链滴

java 中数组拷贝详细介绍（arraycopy,addAll,序列化和反序列化）

作者：oldcaptain

原文链接：https://ld246.com/article/1375445286437

来源网站：链滴

<h2> 一、场景 </h2>
            有时候由于需要可能需要拷贝数组的中的数，从而是我们能更好的操作数据
<h2> 二、使用方法 </h2>
           在这个时候我们一般会想到数组的拷贝的方式，当然种方式可以使用，一般有addALL，和System.arraycopy。但一定要注意你的数组中放的是什么数据是对象还是基本类型的数据。这一点很重要，接下来将通过测试代码详细介绍这些方法。
<h3>   1、addAll（）和System.arraycopy的区别： </h3>

            以下是java底层对addAll实现的方法。可以到实际上是调用了System.arraycopy
<pre>public boolean addAll(int index, Collection<? extends E> c) {
    if (index > size || index < 0)
        throw new IndexOutOfBoundsException(
        "Index: " + index + ", Size: " + size);
<pre><code class="highlight-chroma">Object[] a = c.toArray();
int numNew = a.l ngth;
ensureCapacity(si e + numNew); // Increments modCount

int numMoved = s ze - index;
if (numMoved & mp;gt; 0)
    System.arrayco y(elementData, index, elementData, index + numNew,
                     numMov d);

    System.arrayco y(a, 0, elementData, index, numNew);
size += numNew;
return numNew != 0;
}</pre>
</code></pre>
<h3>   2、测试放入基本数据类型，和对象之间的区别 </h3>

<pre>public class TestArrayCopy {
<pre><code class="highlight-chroma">public static void main(String[] args) {
    Student[] s = {n w Student(18,"Lucy"),new Student(20,"Lily")};
    Student[] sBak new Student[2];

  //arraycopy(Obje t src, int srcStartIndex, Object dest, int destStartIndex, int length)
  //从指定源数组复制一个数组，复制从指定的位置开始，到目标数组的指定位置结束。
  //srcStartIndex 原数组中要开始复制的第一个元素的位置

```
</span></span><span class="highlight-line"><span class="highlight-cl">  //destStartIndex: 目标数组中要开始替换的第一个元素的位置
</span></span><span class="highlight-line"><span class="highlight-cl">  //length: 要复制元素的个数
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(&amp;quot;没修改前&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    for(int i=0; i&amp;lt;s.length; i++) {
</span></span><span class="highlight-line"><span class="highlight-cl">        System.out.println(sBak[i] + &amp;quot;\t&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    }
</span></span><span class="highlight-line"><span class="highlight-cl">    System.arraycopy(s,0,sBak,0,s.length);
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(&amp;quot;----------------拷贝后----&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    for(int i=0; i&amp;lt;s.length; i++) {
</span></span><span class="highlight-line"><span class="highlight-cl">        System.out.println(sBak[i] + &amp;quot;\t&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    }
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println();
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(&amp;quot;---------------------------------&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(&amp;quot;修改后输出原数组&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    s[0].setAge(888);
</span></span><span class="highlight-line"><span class="highlight-cl">    s[0].setName(&amp;quot;MicXP.com&amp;quot;); //这里改的是sBak[0]引用的对像的值。
</span></span><span class="highlight-line"><span class="highlight-cl">    //sBak[0]=new tudent(88,&amp;quot;MicXP&amp;quot;); //这里是直接改sBak[0]引用地址。
</span></span><span class="highlight-line"><span class="highlight-cl">    for(int i=0; i&amp;lt;s.length; i++) {
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(s[i] + &amp;quot;\t&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    }
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println();
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(&amp;quot;---------------------------------&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(&amp;quot;修改后输出拷贝的数组&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    for(int i=0; i&amp;lt;s.length; i++) {
</span></span><span class="highlight-line"><span class="highlight-cl">    System.out.println(sBak[i] + &amp;quot;\t&amp;quot;);
</span></span><span class="highlight-line"><span class="highlight-cl">    }
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span></code></pre>
<p>}<br>
class Student {<br>
private int age ;<br>
```

```
private String name ;<br>
Student(int age, String name) {<br>
this.age = age;<br>
this.name = name;<br>
}<br>
public void setAge(int age) {<br>
this.age = age;<br>
}<br>
public int getAge() {<br>
return age;<br>
}<br>
public void setName(String name) {<br>
this.name = name;<br>
}<br>
public String getName() {<br>
return name;<br>
}<br>
@Override<br>
public String toString() {<br>
return "姓名：  "+ this.name +"\t"+"年龄:"+this.age;<br>
}<br>
}<br>
</p></pre><p></p>
<p> &amp;nbsp; 从以上输出结果可以看出，当我们修改了源数组中的对象，会影响目标数组的对象数据所以这种拷贝方式是不靠谱的，并没有真正将数据进行复制。 </p>
<p> 下面是测试基本数据类型的代码。 </p>
<pre>public class TestArr {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[] = {1,2,3,4};
        int b[] = {2,3,4,5};
          System.arraycopy(a,0,b,0,a.length);
          System.out.println("---------拷贝后-----" );
          System.out.println("a----------&amp;gt;");
          for(int i=0;i&amp;lt;a.length;i++){
              System.out.print(a[i]+",");
          }
          System.out.println();
          System.out.println("b----------&amp;gt;");
          for(int i=0;i&amp;lt;b.length;i++){
              System.out.print(b[i]+",");
          }
          a[3] = 45;
          System.out.println();
          System.out.println("--------修改原数组");
          System.out.println("a----------&amp;gt;");
          for(int i=0;i&amp;lt;a.length;i++){
              System.out.print(a[i]+",");
          }
          System.out.println();
          System.out.println("b----------&amp;gt;");
          for(int i=0;i&amp;lt;b.length;i++){
              System.out.print(b[i]+",");
          }
```

```
    }
<p>}<br>
</p></pre><p></p>
<p> 从以上数据结果可以看出，当我们拷贝数据后，修改源数组数据，不影响目标数组数据。 </p>
<p> 所以再数组拷贝的时候要注意这些问题，如果是对象则是拷贝引用。如果是基本数据类型，则
贝的是真正的数据。 </p>
<h2> 三、疑问 </h2>
<p>         有什么比较高效的真正拷贝数据的方法？clone？for循环再
新构造？如果你知道你通过评论告诉我，谢谢。 </p>
<h2> 四、疑问的答案 </h2>
<h3>       1、序列化和反序列化 </h3>
<p>          序列化和反序列化的定义：把Java对象转换为字节序
的过程称为对象的序列化。把字节序列恢复为Java对象的过程称为对象的反序列化。 </p>
<h3>       2、序列化和反序列化的应用场景 </h3>
<p>         永久性保存对象，保存对象的字节序列到本地文件中 </p>
<p>         通过序列化在网络中传递对象 </p>
<p>         通过序列化在进程间传递对象 </p>
<h3>        3、可以通过对象的序列化和反序列化来进行对象的深度拷贝
/h3>
<p>        (数组中的对象需要implements Serializable)  </p>
<pre>public static &lt;T&gt; List&lt;T&gt; deepCopy(List&lt;T&gt; src) throws IOException, Cl
ssNotFoundException {
        ByteArrayOutputStream byteOut = new ByteArrayOutputStream();
        ObjectOutputStream out = new ObjectOutputStream(byteOut);
        out.writeObject(src);
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">    ByteArrayInputStream byteIn = new ByteArrayInputStream(byteOut.toByteArray());
</span></span><span class="highlight-line"><span class="highlight-cl">    ObjectInputStr
am in = new ObjectInputStream(byteIn);
</span></span><span class="highlight-line"><span class="highlight-cl">    @SuppressWar
ings(&amp;quot;unchecked&amp;quot;)
</span></span><span class="highlight-line"><span class="highlight-cl">    List&amp;lt;T&
mp;gt; dest = (List&amp;lt;T&amp;gt;) in.readObject();
</span></span><span class="highlight-line"><span class="highlight-cl">    return dest;
</span></span><span class="highlight-line"><span class="highlight-cl">}  &lt;/pre&gt;
</span></span></code></pre>
</pre></pre>
```