CopyOnWriteArrayList

The name "CopyOnWrite" is important.

Whenever this list is modified, by adding, updating, or removing elements, a new copy of the underlying array is created.

The modification is performed on the new copy, allowing concurrent read operations to use the original unmodified array.

This ensures that reader threads aren't blocked by writers.

Since changes are made to a separate copy of the array, there aren't any synchronization issues between the reading and writing threads.

This is ordinarily too costly, but may be *more* efficient than alternatives when traversal operations, vastly outnumber mutations.

Removing Single Element From the ArrayBlockingQueue

The ArrayBlockingQueue has several different methods to get an element from the queue.

Most of these will get the element at the head of the queue, or the first in.

If the queue is empty, a method will return null or block as described on this slide.

	Blocks?	Returns	Throws InterruptedException?	Removes element from queue
`peek()`	No	Array item or null	No	No
`poll()`	No	Array item or null	No	Yes
`poll(long timeout, TimeUnit unit)`	Temporarily	Array item or null	Yes	Yes
`remove()`	Yes, when Queue is empty	Array item	No	Yes
`remove(Object o)`	No	boolean	No	Yes, if o was in the queue
`take()`	Yes, when Queue is empty	Array item	Yes	Yes