# Special collections

## Description

Implement special collections by given specifications

## [Pair String List](/gogibystrov/special-collections/-/blob/master/src%5Cmain%5Cjava%5Ccom%5Cefimchick%5Cifmo%5Ccollections%5CPairStringList.java)

Implement a list of Strings that adding/removing elements by pairs.

Each time you add a string to the list, you should add it twice, so the list would contain a pair of entries of that string.

Each time you remove a string, you should remove a pair of it.

List should keep pairs together: if you add a string by index ensure it will not break a pair. Put pair of new element after it.

What should be covered in your implementation:

* adding
* adding by index
* removing of object
* removing by index
* getting by index
* setting by index
* adding of a collection
* adding of a collection by index
* iterator (removing via iterator is not required)

## [Sorted By Absolute Value Integer Set](/gogibystrov/special-collections/-/blob/master/src%5Cmain%5Cjava%5Ccom%5Cefimchick%5Cifmo%5Ccollections%5CSortedByAbsoluteValueIntegerSet.java)

Implement a class of Integer sorted set. Values in set should be sorted by their absolute values in ascending order.

What should be covered in your implementation:

* adding element
* removing element
* searching element
* adding of a collection
* iterator (removing via iterator is not required)

## [Median Queue](/gogibystrov/special-collections/-/blob/master/src%5Cmain%5Cjava%5Ccom%5Cefimchick%5Cifmo%5Ccollections%5CMedianQueue.java)

Implement a queue of Integer that returns its median element.  
Median here is an element which has equal amounts of lesser and greater elements. Median represents middle value of the collection.  
For instance, if you put 1, 2, 3, 4, 5 to the queue and then pull element of it, queue will return 3.  
If there are even amount of elements, there are two possible values to return. Return lesser one.  
More examples:

* 1, 10, 100 → 10
* 100, 10, 1 → 10
* 100, 1, 10 → 10
* 1, 987, 2 → 2
* 1, 987, 2, 3 → 2
* 1, 987, 4, 2, 3 → 3
* 1, 2, 3, 3, 3 → 3

You need to implement following methods:

* offer - push element to the queue
* poll - pull element out of the queue
* peek - get element on the top of the queue (just get, no pulling out)
* iterator - iterate over elements of the queue (no need to keep order)
* size - just amount of current queue elements