Lab: Associative Arrays, Lambda and Stream API

Problems for exercises and homework for the "Programming Fundamentals" course @ SoftUni

You can check your solutions in Judge.

I. Associative Arrays

1. Count Real Numbers

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

Examples

Input	Output
82282	2 -> 3 8 -> 2

Input	Output
1513	1 -> 2 3 -> 1 5 -> 1

Input	Output
-2002	-2 -> 1 0 -> 2 2 -> 1

Solution

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

```
double[] nums = Arrays
        .stream(sc.nextLine()
                .split( regex: " "))
        .mapToDouble(Double::parseDouble)
        .toArray();
```

Use **TreeMap<Double, Integer>** named **counts**.

```
TreeMap<Double, Integer> counts = new TreeMap<>();
```

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

```
for (double num : nums) {
    if (!counts.containsKey(num)) {
        counts.put(num, 0);
    }
    counts.put(num, counts.get(num) + 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);

```
for (Map.Entry<Double, Integer> entry : counts.entrySet()) {
    DecimalFormat df = new DecimalFormat( pattern: "#.#######");
    System.out.printf("%s -> %d%n", df.format(entry.getKey()), entry.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

```
LinkedHashMap<String, ArrayList<String>> words = new LinkedHashMap<>();
```

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

```
for (int i = 0; i < n; i++) {
    String word = sc.nextLine();
    String synonym = sc.nextLine();

    words.putIfAbsent(word, new ArrayList<>());
    words.get(word).add(synonym);
}
```

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

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:

```
Scanner sc = new Scanner(System.in);

String[] words = sc.nextLine().split( regex: " ");
```

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

```
LinkedHashMap<String, Integer> counts = new LinkedHashMap<>();
```

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

```
for (String word : words) {
    String wordInLowerCase = word.toLowerCase();
    if (counts.containsKey(wordInLowerCase)) {
        counts.put(wordInLowerCase, counts.get(wordInLowerCase) + 1);
    } else {
        counts.put(wordInLowerCase, 1);
    }
}
```

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

```
ArrayList<String> odds = new ArrayList<>();

for (var entry : counts.entrySet()) {
    if (entry.getValue() % 2 == 1) {
        odds.add(entry.getKey());
    }
}
```

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

```
for (int i = 0; i < odds.size(); i++) {
    System.out.print(odds.get(i));
    if (i < odds.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

```
String[] words = Arrays
        .stream(sc.nextLine()
                .split( regex: " "))
        .filter(w -> w.length() % 2 == 0)
        .toArray(String[]::new);
```

- Print each word on a new line

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
10 30 15 20 50 5	50 30 20

Input	Output
20 30	30 20

Hints

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

```
List<Integer> sorted = Arrays
        .stream(sc.nextLine()
                .split( regex: " "))
        .map(Integer::parseInt).sorted((n1, n2) -> n2.compareTo(n1))
        .collect(Collectors.toList());
```

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