Difference between ArrayList<E> and Vector<E>

[ArrayList<E>](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-5list-interface.html) and Vector<E> both of them comes under [Java Collection Framework](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-1.html) and implements [List<E>](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-5list-interface.html) interface. Learn more about [Collection interface here](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-4collection.html) and various [interfaces under Java Collections Framework here](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-2-interfaces.html).

Previously I wrote on [difference between ArrayList<E> and LinkedList<E>](http://data-structure-learning.blogspot.com/2015/05/difference-between-arraylist-and_28.html), [similarities between ArrayList<E> and LinkedList<E>](http://data-structure-learning.blogspot.com/2015/05/similarities-between-arraylist-and.html) and [when to use either of them](http://data-structure-learning.blogspot.com/2015/05/when-to-use-arraylist-and-linkedlist.html).

You can also see [Iterating ArrayList<E> 6 different ways](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-6iterating-over.html), [Iterating HashMap<K, V> 5 different ways](http://data-structure-learning.blogspot.com/2015/05/different-ways-to-traverse-map.html).

Now let us start with the difference between ArrayList<E> and Vector<E>. As we know both of the class uses the backing array.

1. Resizing operation on backing array.
   1. In ArrayList<E> resizing is roughly 1.5 times of old array. Elements of old array are copied to new array using Arrays.copyOf(…)
   2. In Vector<E> resizing if twice of old array. Elements of old array are copied to new array using Arrays.copyOf(..)
2. [Fail-Fast property](http://data-structure-learning.blogspot.com/2015/05/fail-fast-iterator.html).
   1. [Iterator<E>](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-8-iterator.html) and [ListIterator<E>](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-9-listiterator.html) returned by ArrayList<E> are fail-fast. Means than if the list is structurally modified then [ConcurrentModificationException](http://data-structure-learning.blogspot.com/2015/05/concurrentmodificationexception.html) is thrown.
   2. Vector returns Enumeration. Enumeration is not fail-fast.
3. Synchronization.
   1. ArrayList<E> is not synchronized. That means if multiple threads tries to modify the list then the backing array might result in inconsistent data. Alternative to that is to use CopyOnWriteArrayList<E> which is thread safe.
   2. Vector<E> is synchronized so multiple thread can access it safely.
4. Performance.
   1. As ArrayList<E> is not synchronized it has no overhead of managing threading or synchronization issues so it is faster.
   2. Vector<E> is synchronized as its performance is poor. Methods are synchronized and hence multiple threads needs to wait as there might be another thread working with lock acquired.