id = id;

this.name = name;

this.description = description;

this.category = category;

this.price = price;

}

public String getId() { return id; }

public String getName() { return name; }

public String getDescription() { return description; }

public String getCategory() { return category; }

public double getPrice() { return price; }

@Override

public String toString() {

return "Product{" + "name='" + name + '\'' + ", category='" + category + '\'' + ", price=" + String.format("%.2f", price) + '}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Product product = (Product) o;

return Objects.equals(id, product.id);

}

@Override

public int hashCode() {

return Objects.hash(id);

}

}

static class ProductCatalog {

private final Map<String, Product> allProductsById = new HashMap<>();

private final Map<String, List<Product>> productsByCategory = new HashMap<>();

private final Map<String, Set<Product>> invertedIndex = new HashMap<>();

public void addProduct(Product product) {

if (product == null || allProductsById.containsKey(product.getId())) return;

allProductsById.put(product.getId(), product);

productsByCategory.computeIfAbsent(product.getCategory().toLowerCase(), k -> new ArrayList<>()).add(product);

indexTextForKeywords(product.getName(), product);

indexTextForKeywords(product.getDescription(), product);

}

private void indexTextForKeywords(String text, Product product) {

if (text == null || text.trim().isEmpty()) return;

String[] words = text.toLowerCase().split("\\W+");

for (String word : words) {

if (!word.isEmpty()) {

invertedIndex.computeIfAbsent(word, k -> new HashSet<>()).add(product);

}

}

}

public Collection<Product> getAllProducts() {

return allProductsById.values();

}

}

public static List<Product> search(ProductCatalog catalog, String keywordQuery, String category, Double minPrice, Double maxPrice) {

Set<Product> currentResults = new HashSet<>(catalog.getAllProducts());

if (keywordQuery != null && !keywordQuery.trim().isEmpty()) {

Set<Product> keywordMatches = new HashSet<>();

String[] queryWords = keywordQuery.toLowerCase().split("\\W+");

for (String word : queryWords) {

if (catalog.invertedIndex.containsKey(word)) {

keywordMatches.addAll(catalog.invertedIndex.get(word));

}

}

currentResults.retainAll(keywordMatches);

}

if (category != null && !category.trim().isEmpty()) {

List<Product> categoryProducts = catalog.productsByCategory.getOrDefault(category.toLowerCase(), Collections.emptyList());

currentResults.retainAll(categoryProducts);

}

if (minPrice != null || maxPrice != null) {

final double actualMin = (minPrice != null) ? minPrice : Double.MIN\_VALUE;

final double actualMax = (maxPrice != null) ? maxPrice : Double.MAX\_VALUE;

currentResults = currentResults.stream()

.filter(p -> p.getPrice() >= actualMin && p.getPrice() <= actualMax)

.collect(Collectors.toSet());

}

return new ArrayList<>(currentResults);

}

public static void main(String[] args) {

ProductCatalog catalog = new ProductCatalog();

System.out.println("--- Adding Products ---");

catalog.addProduct(new Product("P001", "Laptop Pro X", "Powerful laptop for professionals", "Electronics", 1200.00));

catalog.addProduct(new Product("P002", "Mechanical Keyboard", "RGB gaming keyboard", "Electronics", 95.50));

catalog.addProduct(new Product("P003", "Wireless Mouse", "Ergonomic mouse for daily use", "Electronics", 25.00));

catalog.addProduct(new Product("P004", "Desk Chair Ergonomic", "Comfortable office chair", "Furniture", 180.75));

catalog.addProduct(new Product("P005", "4K Monitor 27 inch", "High resolution display", "Electronics", 350.00));

catalog.addProduct(new Product("P006", "Gaming Headset", "Surround sound headset", "Electronics", 70.00));

catalog.addProduct(new Product("P007", "Wooden Bookshelf", "Solid wood bookshelf", "Furniture", 120.00));

System.out.println("Products added.\n");

System.out.println("--- Search Scenarios ---");

System.out.println("\nSearch for 'laptop':");

printResults(search(catalog, "laptop", null, null, null));

System.out.println("\nSearch for 'Electronics' category:");

printResults(search(catalog, null, "Electronics", null, null));

System.out.println("\nSearch for products between $50 and $200:");

printResults(search(catalog, null, null, 50.0, 200.0));

System.out.println("\nSearch for 'keyboard' in 'Electronics' category:");

printResults(search(catalog, "keyboard", "Electronics", null, null));

System.out.println("\nSearch for 'Furniture' between $100 and $150:");

printResults(search(catalog, null, "Furniture", 100.0, 150.0));

System.out.println("\nSearch for 'mouse' in 'Electronics' between $10 and $30:");

printResults(search(catalog, "mouse", "Electronics", 10.0, 30.0));

System.out.println("\nSearch for 'nonexistent':");

printResults(search(catalog, "nonexistent", null, null, null));

System.out.println("\nSearch with no criteria (all products):");

printResults(search(catalog, null, null, null, null));

}

private static void printResults(List<Product> results) {

if (results.isEmpty()) {

System.out.println(" No products found.");

} else {

results.forEach(product -> System.out.println(" " + product));

}

System.out.println(" (" + results.size() + " results)");

}

}

**Output:**

****