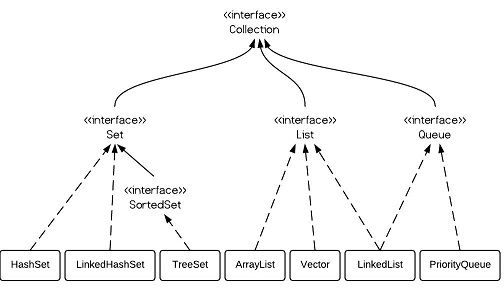
**[ArrayList vs LinkedList vs Vector](http://www.cnblogs.com/chenpi/p/5505375.html)**

**阅读目录**

* [List概览](http://www.cnblogs.com/chenpi/p/5505375.html#_label0)
* [ArrayList vs LinkedList vs Vector](http://www.cnblogs.com/chenpi/p/5505375.html#_label1)
* [ArrayList例子](http://www.cnblogs.com/chenpi/p/5505375.html#_label2)
* [LinkedList例子](http://www.cnblogs.com/chenpi/p/5505375.html#_label3)
* [Vector](http://www.cnblogs.com/chenpi/p/5505375.html#_label4)
* [ArrayList和LinkedList性能对比](http://www.cnblogs.com/chenpi/p/5505375.html#_label5)

**List概览**

List，正如它的名字，表明其是有顺序的。当讨论List的时候，最好拿它跟Set作比较，Set中的元素是无序且唯一；下面是一张类层次结构图，从这张图中，我们可以大致了解java集合类的整体架构；



**ArrayList vs LinkedList vs Vector**

从上面的类层次结构图中，我们可以发现他们都实现了List接口，它们使用起来非常相似。区别主要在于它们各自的实现，不同的实现导致了不同的性能和不同的操作。

ArrayList是为可变数组实现的，当更多的元素添加到ArrayList的时候，它的大小会动态增大。它的元素可以通过get/set方法直接访问，因为ArrayList本质上是一个数组。

LinkedList是为双向链表实现的，添加、删除元素的性能比ArrayList好，但是get/set元素的性能较差。

Vector与ArrayList相似，但是它是同步的。

如果你的程序是线程安全的，ArrayList是一个比较好的选择。当更多的元素被添加的时候，Vector和ArrayList需要更多的空间。Vector每次扩容会增加一倍的空间，而ArrayList增加50%。

注意：ArrayList默认的初始空间大小相当的小，通过构造函数去初始化一个更大的空间是一个好习惯，可以避免扩容开销。

**ArrayList例子**

[复制代码](javascript:void(0);)

ArrayList<Integer> al = new ArrayList<Integer>();

al.add(3);

al.add(2);

al.add(1);

al.add(4);

al.add(5);

al.add(6);

al.add(6);

Iterator<Integer> iter1 = al.iterator();

while(iter1.hasNext()){

System.out.println(iter1.next());

}

[复制代码](javascript:void(0);)

**LinkedList例子**

[复制代码](javascript:void(0);)

LinkedList<Integer> ll = new LinkedList<Integer>();

ll.add(3);

ll.add(2);

ll.add(1);

ll.add(4);

ll.add(5);

ll.add(6);

ll.add(6);

Iterator<Integer> iter2 = ll.iterator();

while(iter2.hasNext()){

System.out.println(iter2.next());

}

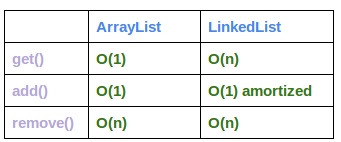
[复制代码](javascript:void(0);)

从以上代码，我们可以发现它们的使用非常相似，真正地区别在于它们的底层实现和操作复杂度。

**Vector**

Vector和ArrayList几乎是一样的，区别在于Vector是线程安全的，因为这个原因，它的性能较ArrayList差。通常情况下，大部分程序员都使用ArrayList，而不是Vector，因为他们可以自己做出明确的同步操作。

**ArrayList和LinkedList性能对比**



表中的add()方法指add(E e)， remove()方法指remove(int index)

* ArrayList对任意的add，remove操作，时间复杂度为O(n)，但是在列表末尾的操作，其时间复杂度为O(1)。
* LinkedList对任意的add，remove操作，时间复杂度为O(n)，但是在列表末尾的操作，其时间复杂度为O(1)。

我使用如下代码测试它们的性能：

[复制代码](javascript:void(0);)

package simplejava;

import java.util.ArrayList;

import java.util.LinkedList;

public class Q24 {

public static void main(String[] args) {

ArrayList<Integer> arrayList = new ArrayList<Integer>();

LinkedList<Integer> linkedList = new LinkedList<Integer>();

// ArrayList add

long startTime = System.nanoTime();

for (int i = 0; i < 100000; i++) {

arrayList.add(i);

}

long endTime = System.nanoTime();

long duration = endTime - startTime;

System.out.println("ArrayList add: " + duration);

// LinkedList add

startTime = System.nanoTime();

for (int i = 0; i < 100000; i++) {

linkedList.add(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("LinkedList add: " + duration);

// ArrayList get

startTime = System.nanoTime();

for (int i = 0; i < 10000; i++) {

arrayList.get(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("ArrayList get: " + duration);

// LinkedList get

startTime = System.nanoTime();

for (int i = 0; i < 10000; i++) {

linkedList.get(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("LinkedList get: " + duration);

// ArrayList remove

startTime = System.nanoTime();

for (int i = 9999; i >= 0; i--) {

arrayList.remove(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("ArrayList remove: " + duration);

// LinkedList remove

startTime = System.nanoTime();

for (int i = 9999; i >= 0; i--) {

linkedList.remove(i);

}

endTime = System.nanoTime();

duration = endTime - startTime;

System.out.println("LinkedList remove: " + duration);

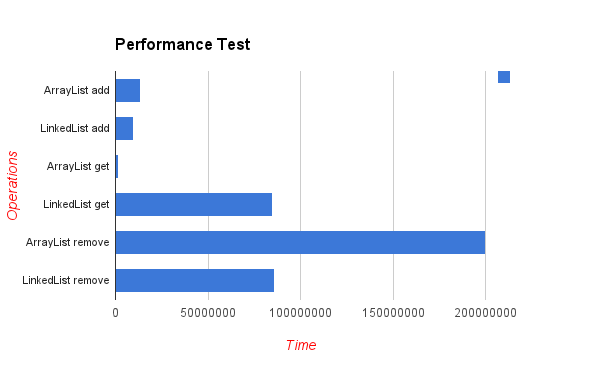
}

}

[复制代码](javascript:void(0);)

结果打印如下：

ArrayList add: 13265642  
LinkedList add: 9550057  
ArrayList get: 1543352  
LinkedList get: 85085551  
ArrayList remove: 199961301  
LinkedList remove: 85768810



它们性能的区别很明显。对于Add和remove操作，LinkedList性能较好，但是get操作性能较差。基于上面的时间复杂度表和测试结果，我们可以得出什么时候使用ArrayList还是LinkedList。简单的说，LinkedList适用于如下情况：

* 没有大量的随机访问操作
* 有大量的add/remove操作

译文链接：<http://www.programcreek.com/2013/03/arraylist-vs-linkedlist-vs-vector/>