Intro to Java

Exercises: Arrays

Write a method that calculates the sum of the items

```
public static int sumScores(ArrayList<Integer> scoreBoard) {
   //... your code goes here!
}
```

Example:

```
Input: [1,4,7,3,6]
Output: 21
```

Write a method that finds the highest integer in the list

```
public static int getHighScore(ArrayList<Integer> scoreBoard) {
   //....
}
```

Example:

```
Input: [1,4,7,3,6]
Output: 7
```

Write a method that finds the average value in a list

```
public static int average(ArrayList<Integer> scoreBoard) {
   //....
}
```

Example:

```
Input: [1,4,7,3,6]
Output: 4.2
```

Write a method that returns a new array that contains the elements of the input array in the reverse order

```
public static ArrayList<Integer> reverseScores(ArrayList<Integer> scoreBoard) {
   //...
}
```

Example:

```
Input: [1,4,7,3,6]
Output: [6,3,7,4,1]
```

Remove words too short

- Let the user choose how many words they want to input
- Read that many words
- Let the user choose a number bigger than 0
- Remove all words from the ArrayList that are shorter than the given number

Example:

```
Input:
4 // four words
Ananas
Zimtzitroneneis
Apple
Zartbitterschokolade
6 // remove all shorter than six

Output:
```

```
Zimtzitroneneis
Zartbitterschokolade
```

To implement the exercise, introduce a method with the following signature and implement it and use it inside `main`:

```
public static ArrayList<String> removeShortWords(ArrayList<String> words, int minLength) {
        // Return a new ArrayList that the words of 'words' that have a length equal or more than `minLeng
}
```

IMPORTANT: make sure to return a new ArrayList instead of modifying the one passed as parameter.

Truncate

- Let the user choose how many words they want to input
- Read that many words into an `ArrayList`
- Let the user choose another word
- Print the `ArrayList` truncated at the word the user provided

To implement the exercise, introduce a method with the following signature and implement it and use it inside `main`:

```
public static ArrayList<String> truncateAfter(ArrayList<String> values, String x) {
        // Return a new ArrayList that contains what 'values'
        // contains up to a certain String, passed as 2nd parameter
        // Values after 'x', including 'x', should be discarded and not returned
}
```

IMPORTANT: make sure to return a new ArrayList instead of modifying the one passed as parameter.

Finding elements

Implement a method that finds the position of a `String` inside an `ArrayList<String>`. It shall return the position of the element (where 0 means the 1st element, 1 the 2nd, and so on). In case the element searched is not present in the `ArrayList`, the method shall return -1.

```
public static int findValue(ArrayList<String> values, String x) {
    // Return the position of `x` inside `values`,
    // return -1 if `x` is not found
}
```

NOTE: this method actually exists and it's called `indexOf()`. A simpler method that checks if an object is present in an `ArrayList` without returning its position, but just returning a `boolean` also already exists and it's called `contains()`.

Remove duplicates

Implement the following method and use it inside a `main` to show that it behaves as described:

```
public static ArrayList<String> removeDuplicates(ArrayList<String> values) {
        // Return a new ArrayList that contains what 'values'
        // contains, with duplicate elements removed
}
```

HINT: reuse `findValue` implemented previously or use `indexOf()` or, better, `contains()`.

Common values (intersection)

Implement the following method and use it inside a `main` to show that behaves as described:

```
public static ArrayList<String> commonValues(
                ArrayList<String> values1,
                ArrayList<String> values2) {

        // Return a new ArrayList containing the elements that
        // are present both in 'values1' and 'values2', without duplicates
}
```

HINT: reuse methods defined so far.

Union

Let's implement the `union` of two `ArrayList`s. The exercise is also about reusing existing methods. Reuse as much as you can of what we've written so far, and what is also available from Java ArrayList. The shorter the solution the better.

```
public static ArrayList<String> union(
                ArrayList<String> values1,
                ArrayList<String> values2) {


        // Return the union of 'values1' and 'values2'
        // that is basically every element contained in
        // any of the two, without duplicates
}
```

Difference

Take two lists as input and return a new list with all values of the first list that are not in the second list

Example:

```
Input:
[1, 2, 3, 4] and [1, 4, 5, 9]
Output:
[2, 3]
```

Unique Values

Take two lists as input and return a new list with all values that are only in one of both lists

Example:

```
Input:
[1, 2, 3, 4] and [1, 4, 5, 9]
Output:
[2, 3, 5, 9]
```

Word count

A common exercise is counting how many times a word occur in a set. For every unique word in the set, you should print the number of occurrences:

```
Input:
6 // number of words
hello
world
hello
germany
hello
berlin

Output:
hello, 3
world, 1
germany, 1
berlin, 1
```

DIFFICULT: implementing this exercise in just one method is difficult. Try to split the problem in smaller ones and reuse existing methods instead.

HINT: define and implement first a `count` method and use it in the `wordCount` exercise above:

```
public static int count(ArrayList<String> values, String x) {
    // Return how many times the string 'x' is present in 'values'
}
```

Swap

Write a program that allows a user to enter numbers. Save the numbers in an ArrayList.

If the user enters a -1, stop and print the list.

Ask the user now for two indices (positions) in the ArrayList. Swap the elements at these positions and print the resulting list.

Example:

```
Enter some numbers:
8
3
6
5
4
-1

Your list:
[8,3,6,5,4]

Enter two indices:
1
3

Your new list:
[8,5,6,3,4]
```

Zip

Create a program that has two arraylists of Strings (you can decide if you define these in your code directly or if the user should enter the strings).

Now create a new arraylist from these two in the following way:

- take as first value for the new list the first value from the first list
- take as next value for the new list the first value from the second list
- take as next value for the new list the second value from the first list …

until all values from both lists are in the new list. (start with two lists of the same size => if you have a solution, extend it by handling lists of different sizes => as soon as one source list has no value anymore, just add all remaining values from the other list)

Example: Given two lists of same size, list 1 being [“a”, “b”, “c”] list 2 being [“x”, “y”, “z”]

resulting list: [“a”,”x”,”b”,”y”,”c”,”z”]

Example: Given two lists of different size list 1 being [“a”, “b”, “c”, “x”] list 2 being [“y”, “z”] [“a”,”y”,”b”,”z”,”c”,”x”]

Guess character

Write a program that creates an ArrayList of characters (choose as many and which you like).

Now let the user enter characters to guess which characters are in the arraylist.

Let him continue guessing until he has entered all characters in the list (bonus point: print at the end how many guesses he took).

Example: Let the arraylist in my code contains the characters a,d,n

```
Guess a character
a

Correct

Guess a character
z

Wrong

Guess a character
d

Correct

Guess a character
n

Correct

You guessed all characters in 4 tries
```

Sorting (difficulty: higher)

Implement a sorting function!

Your method should take a list of numbers and return the same numbers sorted (from smallest to greatest).

```java
import java.util.ArrayList;


class Main {
  public static void main(String[] args) {


  }


  public static ArrayList<Integer> sort(ArrayList<Integer> values) {
    // implement me
  }
}
```

Example:

```
Input:
[4, 7, 3, 2, 6, 1, 5]
Output:
[1, 2, 3, 4, 5, 6, 7]
```

You can implement your own idea of a sorting algorithm.

Or try to implement an existing sort algorithm:

these ones are some simple sort algorithms
https://en.wikipedia.org/wiki/Bubble_sort#Pseudocode_implementation

https://en.wikipedia.org/wiki/Gnome_sort

https://en.wikipedia.org/wiki/Insertion_sort