初探 ArrayList 动态扩容原理

作者：Gouzhong1223
原文链接：https://ld246.com/article/1582881797965
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

1.概述

传统的 Java 数组都有一个通病，就是不支持动态扩容，我们在初始化一个数组的时候一旦制定了容，那么在后面遇到数组越界的时候，就会抛出异常，这对于我们的开发是非常的不变的，于是就有了种对数组进行了封装的容器类：ArrayList。ArrayList 是一种支持动态扩容的容器，**其本质上是对数进行了封装**，所以，数组所持有的特性ArrayList 依然持有，比如说随机访问。

2.ArrayList初始化

2.1ArrayList 的空参构造如下：

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

这里可以看到，当我们在初始化一个 ArrayList 的时候，成员变量被赋予一个叫做

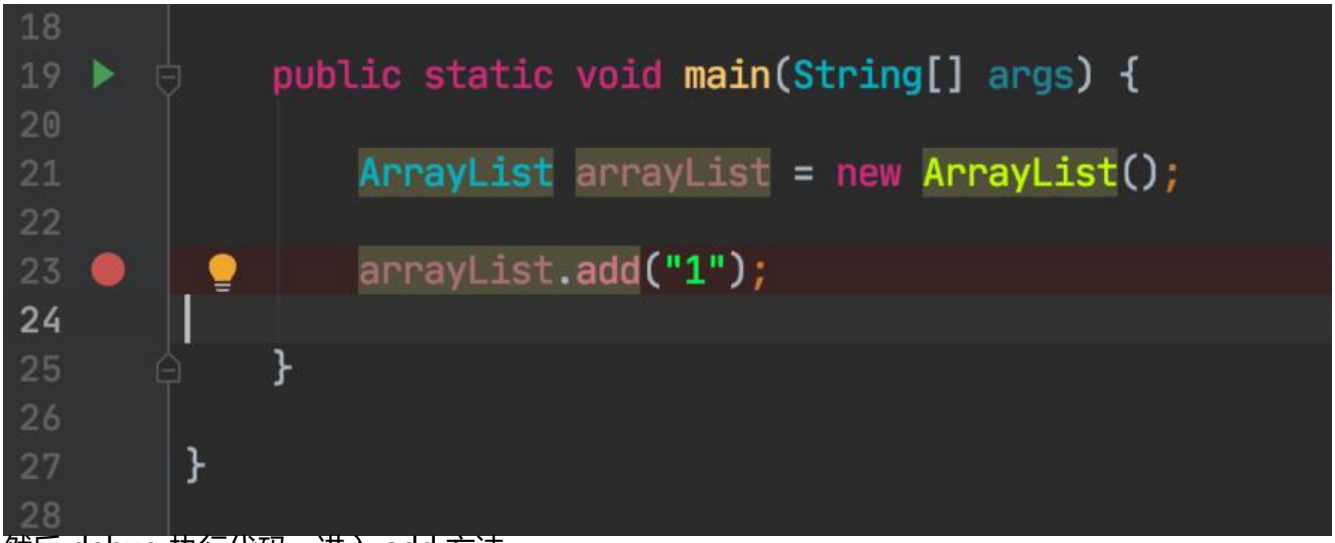
DEFAULTCAPACITY_EMPTY_ELEMENTDATA的成语变量，下面来看一下 elementData 还有DEFAUTCAPACITY_EMPTY_ELEMENTDATA具体的含义。

这个地方就可以很明显的看出来， 在初始化一个 ArrayList 的时候并没有指定容量，而是把一个默认空数组赋予 ArrayList 的 elementData，很明显，ArrayList 装在元素的功能其实是 elementData 来成的，所以，刚刚初始化的 ArrayList 数组可以堪称是 size 是 0 的数组。

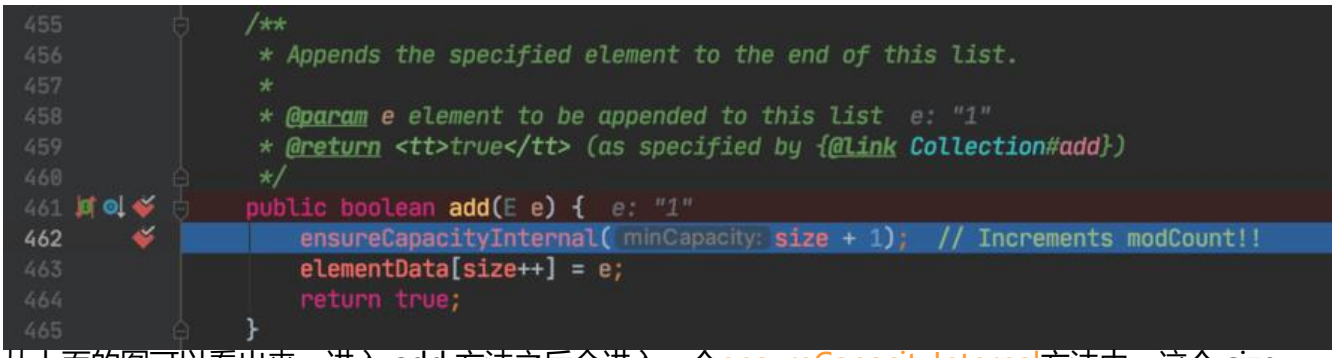
2.2ArrayList 初始化容量

初始化出来一个空的容器之后，开始尝试着给这个容器添加一些元素，下面，很好奇这个容器是怎么一个容量为零的数组中添加元素的。

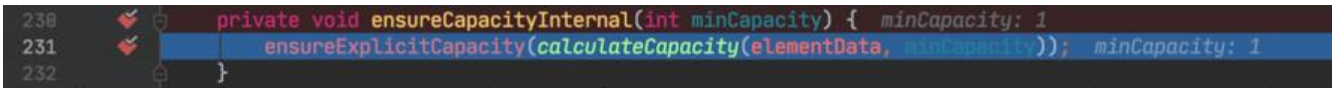
现在就往数组中添加一个字符串，在 add 方法执行的地方，打上一个断点



然后 debug 执行代码，进入 add 方法

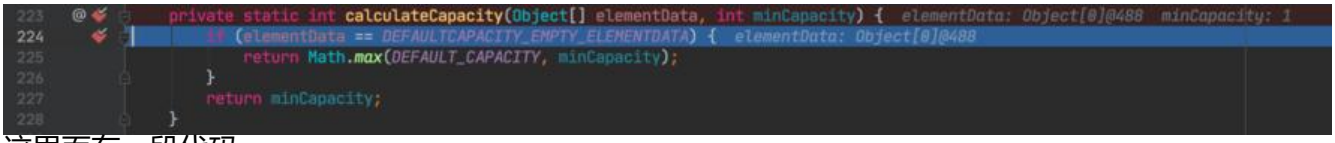


从上面的图可以看出来，进入 add 方法之后会进入一个ensureCapacityInternal方法中，这个 size 是初始化 ArrayList 默认的容量，是 **0**，我们进入ensureCapacityInternal，看一看这里面是怎么执的



这里进入ensureCapacityInternal方法之后，会调用一个ensureExplicitCapacity方法，而这个方法

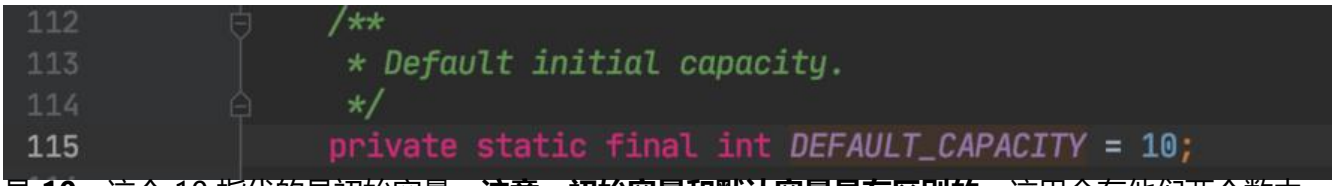
依赖一个叫做calculateCapacity的方法，点进去看，这个里面是怎么执行的



这里面有一段代码

```
if (elementData == DEFAULTCAPACITY_EMPTY_ELEMENTDATA) {
        return Math.max(DEFAULT_CAPACITY, minCapacity);
    }
```

看到这个地方的DEFAULTCAPACITY_EMPTY_ELEMENTDATA是不是很眼熟啊，这个不就是最开始始化 ArrayList 的时候赋予的默认数组么？所以，这个方法就是判断这个需要添加元素的数组是否是初始化的数组的，如果是，那就会再做一个比较，看一下怎么比较的，看这个minCapacity，翻译过就是最小容量，什么最小容量呢？意思就是说，当前这个ArrayList的 add 方法执行成功之后，数组一个长度，我们当前的ArrayList 是刚刚初始化的，size 是 0，如果成功添加这个元素之后，那就是 1 所以minCapacity就是 1，和它作比较的DEFAULT_CAPACITY呢？



是 **10**，这个 10 指代的是初始容量，**注意：初始容量和默认容量是有区别的。**这里会在他们两个数中去一个大的数，然后返回。按照逻辑，这里返回的应该是 10，这个 10 将会被传递给ensureExplicitCpacity，我们再进入这个方法里面看一看这是怎么执行的。

这个 modCount 指的是对这个 ArrayList 的操作次数，然后就会进入一个判断：

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

这里就会把帮刚刚重新初始化的一个容量和当前 ArrayList 的大小做差比较大小，看重新初始化的这容量是不是可以再容下一个元素。因为当前的 ArrayList 是刚刚初始化的，里面什么都没有，所以就跳入grow方法，并将我们最新初始化的一个容量传递过去，点进去看一下grow方法是怎样执行的。

```java
/**
 * 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);
}
```

在这个grow方法中，会做两个判断，说一下执行步骤：

- 首先把这个 当前ArrayList 的底层容器 elementData 里面的长度取出来
- 然后将老容量加上老容量乘以 0.5 倍的数值定义为新容量（这里老容量本来就是 0，乘以 1.5 倍之还是 0）
- 比较新容量和当前重新初始化的一个数组容量（10）的大小，如果新容量小于这个，就把新容量重赋值为重新新初始化的数组容量。
- 然后再判断新容量和MAX_ARR_SIZE 的数值大小

```java
/**
 * The maximum size of array to allocate.
 * Some VMs reserve some header words in an array.
 * Attempts to allocate larger arrays may result in
 * OutOfMemoryError: Requested array size exceeds VM limit
 */
private static final int MAX_ARRAY_SIZE = Integer.MAX_VALUE - 8;
/**
 * A constant holding the maximum value an {@code int} can
 * have, 2<sup>31</sup>-1.
 */
@Native public static final int   MAX_VALUE = 0x7fffffff;
```

如果大于这个数，name 就会给新容量重新赋值为一个更大的数

- 然后就会把当前的 elementData 拷贝到一个新的，容量为 10 的一个数组中

至此，ArrayList 的第一次初始化容量就完毕了

这时候就要开始往里面添加元素了

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

这里 size 是一个后自增的使用方法，添加完成之后，size 就会+1。

3.ArrayList 其余时候元素的添加

3.1 添加元素之后不会超过 ArrayList 的容量

当我们再次往里面添加元素的时候，当走到calculateCapacity方法的时候，判断为不为新初始化的数时，就会直接返回当前ArrayList 的 size+1。

走到ensureExplicitCapacity方法的时候，也就自然**不会触发我们的扩容机制**了

后面的就是添加元素，然后 size+1 了。

3.2添加元素之后超过最大容量

比如说，当我们添加第十一个元素的时候，原本的 ArrayList 容量为 10，这时候，走到ensureExplici Capacity方法的时候，添加之后的最小容量是大于我们的 elementData 的，所以，又会进入到grow 法中，在这里，会把当前数组的容量和当前容量的 0.5 倍加起来，赋值给一个新的容量，然后将整个 e ementData 数组拷贝到这个拥有新容量的数组中，老的数组 gc 会周期就会将它处理掉。

完结