链滴

jdk 源码 --ArrayList

作者：fyzzz

原文链接：https://ld246.com/article/1561962739809

来源网站：链滴

本文章基于jdk1.8_171

ArrayList介绍

java中用的最多（个人感觉）的一个集合，内部维护着一个数组，方便，不用像数组一样事先给定大小。

成员变量

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

默认容量，如果新建一个对象时没有指定容量，会新建一个空数组，并在第一次添加元素的时候把容量改为10（DEFAULT_CAPACTIY），当然，前提是添加元素的个数不大于10个。

```
private static final Object[] EMPTY_ELEMENTDATA = {};
```

一个空数组。

```
private static final Object[] DEFAULTCAPACITY_EMPTY_ELEMENTDATA = {};
```

也是一个空数组，新建对象时如果没有指定容量，就把这个数组作为内部的数组。与EMPTY_ELEMENTDATA 的区别是，第一次添加元素时，如果数组是DEFAULTCAPACITY_EMPTY_ELEMENTDATA ，量就变为DEFAULT_CAPACTIY 。（添加元素不超过10个）。

```
transient Object [] elementData;
```

集合里真正放数据的地方，　这个数组的长度就是集合的容量。

```
private int size;
```

集合里实际的元素个数，不是数组长度。

```
private static final int MAX_ARRAY_SIZE = Integer.MAX_VALUE - 8;
```

可以设置数组的最大长度，为什么-8，因为数组需要8byte存储本身的大小。参考链接

构造器

```
public ArrayList(){
    this.elementData = DEFAULTCAPACITY_EMPTY_ELEMENTDATA;
}
```

无参构造器，把DEFAULTCAPACITY_EMPTY_ELEMENTDATA作为内部数据，添加第一个元素时容量增长为DEFAULT_CAPACTIY 。

```
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);
```

```
        }
    }
```

新建一个容量为initialCapacity的对象，initialCapacity小于0时抛出异常。

```
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;
        }
}
```

新建一个对象，并把集合c的数组赋予elementData ，如果集合c为空，elementData = EMPTY_ELEENTDATA。注意，此时数组长度就是size。

方法

```
public boolean add(E e) {
    ensureCapacityInternal(size + 1);  // Increments modCount!!
    elementData[size++] = e;
    return true;
}
```

常用方法，向集合内添加一个元素。先看容量够不够添加元素，不够就扩容。然后在末尾添加元素，size+1。

```
private void calculateCapacity(Object[] elementDate,int  minCapacity){
    if (elementData == DEFAULTCAPACITY_EMPTY_ELEMENTDATA) {
        minCapacity = Math.max(DEFAULT_CAPACITY, minCapacity);
    }
    return minCapacity;
}
private  void  ensureCapacityInternal(int  minCapacity) {
    ensureExplicitCapacity(calculateCapacity(elementDate,minCapacity));
}
private void ensureExplicitCapacity(int minCapacity) {
    modCount++;

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

这三个放在一起说，表示扩容前的检查。如果内部数组是DEFAULTCAPACITY_EMPTY_ELEMENTDAA并且需要的最小容量不大于

DEFAULT_CAPACITY，那么就把容量定成DEFAULT_CAPACITY。最后如果需要的容量大于内部数组度，则进行扩容。注意：modCount表示此集合被修改的次数，这个变量定义在父类AbstractList中。

```
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);
}

private static int hugeCapacity(int minCapacity) {
    if (minCapacity < 0) // overflow
        throw new OutOfMemoryError();
    return (minCapacity > MAX_ARRAY_SIZE) ?
        Integer.MAX_VALUE :
        MAX_ARRAY_SIZE;
}
```

真正的扩容方法，首先扩容1.5倍，如果还小于所需最小容量，那就把所需最小容量设置为新的容量如果新的容量大于MAX_ARRAY_SIZE，把int的最大值作为容量。注意，如果你的集合需要这么多容，那应该考虑一下拆分了。

```
public void ensureCapacity(int minCapacity) {
    int minExpand = (elementData != DEFAULTCAPACITY_EMPTY_ELEMENTDATA)
        // any size if not default element table
        ? 0
        // larger than default for default empty table. It's already
        // supposed to be at default size.
        : DEFAULT_CAPACITY;

    if (minCapacity > minExpand) {
        ensureExplicitCapacity(minCapacity);
    }
}
```

ArrayList对外提供的扩容方法，需要大量添加元素时可以先手动扩容。

```
public void add(int index, E element) {
    rangeCheckForAdd(index);
    ensureCapacityInternal(size + 1);  // Increments modCount!!
    System.arraycopy(elementData, index, elementData, index + 1,
                     size - index);
    elementData[index] = element;
    size++;
}

private void rangeCheckForAdd(int index) {
    if (index > size || index < 0)
        throw new IndexOutOfBoundsException(outOfBoundsMsg(index));
}
```

在指定位置添加元素。首先检查index是否合法，然后看情况进行扩容，再然后把index和index之后元素往后移一位，最后把要添加的元素放在下标index的位置上。

```
public int indexOf(Object o) {
    if (o == null) {
        for (int i = 0; i < size; i++)
            if (elementData[i]==null)
                return i;
    } else {
        for (int i = 0; i < size; i++)
            if (o.equals(elementData[i]))
                return i;
    }
    return -1;
}
```

查找指定元素的位置，显示的是第一次出现的位置，支持null的查找。如果找不到，返回-1。lastIndeOf方法同理。

未完待续。。。