1. The problem consists on the creation of a generic class to handle the document buffer of a printer. Documents are printed using a numeric priority system (lower number => higher priority), being FIFO when priorities are equal. The printer has a limited buffer for documents and can only receive documents as long as its total memory is not overflowed. The printer works with objects, which implement the ***Document*** interface. Consider that a class named ***BufferDocument*** exists and implements the ***Document interface***.

```
public interface Document extends Comparable <Document> {
     Integer getPriority();
     Integer getSize();
     String getName();
     String getAuthor(); }
```

a) Create the generic class **PriorityBufferPrinter** (and its attributes), which parameter is or extends Document and implements the native generic interface **Iterable**.

b) Implement one constructor of **PriorityBufferPrinter** which receives the object's maximum allowed memory.

c) Create the method **addDocument** of **PriorityBufferPrinter** which adds a Document to the buffer in the order presented above. Use ordered insertion.

d) Create the method **getDocument** of **PriorityBufferPrinter** which returns and deletes from the buffer the next Document in the order presented above.

e) Create the method **delDocument** of **PriorityBufferPrinter** which deletes a document from the buffer (if it exists), given a name and an author.

f) Create the method **delDocumentsAbove** of **PriorityBufferPrinter** which deletes all the documents which size is superior to a given size.

g) Using **JUnit**, create test(s) for the methods in c), d), e) and f).

2. Given a list of non-repeated integers and k centers integers of this list, we intend to create k-sublists (k >= 2) such that in each k-sublist are the integers closest to the center, i.e. integers of initial list with least absolute difference with respect to the center k. If an integer has the same distance to several centers it is placed in only one sublist (any one). As an example, the following list L = {2, 9, 7, 5, 10, 15, 6, 12, 3} and centers C1 = 3, C2 = 6 and C3 = 10. The original list L will be divided into three Sublists: L1 = {3, 2}, L2 = {6, 7, 5} and L3 = {10, 9, 15, 12} that must be returned in a container that stores the <Center, sublist> pairs.

```
Map<Integer, LinkedList<Integer>> KsubLists(LinkedList<Integer> list,
                                                    ArrayList<Integer> centers)
```

Complementary Exercises

1. Write a method that takes a **synonym map**, where the key is a word and the value is a list of synonyms for that word. The goal is to return a list of synonyms that appear **more than n times**. The return list must not have repeated synonyms. Consider the following signature:

```
List<String> moreSynonyms(Map<String, List<String>> mapSyn, Integer i)
```

2. Write a generic method for maintaining a **list of favourites**. The elements in this list must be sorted in descending order of its search. For example, given the integer list l={4, 5, 8, 9}, and the elements search in the following order p={8,9,9,3,9,5,8,8,3}, produces the list of favourites f={9, 8, 5, 4}. Assume that list l does not contain repeats and two or more elements with the same search frequency appear in arbitrary order (the favourite list of the indicated search can also be l={8, 9, 5, 4}).

```
public static<E> void favouriteList(List<E> l, List<E> p)
```