复习计划——链表与 Java

作者：tianxin
原文链接：https://ld246.com/article/1480952146211
来源网站：链滴

<h2 id="个人复习计划--Java中链表的实现-部分分析--待更新-">个人复习计划——Java 中链表的现（部分分析）（待更新）</h2>
声明
<blockquote>
写给今天的自己，只是简单总结下今天所整理的知识点。
当初学习数据结构的时候第一篇便是链表，链表也是一个再简单不过的概念，那么看看那些大师级的物是如何编写 Java 中链表的实现的。本篇暂时先分析今天刚刚看到的 ArrayList 的源码 （jdk8 版本和自己遇到的困惑
</blockquote>
<hr>
<h3 id="12月5日所学">12 月 5 日所学</h3>
<h2 id="ArrayList-基于数组的链表">ArrayList 基于数组的链表</h2>
<h3 id="整体思路">整体思路</h3>
1. 一个在简单不过的数组来存放元素
<pre><code class="highlight-chroma">transient Object[] elementData; // non-private to simplify nested class access
</code></pre>
这里有两个困惑点

关键字 <code>transient</code> 的作用
为什么没有使用 <code>private</code> 关键字，而是默认的范围

2. 易忽略点： index 下标的检查
<blockquote>
当涉及到 add(index) , get(index), remove(index) 等操作的时候，首先需要对 index 进行范围检查
</blockquote>
<pre><code class="highlight-chroma">if (index < 0 || index > size) {
    throw new IndeOutofBoundException("Index: "+index+", Size: "+size);
}
</code></pre>
3. 数组扩容的实现
数组扩容的道理很简单，但如何去制定这个扩量的度呢？我们看下他们是思路是什么。

Step 1： 数组的创建。（分两种情况）

创建数组时传递了数组的创建时的大小
创建数组时没有传递任何信息（数组默认为一个长度为 0 的数组{}）

Step 2： 数组的扩容

当数组长度为 0 时，则扩充到 max(minCapacity, DEFAULT_CAPACITY)
当数组长度不为 0 是，扩充 1.5 倍
当数组长度超过最大限制时，抛出异常

4. 易忽略点： 删除操作

版本一：数组实现的链表删除一个节点的思路十分简单：从即将删除的 index 节点开始，将后面 index + 1 到 size（元素个数）向前平移一位。

<pre><code class="highlight-chroma">// 参数的意义分别为 : src, srcPos, dest, destPos, length
System.arraycopy elementData, index + 1, elementData, index, size - index);
size--;
</code></pre>

漏洞一：虽然实现了功能点，却漏掉了一个重要的内容，链表的最后一个对象依然被引用，所以们需要释放它

<pre><code class="highlight-chroma">// 参数的意义分别为 : src, srcPos, dest, destPos, length
System.arraycopy elementData, index + 1, elementData, index, size - index);
elementData[--siz] = null; //由GC垃圾回收器回收未被引用的对象
</code></pre>

漏洞二：删除操作一定需要平移吗？ NO！当删除的节点是尾节点时，此时不必再向前平移数组

<pre><code class="highlight-chroma">int numMoved = size - index - 1;
if (numMoved > 0) {
    System.arraycopy(elementData, index + 1, elementData, index, size - index);
}
elementData[--siz] = null; //由GC垃圾回收器回收未被引用的对象
</code></pre>
<h3 id="transient-关键字">transient 关键字</h3>
只能修饰变量，是在类的序列化和反序列化中起到作用。被 transient 关键字修饰到的成员变量会再被序列化（对于静态变量而言，永远都不能被序列化）