

# ArrayList实现原理分析（Java源码剖析）



六尺帐篷 (/u/f8e9b1c246f1)

2017.08.05 21:18\* 字数 1579 阅读 711 评论 6 喜欢 35

(/u/f8e9b1c246f1)

[编辑文章 \(/writer#/notebooks/15099409/notes/15440753\)](#)

- ArrayList使用的存储的数据结构
- ArrayList的初始化
- ArrayList是如何动态增长
- ArrayList如何实现元素的移除
- ArrayList小结

ArrayList是我们经常使用的一个数据结构，我们通常把其用作一个可变长度的动态数组使用，大部分时候，可以替代数组的作用，我们不用事先设定ArrayList的长度，只需要往里不断添加元素即可，ArrayList会动态增加容量。ArrayList是作为List接口的一个实现。

那么ArrayList背后使用的数据结构是什么呢？

ArrayList是如何保证动态增加容量，使得能够正确添加元素的呢？

要回答上面的问题，我们就需要对ArrayList的源码进行一番分析，深入了解其实现原理的话，我们就自然能够解答上述问题。

需要说明的是，本文所分析的源码引用自JDK 8版本

## ArrayList使用的存储的数据结构

从源码中我们可以发现，ArrayList使用的存储的数据结构是Object的对象数组。其实这也不能想象，我们知道ArrayList是支持随机存取的类似于数组，所以自然不可能是链表结构。

```
/**
 * The array buffer into which the elements of the ArrayList are stored.
 * The capacity of the ArrayList is the length of this array buffer. Any
 * empty ArrayList with elementData == DEFAULTCAPACITY_EMPTY_ELEMENTDATA
 * will be expanded to DEFAULT_CAPACITY when the first element is added.
 */
transient Object[] elementData; // non-private to simplify nested class access
```

我想大家一定对这里出现的transient关键字很疑惑，我们都知道ArrayList对象是可序列化的，但这里为什么要用transient关键字修饰它呢？查看源码，我们发现ArrayList实现了自己的readObject和writeObject方法，所以这保证了ArrayList的可序列化。具体序列化的知识我们在此不过多赘述。有兴趣的读者可以参考笔者关于序列化的文章。

## ArrayList的初始化

ArrayList提供了三个构造函数。下面我们依次来分析



- `public ArrayList(int initialCapacity)` 当我们初始化的时候，给ArrayList指定一个初始化大小的时候，就会调用这个构造方法。

```
List<String> myList = new ArrayList<String>(7);
```

源码中这个方法的实现如下

```
/**
 * Constructs an empty list with the specified initial capacity.
 *
 * @param initialCapacity the initial capacity of the list
 * @throws IllegalArgumentException if the specified initial capacity
 *         is negative
 */
public ArrayList(int initialCapacity) {
    if (initialCapacity > 0) {
        this.elementData = new Object[initialCapacity];
    } else if (initialCapacity == 0) {
        this.elementData = EMPTY_ELEMENTDATA;
    } else {
        throw new IllegalArgumentException("Illegal Capacity: "+
                                         initialCapacity);
    }
}
```

这里的EMPTY\_ELEMENTDATA 实际上就是一个共享的空的Object数组对象。

```
/**
 * Shared empty array instance used for empty instances.
 */
private static final Object[] EMPTY_ELEMENTDATA = {};
```

上述代码很容易理解，如果用户指定的初始化容量大于0，就new一个相应大小的数组，如果指定的大小为0，就复制为共享的那个空的Object数组对象。如果小于0，就直接抛出异常。

- `public ArrayList()` 默认的空的构造函数。  
我们一般会这么使用

```
myList = new ArrayList();
```

源码中的实现是

```
/**
 * Constructs an empty list with an initial capacity of ten.
 */
public ArrayList() {
    this.elementData = DEFAULTCAPACITY_EMPTY_ELEMENTDATA;
}
```

其中DEFAULTCAPACITY\_EMPTY\_ELEMENTDATA 定义为

```
/**
 * Shared empty array instance used for default sized empty instances. We
 * distinguish this from EMPTY_ELEMENTDATA to know how much to inflate when
 * first element is added.
 */
private static final Object[] DEFAULTCAPACITY_EMPTY_ELEMENTDATA = {};
```

注释中解释的很清楚，就是说刚初始化的时候，会是一个共享的类变量，也就是一个Object空数组，当第一次add的时候，这个数组就会被初始化一个大小为10的数组。

- `public ArrayList(Collection<? extends E> c)` 如果我们想要初始化一个list，这个list包含另外一个特定的collection的元素，那么我们就可以调用这个构造函数。



我们通常会这么使用

```
Set<Integer> set = new HashSet<>();
    set.add(1);
    set.add(2);
    set.add(3);
    set.add(4);

ArrayList<Integer> list = new ArrayList<>(set);
```

源码中是这么实现的

```
/**
 * Constructs a list containing the elements of the specified
 * collection, in the order they are returned by the collection's
 * iterator.
 *
 * @param c the collection whose elements are to be placed into this list
 * @throws NullPointerException if the specified collection is null
 */
public ArrayList(Collection<? extends E> c) {
    elementData = c.toArray();
    if ((size = elementData.length) != 0) {
        // c.toArray might (incorrectly) not return Object[] (see 6260652)
        if (elementData.getClass() != Object[].class)
            elementData = Arrays.copyOf(elementData, size, Object[].class);
    } else {
        // replace with empty array.
        this.elementData = EMPTY_ELEMENTDATA;
    }
}
```

首先调用给定的collection的toArray方法将其转换成一个Array。

然后根据这个array的大小进行判断，如果不为0，就调用Arrays的copyOf的方法，复制到Object数组中，完成初始化，如果为0，就直接初始化为空的Object数组。

## ArrayList是如何动态增长

当我们像一个ArrayList中添加数组的时候，首先会先检查数组中是不是有足够的空间来存储这个新添加的元素。如果有的话，那就什么都不用做，直接添加。如果空间不够用了，那么就根据原始的容量增加原始容量的一半。

源码中是如此实现的：

```
/**
 * Appends the specified element to the end of this list.
 *
 * @param e element to be appended to this list
 * @return <tt>true</tt> (as specified by {@link Collection#add})
 */
public boolean add(E e) {
    ensureCapacityInternal(size + 1); // Increments modCount!!
    elementData[size++] = e;
    return true;
}
```

ensureCapacityInternal的实现如下：

```
private void ensureCapacityInternal(int minCapacity) {
    if (elementData == DEFAULTCAPACITY_EMPTY_ELEMENTDATA) {
        minCapacity = Math.max(DEFAULT_CAPACITY, minCapacity);
    }

    ensureExplicitCapacity(minCapacity);
}
```

DEFAULT\_CAPACITY为：

```
private static final int DEFAULT_CAPACITY = 10;
```



这也就实现了当我们不指定初始化大小的时候，添加第一个元素的时候，数组会扩容为10.

```
private void ensureExplicitCapacity(int minCapacity) {
    modCount++;

    // overflow-conscious code
    if (minCapacity - elementData.length > 0)
        grow(minCapacity);
}
```

这个函数判断是否需要扩容，如果需要就调用grow方法扩容

```
/**
 * Increases the capacity to ensure that it can hold at least the
 * number of elements specified by the minimum capacity argument.
 *
 * @param minCapacity the desired minimum capacity
 */
private void grow(int minCapacity) {
    // overflow-conscious code
    int oldCapacity = elementData.length;
    int newCapacity = oldCapacity + (oldCapacity >> 1);
    if (newCapacity - minCapacity < 0)
        newCapacity = minCapacity;
    if (newCapacity - MAX_ARRAY_SIZE > 0)
        newCapacity = hugeCapacity(minCapacity);
    // minCapacity is usually close to size, so this is a win:
    elementData = Arrays.copyOf(elementData, newCapacity);
}
```

我们可以看到grow方法将数组扩容为原数组的1.5倍，调用的是Arrays.copy方法。

在jdk6及之前的版本中，采用的还不是右移的方法

```
int newCapacity = (oldCapacity * 3)/2 + 1;
```

现在已经优化成右移了。

## ArrayList如何实现元素的移除

我们移除元素的时候，有两种方法，一是指定下标，二是指定对象

```
list.remove(3);//index
list.remove("aaa");//object
```

下面先来分析第一种，也就是

- public E remove(int index)  
源码中是如此实现的



```

/**
 * Removes the element at the specified position in this list.
 * Shifts any subsequent elements to the left (subtracts one from their
 * indices).
 *
 * @param index the index of the element to be removed
 * @return the element that was removed from the list
 * @throws IndexOutOfBoundsException {@inheritDoc}
 */
public E remove(int index) {
    rangeCheck(index);

    modCount++;
    E oldValue = elementData(index);

    int numMoved = size - index - 1;
    if (numMoved > 0)
        System.arraycopy(elementData, index+1, elementData, index,
            numMoved);
    elementData[--size] = null; // clear to let GC do its work

    return oldValue;
}

```

对于数组的元素删除算法我们应该很熟悉，删除一个数组元素，我们需要将这个元素后面的元素全部向前移动，并将size减1。

我们看到源码中，首先检查下标是否在可用范围内。然后调用System.arraycopy方法将右边的数组向左移动，并且将size减一，并置为null。

- public boolean remove(Object o)

源码中实现如下：

```

/**
 * Removes the first occurrence of the specified element from this list,
 * if it is present. If the list does not contain the element, it is
 * unchanged. More formally, removes the element with the lowest index
 * <tt>i</tt> such that
 * <tt>(o==null ? get(i)==null : o.equals(get(i)))</tt>
 * (if such an element exists). Returns <tt>true</tt> if this list
 * contained the specified element (or equivalently, if this list
 * changed as a result of the call).
 *
 * @param o element to be removed from this list, if present
 * @return <tt>true</tt> if this list contained the specified element
 */
public boolean remove(Object o) {
    if (o == null) {
        for (int index = 0; index < size; index++)
            if (elementData[index] == null) {
                fastRemove(index);
                return true;
            }
    } else {
        for (int index = 0; index < size; index++)
            if (o.equals(elementData[index])) {
                fastRemove(index);
                return true;
            }
    }
    return false;
}

```

我们可以看到，这个remove方法会移除数组中第一个符合的给定对象，如果不存在就什么也不做，如果存在多个只移除第一个。

fastRemove方法如下



```
/**
 * Removes the first occurrence of the specified element from this list,
 * if it is present. If the list does not contain the element, it is
 * unchanged. More formally, removes the element with the lowest index
 * <tt>i</tt> such that
 * <tt>(o==null ? get(i)==null : o.equals(get(i)))</tt>
 * (if such an element exists). Returns <tt>true</tt> if this list
 * contained the specified element (or equivalently, if this list
 * changed as a result of the call).
 *
 * @param o element to be removed from this list, if present
 * @return <tt>true</tt> if this list contained the specified element
 */
public boolean remove(Object o) {
    if (o == null) {
        for (int index = 0; index < size; index++)
            if (elementData[index] == null) {
                fastRemove(index);
                return true;
            }
    } else {
        for (int index = 0; index < size; index++)
            if (o.equals(elementData[index])) {
                fastRemove(index);
                return true;
            }
    }
    return false;
}
```

可以理解为简化版的remove (index) 方法。

## ArrayList小结

- ArrayList是List接口的一个可变大小的数组的实现
- ArrayList的内部是使用一个Object对象数组来存储元素的
- 初始化ArrayList的时候，可以指定初始化容量的大小，如果不指定，就会使用默认大小，为10
- 当添加一个新元素的时候，首先会检查容量是否足够添加这个元素，如果够就直接添加，如果不够就进行扩容，扩容为原数组容量的1.5倍
- 当删除一个元素的时候，会将数组右边的元素全部左移

Java Collection实现原理分析 (/nb/15099409)

© 著作权归作者所有



六尺帐篷 (/u/f8e9b1c246f1)

写了 245418 字，被 15240 人关注，获得了 1429 个喜欢  
(/u/f8e9b1c246f1)

如果觉得我的文章对您有用，请随意赞赏。您的支持将鼓励我继续创作！

赞赏支持

❤ 喜欢 35



更多分享

(http://cwb.assets.jianshu.io/notes/images/154407f





写下你的评论...

6条评论

只看作者

关闭评论

按喜欢排序 按时间正序 按时间倒序



z\_xm\_\_ (/u/a1da343fe466)  
2楼 · 2017.08.08 08:19  
(/u/a1da343fe466)  
类似于stl vector

👍 赞    💬 回复



萤火之森ss (/u/2f5f38c8142b)  
3楼 · 2017.08.08 09:06  
(/u/2f5f38c8142b)  
稳的一匹

👍 赞    💬 回复



longsh (/u/86dc44fb59b4)  
4楼 · 2017.08.08 09:24  
(/u/86dc44fb59b4)  
厉害了

👍 赞    💬 回复



imyundong (/u/14329fde947f)  
5楼 · 2017.08.08 09:27  
(/u/14329fde947f)  
最后一段代码是错了吧，那不是fastRemove方法的代码

👍 赞    💬 回复

六尺帐篷 (/u/f8e9b1c246f1): @imyundong (/users/14329fde947f) 嗯，放成remove的代码了。  
2017.08.08 10:46    💬 回复

✎ 添加新评论



Susie\_Memories (/u/dc8c2aa4f380)  
6楼 · 2017.08.09 10:43  
(/u/dc8c2aa4f380)  
赞一个

👍 赞    💬 回复

被以下专题收入，发现更多相似内容

🔧 投稿管理

+ 收入我的专题



Android... (/c/5139d555c94d?utm\_source=desktop&utm\_medium=notes-included-collection)



Android开发 (/c/d1591c322c89?utm\_source=desktop&utm\_medium=notes-included-collection)






Android... (/c/58b4c20abf2f?utm\_source=desktop&utm\_medium=notes-included-collection)



小金库 (/c/27d03e4a4bd2?utm\_source=desktop&utm\_medium=notes-included-collection)



-  Android知识 (/c/3fde3b545a35?utm\_source=desktop&utm\_medium=notes-included-collection)
-  程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)
-  程序员首页投稿 (/c/89995286335f?utm\_source=desktop&utm\_medium=notes-included-collection)

展开更多 ▾

推荐阅读 更多精彩内容 > (/)

LintCode 字符大小写排序 (/p/badafb18ab1a?utm\_campaign=maleskine...

题目 给定一个只包含字母的字符串，按照先小写字母后大写字母的顺序进行排序。 注意事项 小写字母或者大写字母他们之间不一定要保持在原始字符串中的相对位置。 样例给出"abAcD"，一个可能的答案为"acbA..."

六尺帐篷 (/u/f8e9b1c246f1?utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

Top 10 Methods for Java Arrays (/p/e7ff7421e110?utm\_campaign=mal...

1 声明一个array 2 打印一个array 3 从array创建一个list 4 检查array中是否存在某个元素 5 连接两个array 6 Declare an array inline 7 将一个list转为array 8 array转为set

六尺帐篷 (/u/f8e9b1c246f1?utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

之所以单身，是因为不想谈恋爱 (/p/0ac3ce57e55a?utm\_campaign=maleskine&utm\_content=note&utm...

文 | 一个悦己 -01- 为什么还不恋爱呢？很多人觉得二十多岁还单身的人不是长得丑就是性格不好，但事实相反，她们很漂亮，性格也很好。或许很多人的...

一个悦己 (/u/a47b65385096?utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

我的诗和远方里，再无你 (/p/10ed5c312aba?utm\_campaign=maleskine&utm\_content=note&utm...

浓暮秋景，凄落晚亭，光阴它说，人生如水，东流无歇。可我希望，如果时光可以倒流，我再也不要辜负你的深情。 2017.10.22 周日 阴雨 01 门前老树枯枝...

与君成悦 (/u/51995510ee0a?utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

你有多久，没有小心翼翼的喜欢一个人了 (/p/1395c4b...utm\_campaign=maleskine&utm\_content=note&utm...

01 曾经在知乎上看到过这么一个问题：喜欢一个人是一种什么样的感觉？ 一个高赞的答案这么写着：大概就是你小心翼翼的跟在她身后，选取所有的“偶然”...

潇洒小兔 (/u/2d5ddf6d2b96?utm\_campaign=maleskine&utm\_content=user&utm\_medium=pc\_all\_hots&utm\_source=recommendation)

