public enum WordOrder {

ALPHABETIC\_ASCENDING,

ALPHABETIC\_DESCENDING,

LENGTH\_THEN\_ALPHABETIC\_ASCENDING,

INPUT\_ORDER

}

import java.util.\*;

import java.util.stream.Collectors;

public class UniqueWordExtractor {

public Set<String> extractUniqueWords(String paragraph, WordOrder order) {

// Split the paragraph into words and filter out duplicates based on INPUT\_ORDER

Set<String> uniqueWords = new LinkedHashSet<>(Arrays.asList(paragraph.split("\\s+")));

// Return the sorted set based on the specified order

return sortWords(uniqueWords, order);

}

private Set<String> sortWords(Set<String> words, WordOrder order) {

List<String> sortedWords = new ArrayList<>(words);

switch (order) {

case ALPHABETIC\_ASCENDING:

sortedWords.sort(Comparator.naturalOrder());

break;

case ALPHABETIC\_DESCENDING:

sortedWords.sort(Comparator.reverseOrder());

break;

case LENGTH\_THEN\_ALPHABETIC\_ASCENDING:

sortedWords.sort(Comparator.comparingInt(String::length).thenComparing(Comparator.naturalOrder()));

break;

case INPUT\_ORDER:

// Input order is preserved by LinkedHashSet, no sorting needed

return new LinkedHashSet<>(sortedWords);

}

return new LinkedHashSet<>(sortedWords);

}

}

public class Main {

public static void main(String[] args) {

UniqueWordExtractor extractor = new UniqueWordExtractor();

String paragraph = "apple orange banana apple grape orange";

System.out.println(extractor.extractUniqueWords(paragraph, WordOrder.ALPHABETIC\_ASCENDING));

System.out.println(extractor.extractUniqueWords(paragraph, WordOrder.ALPHABETIC\_DESCENDING));

System.out.println(extractor.extractUniqueWords(paragraph, WordOrder.LENGTH\_THEN\_ALPHABETIC\_ASCENDING));

System.out.println(extractor.extractUniqueWords(paragraph, WordOrder.INPUT\_ORDER));

}

}