# Java知识点 手写ArrayList

ArrayList是最常用的类了，很多面试都要求自己动手写一个，其实就是考察应聘者对ArrayList的熟悉程度。我试着挑几个重要的方法写写。

因为是熟悉其核心思想，所以很多枝叶能略过去的就略过去了，比如抛出的异常，父类的实现等，另外为了与原ArrayList区分，很多字段和方法也改了名。

## 元素



这里额外要说两点：

DEFAULT\_EMPTY\_ARR。

它对应ArrayList里的字段是DEFAULTCAPACITY\_EMPTY\_ELEMENTDATA。

这是1.8才有的，主要是与EMPTY\_ARR区分。

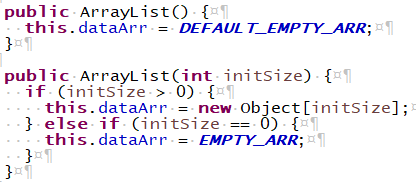
如果我们在构造ArrayList的时候传了参，比如设置默认的容量，传了Collection。那么在构造的时候，会判断一下这个容量值是否为0，如果为0则内部的数组就为EMPTY\_ARR。

如果没有传参，内部数组就是DEFAULT\_EMPTY\_ARR。

第二点是内部数组的transient。这代表本数组不做序列化。那么问题来了，这个数组存放着List的内容，不序列化的话，我们一旦传输这个ArrayList，接收方反序列化的话，这里的内容就全没了。

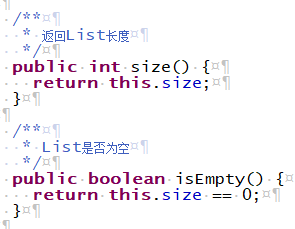
其实ArrayList的序列化与反序列化执行的是writeObject和readObject，因为存数据的这个数组其实是个具有缓存性质的数组，它的容量一般都比实际大，所以序列化的时候，就把缓存数组中为空的那些元素给滤掉了，节省序列化与反序列化的空间和时间。

## 构造函数



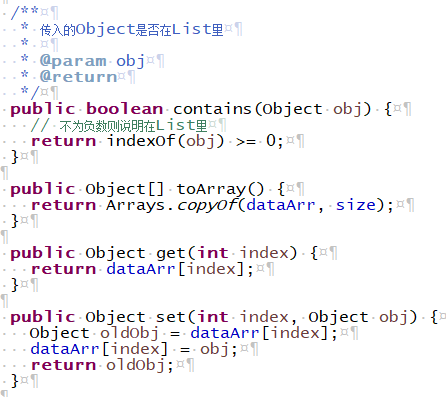
这里少写了一个传Collection的构造函数。

## 方法

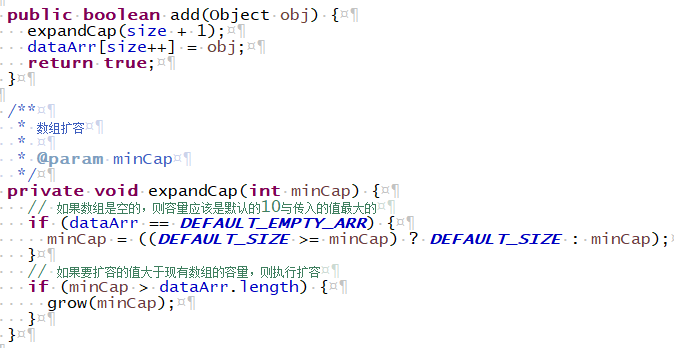




这里少一个lastIndexOf的方法，就是倒着遍历数组。



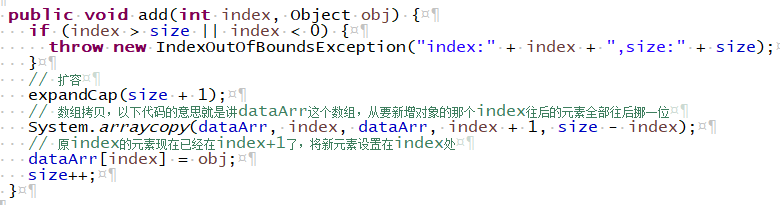
着重要说一下增加元素，主要是麻烦在数组的扩容上。





扩容的步骤还是蛮繁琐的：

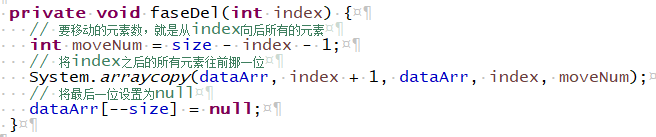
1. 传入一个要扩容的值（在add方法中就是size+1），先看一下原数组是不是空，如果为空则扩容为默认值（10）与传入值中，最大的那个。
2. 如果要扩容的值确实比原数组的容量大，执行扩容操作。
3. 先获取原数组容量的1.5倍（这里说一下，jdk7、8是右移运算（除2）+原长度，结果是1.5倍。JDK6的是\*3/2+1,1.5倍+1。位移的效率高），如果这个数比传入的数还小，则扩容传入的数。
4. 如果要扩容的数比ArrayList的最大长度还要大，那么在不溢出的情况下，要么选ArrayList的最大长度，要么是Integer的长度。这二者就差8个数。



在指定位置插入元素。

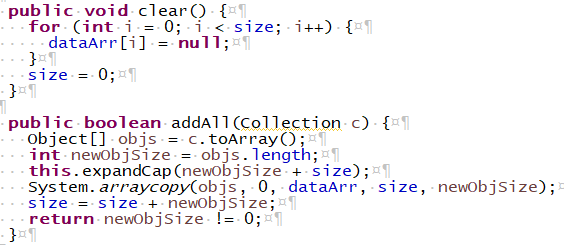
首先要扩容，然后将数组中要插入位置之后的所有元素向后挪一位，再将新元素设置在指定位置上。





删除指定元素，要先遍历数组，获取此元素在数组的具体位置。

删除指定位置的元素，就是讲此元素之后的所有元素都往前挪一位，然后把最后位置的元素置为Null。



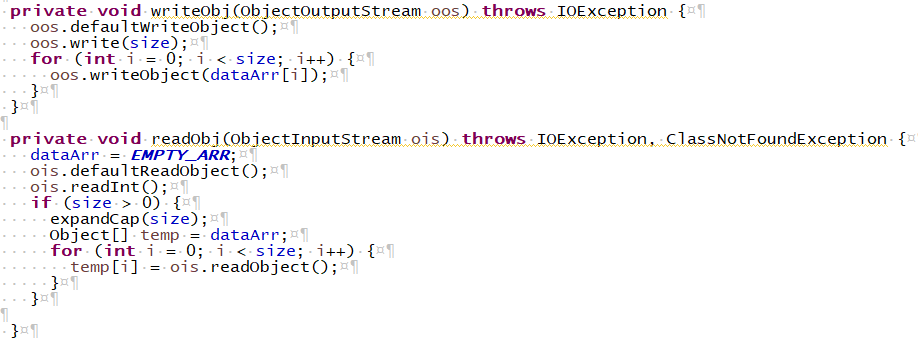
清空方法就是讲内部的数组每个元素置为Null，把size设为0。

将Collection增加至List里，首先是将Collection转为数组，然后将原List数组的长度扩容为原长度+新数组长度。

然后将新数组复制进原数组。

System.arraycopy(要复制的数组，要复制数组的开始位置，被添加的数组，从被添加的数组第几个位置开始添加，添加多少个元素)。

List的Size增加。



在之前说元素的时候指出数组dataArr是transient，即不会被序列化。那么ArrayList的序列化、反序列化就是靠writeObj和readObj两个方法。