ArrayList>

The size, isEmpty, get, set, iterator, and listIterator operations run in constant time **O(1)**. The add operation runs in amortized constant time, that is, adding n elements requires **O(n)** time. All of the other operations run in linear time. The constant factor is low compared to that for the LinkedList implementation.

Q=what is the reason of creating notifall by java people when we can call wait from synchronised variable

Q=difference between countdown latch and Cyclic barrier?

|  |  |
| --- | --- |
| 100down voteaccepted | One major difference is that [CyclicBarrier](http://download.oracle.com/javase/1.5.0/docs/api/java/util/concurrent/CyclicBarrier.html) takes an (optional) Runnable task which is run once the common barrier condition is met.  It also allows you to get the number of clients waiting at the barrier and the number required to trigger the barrier. Once triggered the barrier is reset and can be used again.  For simple use cases - services starting etc... a CountdownLatch is fine. A CyclicBarrier is useful for more complex co-ordination tasks. An example of such a thing would be parallel computation - where multiple subtasks are involved in the computation - kind of like [MapReduce](http://en.wikipedia.org/wiki/MapReduce). |

Q=Handling exception in Future Object?

Ans=