# Lab: Associative Arrays, Lambda and Stream API

Problems for exercises and homework for the ["Programming Fundamentals" course @ SoftUni](https://softuni.bg/courses/programming-fundamentals)

You can check your solutions in [Judge.](https://judge.softuni.bg/Contests/1311)

## Associative Arrays

### Count Real Numbers

Read a **list of real numbers** and **print them in ascending order** along with their **number of occurrences**.

#### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 8 2 2 8 2 | 2 -> 3  8 -> 2 | 1 5 1 3 | 1 -> 2  3 -> 1  5 -> 1 | -2 0 0 2 | -2 -> 1  0 -> 2  2 -> 1 |

#### Solution

Read an array of real numbers (**double**).



Use TreeMap<Double, Integer> named counts.



Pass through each input number num and increase counts (when num exists in the map) or add it with value 1.



Pass through all numbers num in the map and print the number and its count of occurrences after formatting it to a decimal place **without trailing zeros** (otherwise the output will have too much decimal places, e.g. 2.500000 instead of 2.5);



package newpackage;  
  
import java.util.Map;  
import java.util.Scanner;  
import java.util.TreeMap;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 String[] line = scanner.nextLine().split(" ");  
 Double[] realNumbers = new Double[line.length];  
  
 for (int i = 0; i < line.length; i++) {  
 realNumbers[i] = Double.*parseDouble*(line[i]);  
 }  
  
 Map<Double, Integer> countRealNumbers = new TreeMap<>();  
  
 for (Double key : realNumbers) {  
 Integer getCurrentValue = countRealNumbers.get(key);  
 if (!countRealNumbers.containsKey(key)) {  
 countRealNumbers.put(key, 1);  
 } else {  
 countRealNumbers.put(key, getCurrentValue + 1);  
 }  
 }  
  
 for (Map.Entry<Double, Integer> doubleIntegerEntry : countRealNumbers.entrySet()) {  
 System.*out*.printf("%.0f -> %d%n", doubleIntegerEntry.getKey(), doubleIntegerEntry.getValue());  
 }  
  
 }  
}

### 2. Word Synonyms

Write a program which keeps a map with synonyms. The **key** of the map will be the **word**. The **value** will be a **list of all the synonyms of that word**. You will be given a number **n**. On the next **2 \* n** lines you will be given a **word** and a **synonym** each on a separate line like this:

* {**word**}
* {**synonym**}

If you get the same word for second time, just add the new synonym to the list.

Print the words in the following format:

**{word} - {synonym1, synonym2… synonymN}**

#### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  cute  adorable  cute  charming  smart  clever | cute - adorable, charming  smart - clever |
| 2  task  problem  task  assignment | task – problem, assignment |

#### Hints

* Use **LinkedHashMap (String -> ArrayList<String>)** to keep track of all words

****

* **Read 2 \* n lines**
* **Add the word in the Map if it is not present**
* **Add the synonym as value to the given word**



* **Print each word with the synonyms in the required format specified above**

package newpackage;  
import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 Map<String, List<String>> keysAndSynonyms = new TreeMap<>();  
 int pairsCount = Integer.*parseInt*(scanner.nextLine());  
  
 for (int i = 0; i < pairsCount; i++) {  
 String key = scanner.nextLine();  
 String synonym = scanner.nextLine();  
  
 if (keysAndSynonyms.containsKey(key)) {  
 List<String> stringList = keysAndSynonyms.get(key);  
 stringList.add(synonym);  
 keysAndSynonyms.put(key,stringList);  
 } else {  
 List <String> synonymsList = new ArrayList<>();  
 synonymsList.add(synonym);  
 keysAndSynonyms.put(key,synonymsList);  
  
 }  
 }  
 for (Map.Entry<String, List<String>> entry : keysAndSynonyms.entrySet()) {  
 System.*out*.printf("%s - %s%n",entry.getKey(),entry.getValue().toString().replaceAll("[\\]\\[]",""));  
 }  
 }

### 3. Odd Occurrences

Write a program that extracts from a given sequence of words all elements that are present in it an **odd number of times** (**case-insensitive**).

* Words are given in a single line, **space** separated
* Print the result elements in lowercase in their order of appearance

#### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Java C# PHP PHP JAVA C java | java, c#, c |
| 3 5 5 hi pi HO Hi 5 ho 3 hi pi | 5, hi |
| a a A SQL xx a xx a A a XX c | a, sql, xx, c |

#### Hints

Read a line from the console and split it by a space:



Use a **LinkedHashMap** (String 🡪 int) to count the occurrences of each word:



Pass through all elements in the array and count each word:



Create a new **ArrayList** (**String**), which will hold all the words with **odd occurences**:



Now all that is left is to **print** the words, **separated by comma and single space** (**", "**).



package newpackage;  
  
import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 String[] words = scanner.nextLine().split(" ");  
 Map<String, Integer> occurrences = new LinkedHashMap<>();  
  
 for (String word : words) {  
 String wordInLowerCase = word.toLowerCase();  
 if (occurrences.containsKey(wordInLowerCase)) {  
 Integer count = occurrences.get(wordInLowerCase);  
 occurrences.put(wordInLowerCase, count + 1);  
 } else {  
 occurrences.put(wordInLowerCase, 1);  
 }  
 }  
  
  
 List<String> oddWords = new ArrayList<>();  
 for (var entry : occurrences.entrySet()) {  
 if (entry.getValue() % 2 == 1) {  
 oddWords.add(entry.getKey());  
 }  
 }  
 for (int i = 0; i < oddWords.size(); i++) {  
 System.*out*.print(oddWords.get(i));  
 if (i < oddWords.size() - 1) {  
 System.*out*.print(", ");  
 }  
 }  
 }  
}

## II. Stream API

### 4. Word Filter

Read an array of **strings**, take only words which length is **even**. Print each word on a new line.

#### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| kiwi orange banana apple | kiwi  orange  banana |
| pizza cake pasta chips | cake |

* Read an array of strings
* **Filter** those whose length is even



* Print each word on a new line

package newpackage;  
import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 String [] words = scanner.nextLine().split(" ");  
 Arrays.*stream*(words)  
 .filter(x -> x.length() % 2 == 0)  
 .forEach(x -> System.*out*.println(x));  
 }  
}

### 5. Largest 3 Numbers

Read a **list of integers** and **print largest 3 of them**. If there are **less** than 3, print **all** of them.

#### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 10 30 15 20 50 5 | 50 30 20 | 20 30 | 30 20 |

#### Hints

* Read a list of integers
* Order the list using **Stream API**

****

* Print top 3 numbers with **for** loop

package newpackage;  
import java.util.\*;  
import java.util.stream.Collectors;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
  
 List<Integer> nums = Arrays  
 .*stream*(scanner.nextLine().split(" "))  
 .map(e -> Integer.*parseInt*(e))  
 .sorted((n1, n2) -> n2.compareTo(n1))  
 .limit(3)  
 .collect(Collectors.*toList*());  
 for (int num : nums) {  
 System.*out*.print(num + " ");  
 }  
 }  
}