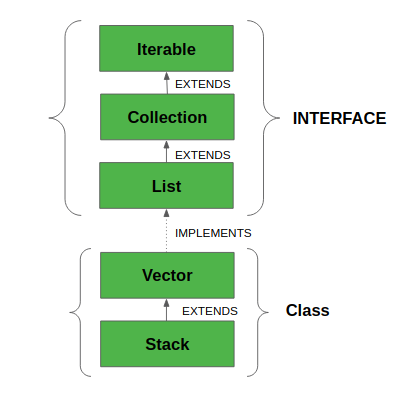
The Vector class implements a growable array of objects. Vectors fall in legacy classes, but now it is fully compatible with collections. It is found in[java.util package](https://www.geeksforgeeks.org/java-util-package-java/) and implement the [List](https://www.geeksforgeeks.org/list-interface-java-examples/) interface, so we can use all the methods of the List interface as shown below as follows:



* Vector implements a dynamic array which means it can grow or shrink as required. Like an array, it contains components that can be accessed using an integer index.
* They are very similar to [ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/), but Vector is synchronized and has some legacy methods that the collection framework does not contain.
* It also maintains an insertion order like an ArrayList. Still, it is rarely used in a non-thread environment as it is **synchronized**, and due to this, it gives a poor performance in adding, searching, deleting, and updating its elements.
* The Iterators returned by the Vector class are fail-fast. In the case of concurrent modification, it fails and throws the **ConcurrentModificationException.**

**Syntax:**

public class Vector<E> extends AbstractList<E> implements List<E>, RandomAccess, Cloneable, Serializable

* It extends [AbstractList](https://www.geeksforgeeks.org/abstractlist-in-java-with-examples/) and implements [List](https://www.geeksforgeeks.org/list-interface-java-examples/) interfaces.
* It implements Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess interfaces.
* The directly known subclass is [Stack](https://www.geeksforgeeks.org/stack-class-in-java/).

**Important points regarding the Increment of vector capacity are as follows:**

If the increment is specified, Vector will expand according to it in each allocation cycle. Still, if the increment is not specified, then the vector’s capacity gets doubled in each allocation cycle. Vector defines three protected data members:

* ***int capacityIncreament:*** Contains the increment value.
* ***int elementCount:*** Number of elements currently in vector stored in it.
* ***Object elementData[]:*** Array that holds the vector is stored in it.

Common Errors in the declaration of Vectors are asfollows**:**

* Vector throws an **IllegalArgumentException** if the InitialSize of the vector defined is negative.
* If the specified collection is null, It throws **NullPointerException**.

### Advantages of using Vector in Java:

1. Synchronization: As mentioned before, Vector is synchronized, making it safe to use in a multi-threaded environment.
2. Dynamic Size: The size of a Vector can grow or shrink dynamically as elements are added or removed, so you don’t have to worry about setting an initial size that will accommodate all elements.
3. Legacy support: Vector has been part of Java since its inception and is still supported, so it’s a good option if you need to work with older Java code that uses Vector.

### Disadvantages of using Vector in Java:

1. Performance: The synchronization in Vector can lead to slower performance compared to other collection classes, such as ArrayList.
2. Legacy Code: While Vector is still supported, newer Java code is often written using the more modern collection classes, so it may be harder to find examples and support for Vector.
3. Unnecessary overhead: If you don’t need the synchronization features of Vector, using it will add unnecessary overhead to your code.

**Vector** is like the dynamic array which can grow or shrink its size. Unlike array, we can store n-number of elements in it as there is no size limit. It is a part of Java Collection framework since Java 1.2. It is found in the java.util package and implements the List interface, so we can use all the methods of List interface here.

It is recommended to use the Vector class in the thread-safe implementation only. If you don't need to use the thread-safe implementation, you should use the ArrayList, the ArrayList will perform better in such case.

The Iterators returned by the Vector class are fail-fast. In case of concurrent modification, it fails and throws the ConcurrentModificationException.

It is similar to the ArrayList, but with two differences-

* Vector is synchronized.
* Java Vector contains many legacy methods that are not the part of a collections framework.

# Java Vector

The Vector class is an implementation of the List interface that allows us to create resizable-arrays similar to the [ArrayList](https://www.programiz.com/java-programming/arraylist) class.

## Java Vector vs. ArrayList

In Java, both ArrayList and Vector implements the List interface and provides the same functionalities. However, there exist some differences between them.

The Vector class synchronizes each individual operation. This means whenever we want to perform some operation on vectors, the Vector class automatically applies a lock to that operation.

It is because when one thread is accessing a vector, and at the same time another thread tries to access it, an exception called ConcurrentModificationException is generated. Hence, this continuous use of lock for each operation makes vectors less efficient.

However, in array lists, methods are not synchronized. Instead, it uses the Collections.synchronizedList() method that synchronizes the list as a whole.

**Note:** It is recommended to use ArrayList in place of Vector because vectors less efficient.