Vector vs ArrayList in Java

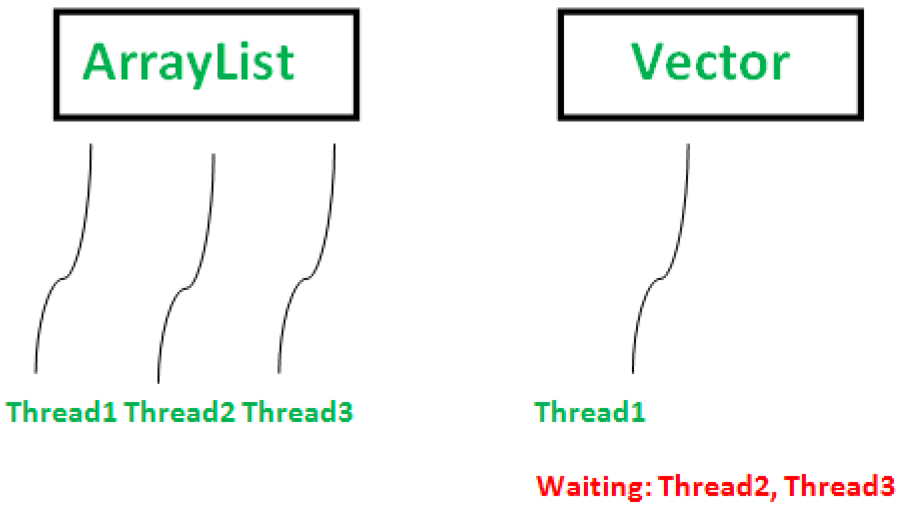
ArrayList and Vectors both implements List interface and both use **array(dynamically resizeable)** as data structure internally very much like using an ordinary array .

Syntax:

```
ArrayList<T> al = new ArrayList<T>();
Vector<T> v = new Vector<T>();
```

Major Differences between ArrayList and Vector:

1. **Synchronization :** Vector is **synchronized** that means at a time only one thread can access the code while arrayList is **not synchronized** that means multiple threads can work on arrayList at same time. For example, if one thread is performing add operation, then there can be another thread performing remove operation in multithreading environment.
   If multiple threads access arrayList concurrently then we must synchronize the block of the code which modifies the list either structurally or simple modifies element. Structural modification means addition or deletion of element(s) from the list. Setting the value of an existing element is not a structural modification.



2. **Performance: ArrayList is faster** as it is non-synchronized while vector operations give slow performance as they are synchronized(thread-safe). If one thread works on vector has acquired lock on it which makes other thread will has to wait till lock is released.
3. **Data Growth:** ArrayList and Vector **both grow and shrink dynamically** to maintain optimal use of storage. But the way they resized is different. ArrayList increments 50% of current array size if number of elements exceeds its capacity while vector increments 100% means doubles the current array size.
4. **Traversal:** Vector can use both Enumeration and Iterator for traversing over elements of vector while ArrayList can only use **Iterator** for traversing.

Note: ArrayList is preferable when there is no specific requirement to use vector.

```
// Java Program to illustrate use of ArrayList
// and Vector in Java
import java.io.*;
import java.util.*;

class GFG
{
    public static void main (String[] args)
    {
        // creating an ArrayList
        ArrayList<String> al = new ArrayList<String>();

        // adding object to arraylist
        al.add("Practice.GeeksforGeeks.org");
        al.add("quiz.GeeksforGeeks.org");
        al.add("code.GeeksforGeeks.org");
        al.add("contribute.GeeksforGeeks.org");

        // traversing elements using Iterator'
        System.out.println("ArrayList elements are:");
        Iterator it = al.iterator();
        while (it.hasNext())
            System.out.println(it.next());

        // creating Vector
        Vector<String> v = new Vector<String>();
        v.addElement("Practice");
        v.addElement("quiz");
        v.addElement("code");

        // traversing elements using Enumeration
        System.out.println("\nVector elements are:");
        Enumeration e = v.elements();
        while (e.hasMoreElements())
            System.out.println(e.nextElement());
    }
}
```

Run on IDE

Output:

```
ArrayList elements are:
Practice.GeeksforGeeks.org
quiz.GeeksforGeeks.org
code.GeeksforGeeks.org
contribute.GeeksforGeeks.org

Vector elements are:
Practice
quiz
code
```

How to choose between ArrayList and Vector?

- ArrayList is unsynchronized and not thread safe whereas Vecrors are. Only one thread can call methods on a Vector at a time which is a slight overhead but helpful when safety is a concern. Therefore, in a single threaded case arrayList is an obvious choice but in multithreading vectors can be preferred.
- If we don't know how much data we are going to have, but know the rate at which it grows, Vector has advantage since we can set the increment value in vectors.
- ArrayList is newer and faster. If we don't have any explicit requirements for using any of them – we use ArrayList over vector.

This article is contributed by **Nitsdheerendra**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner   Company Wise Coding Practice

Java Java-Collections