**Lab: Associative Arrays, Lambda and Stream API**

Problems for exercises and homework for the ["Programming Fundamentals" course @ HYPERLINK "https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023"SoftUni](https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023)

You can check your solutions in [Ju HYPERLINK "https://judge.softuni.org/Contests/1311"d HYPERLINK "https://judge.softuni.org/Contests/1311"ge HYPERLINK "https://judge.softuni.org/Contests/1311".](https://judge.softuni.org/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 many decimal places, e.g., 2.500000 instead of 2.5);



* **Word Synonyms**

Write a program that keeps a map with synonyms. The **key** to 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 the **word** and a **synonym** each on a separate line like this:

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

If you get the same word for the 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 the **word.**



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



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



* **Stream API**
* **Word Filter**

Read an array of **strings**, and take only words whose 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.