Java Collections Framework Reference Sheet

List				
	`get(i)`	`set(i,x)`	`add(i,x)`	`remove(i)`
`ArrayList`	$O(1)$	$O(1)$	$O(1+n-i)$	$O(n-i)$
`LinkedList`	$O(1+\min\{i,n-i\})$			

Set			
	`add(x)`	`remove(x)`	`contains(x)`
`HashSet`	$O(1)$		
`TreeSet`	$O(\log n)$		

SortedSet			
	`headSet(y)`[1]	`tailSet(x)`[1]	`subSet(x,y)`[1]
`TreeSet`	$O(\log n)$		

Map[2]			
	`get(k)`	`put(k,v)`	`containsKey(k)`
`HashMap`	$O(1)$		
`TreeMap`[3]	$O(\log n)$		

Deque				
	`addFirst(x)`	`removeFirst()`	`addLast(x)`	`removeLast(x)`
`ArrayDeque`	$O(1)$			
`LinkedList`	$O(1)$			

Queue			
	`add(x)`	`remove()`	`element()`
`ArrayDeque`	$O(1)$		
`LinkedList`	$O(1)$		
`PriorityQueue`	$O(\log n)$	$O(\log n)$	$O(1)$

Collections			
`sort(list)`	`min(c)/max(c)`	`reverse(list)`	`binarySearch(list,x)`[4]
$O(n \log n)$	$O(n)$	$O(n)$	$O(\log n)$

[1] Avoid using the `size()` method on the sets returned by `headSet(y)`, `tailSet(x)`, or `subSet(x,y)`; it takes $\Omega(n)$ time. Use `isEmpty()` if you only want to check if the set is empty.

[2] Use `keySet()`, `values()`, and `entrySet()` to get direct access to the `Set` of keys, `Collection` of values, or `Set` of key/value pairs in the `Map`.

[3] `TreeMap` implements the `SortedMap` interface, so its `keySet()` is a `SortedSet`.

[4] To run in $O(\log n)$ time, `binarySearch(list,x)` requires that `list` have an $O(1)$-time `get(i)` operation; only use it with an `ArrayList`.