Stream devoxx 2014

Le but du jeu est d'écrire une fonction permettant de concaténer un certain nombre de listes, passées en paramètre sous forme de *var-arg* :

```
public <T> List<T> concatLists(List<T>... lists);
```

Version 1 : foreach

Algorithme

Boucler sur chaque liste et les rajouter dans une liste.

Solution

```
public <T> List<T> concatLists1(List<T>... lists) {
    ArrayList<T> result = new ArrayList<>();
    for (List<T> list : lists) {
        result.addAll(list);
    }
    return result;
}
```

Version 2 : Reduce

Algorithme

L'opérateur Reduce est

```
T reduce(T identity,
         BinaryOperator<T> accumulator)
```

Effectue une réduction sur les éléments de ce flux, à l'aide de la valeur d'identité fournie et d'une fonction d'accumulation associative, et renvoie la valeur réduite. Cela équivaut à :

```
    T result = identity;
      for (T element :  this stream)
          result = accumulator.apply(result, element)
      return result;
```

L'idée est de concater les listes avec un stream

Solution

```
public <T> List<T> concatLists2(List<T>... lists) {
    return Stream.of(lists).reduce(new ArrayList<>(), (list1, list2) -> {
        list1.addAll(list2);
        return list1;
    });
}
```

Version 3 via flatMap

Algorithme

```
<R> Stream<R> flatMap(Function<? super T,? extends Stream<? extends R>> mapper)
```

La méthode **`flatMap()`** permet d'appliquer une fonction à chaque élément du flux puis d'aplatir le résultat en un nouveau flux.

Solution

```
public static <T> List<T> concatLists(List<T>... lists) {
       return Stream.of(lists).flatMap(List::stream).collect(Collectors.toList());
   }
```

Revision #1
Created 19 June 2022 09:15:20 by ggpilou2

Updated 19 June 2022 10:52:23 by ggpilou2