25从零开始学Java25之数组扩容与数组拷贝

作者：孙玉昌，昵称【**一一哥**】，另外【**壹壹哥**】也是我哦

千锋教育高级教研员、CSDN博客专家、万粉博主、阿里云专家博主、掘金优质作者

前言

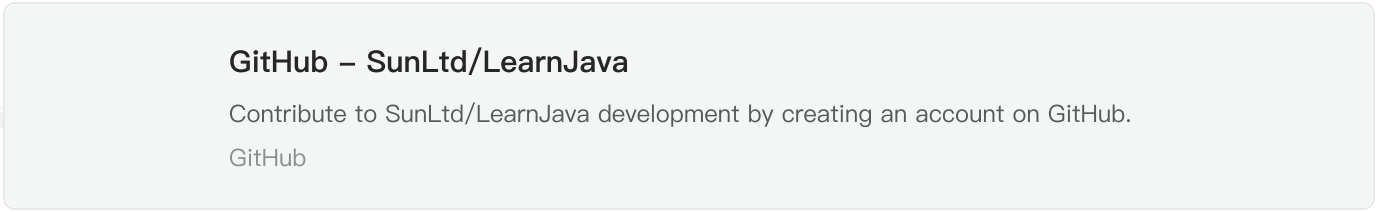
在上一篇文章中，**壹哥**给大家讲解了数组的创建、初始化及遍历方式，这些是我们学习数组的基础。其实数组的内容非常多，今天这篇文章，**壹哥**主要带大家学习数组的扩容、缩容及拷贝，内容同样重要，希望你不要走神哦。

----------**前戏已做完，精彩即开始**----------

全文大约【3000】字，不说废话，只讲可以让你学到技术、明白原理的纯干货！本文带有丰富案例及配图视频，让你更好地理解和运用文中的技术概念，并可以给你带来具有足够启迪的思考.....

配套开源项目资料

Github：



Gitee：

一. 数组内存分析(重点)

1. 简介

Java的内存，可以分为**栈、堆、方法区、本地方法区、程序寄存器**等几个核心部分。这一块的内容，以后**壹哥**会专门编写文章进行介绍，对于初学者来说，这还不适合我们学习。但是我们现在要先对以下三个概念有所了解：

> **栈：栈中可以存储基本类型的数据和引用类型的地址。特点: 先进后出，一般空间比较小，存取速度较快。**

> 堆：堆中可以存储引用类型的数据。特点: 空间比较大，存储速度相对较慢。
>
> 方法区：方法区中可以存储字符串常量池、静态数据、代码和类的元数据。

我们知道，**数组属于引用类型，而数组的引用变量(数组名称)只是一个地址引用。**这个引用变量可以指向任何有效的内存空间，只有当这个引用指向有效的空间时，才可以通过引用去操作数组中真正的数据元素。所以**数组的引用变量(数组名称)是存储在栈空间中，但真正的数组数据是存储在堆空间中。**

2. 代码案例

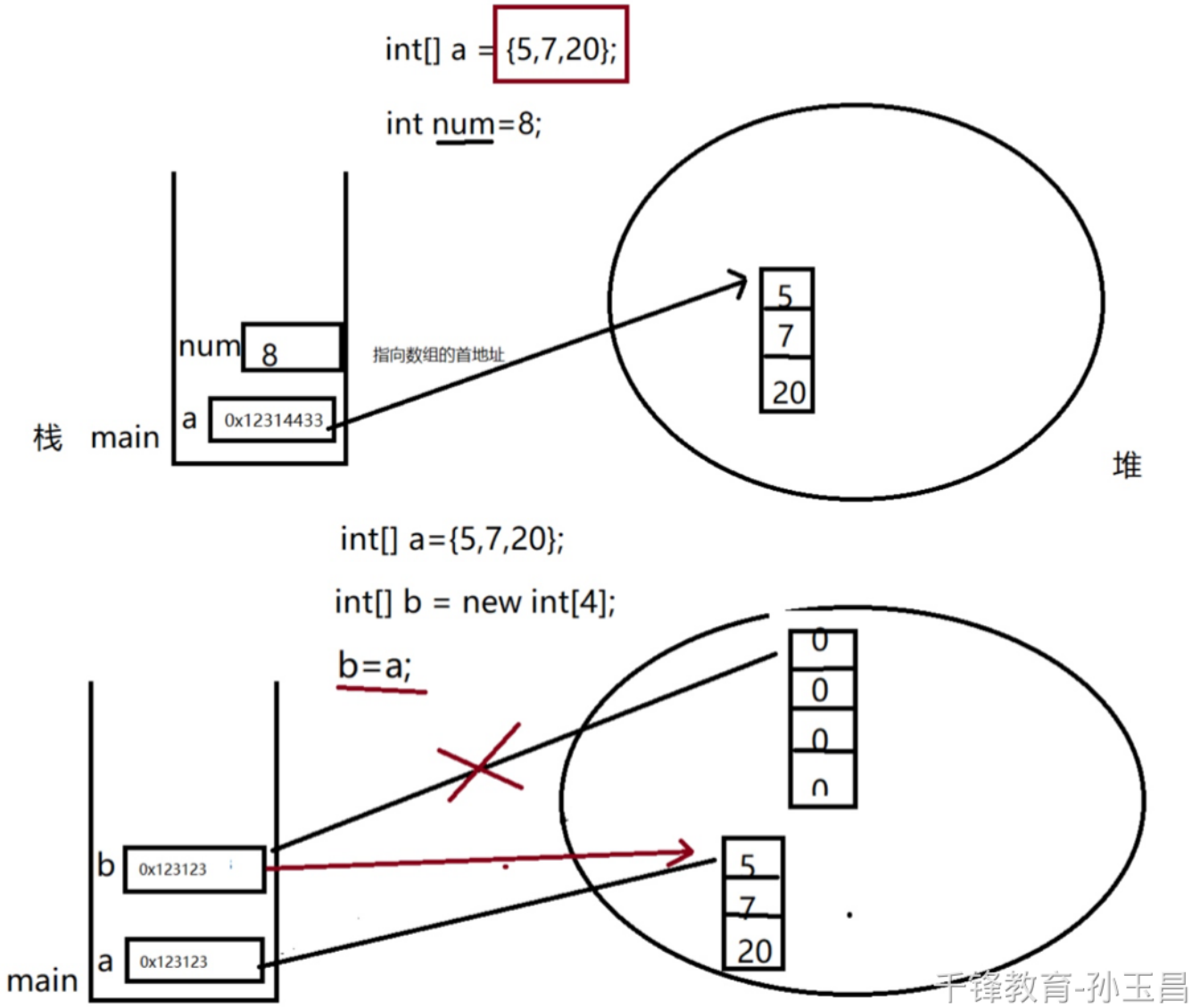
为了让大家更好地理解数组的内存结构，接下来**壹哥**给大家设计一个代码案例，然后给大家分析一下这个数组的内存结构。

Java | 复制代码

```java
/**
 * @author ——哥Sun QQ: 2312119590 CSDN、掘金、知乎找我哦
 */
public class Demo01 {

    public static void main(String[] args) {
        //使用静态初始化的方式初始化一个数组a
        //a存放在栈中，a的值是数组的地址，数组的真正数据{5,7,20}存放在堆中
        int[] a = {5,7,20};
        System.out.println("a的长度为:" + a.length);//3

        //整型变量，存放在栈中
        int num =8;
        System.out.println("num:"+num);

        //定义一个新的数组b
        int[] b=new int[4];
        System.out.println("b的长度是:"+b.length);

        //将a赋值给b，是b的指向改变了，但b原先对应的数组依然存在
        b=a;
        System.out.println("b的长度是:"+b.length);
    }

}
```

3. 内存分析

为了让各位更好地理解基本类型的数据和数组的内存结构，**壹哥**再给大家绘制下面一张图。



根据上面的代码和下面的内存分析图，我们可以得到如下结论：

- 变量a存放在栈中，a的值是数组的首地址，数组的真正数据{5,7,20}存放在堆中；
- 整型变量num存放在栈中；
- 定义新的数组b，数组名称b存放在栈中，b的数据在堆中；
- 将a赋值给b，此时b的指向改变了，但b原先对应的数组依然存在，此时b指向原先a对应的数组数据。

二. 数组扩容与缩容

1. 扩容简介

壹哥在前面给大家说过，数组一旦创建初始化后，其长度就不能被改变。但是有的小伙伴就说了，”不对啊，我看别人的文章说，可以往数组中增加很多新数据啊.....“。那如果是这样，假如我们一开始定义一个长度为5的数组，然后想把10个数据元素都插进去，这能不能实现?

大家想一下，你能把10升水装到5升的瓶子中吗？肯定不行！如果你非要把10升水都装到瓶子里，肯定需要换一个新的更大的瓶子！

所以今天**壹哥**跟大家说的”数组扩容“，其实并不是将这些多余的数据装到原有的数组中，而是创建一个新的更大的数组，再把原有数组中的内容都复制到新数组中来！

2. 扩容与缩容流程(重点)

在Java中，数组的”扩容“和”缩容“，并不是真的改变原有数组的大小，而是创建一个新的数组，然后再进行操作，具体流程如下：

> - 步骤1：**定义一个新数组，新数组的长度要比原数组增加或者减小；**
> - 步骤2：**将原数组中的元素拷贝到新数组中；**
> - 步骤3：**将原数组的名称变量指向新数组。**

3. 代码实现

接下来**壹哥**就按照上面的流程，来带大家实现一下数组的扩容和缩容。

3.1 扩容代码

以下代码是进行数组扩容的案例。

```java
/**
 * @author ——哥Sun
 * QQ：2312119590
 * CSDN、掘金、知乎找我哦
 */
public class Demo05 {

    public static void main(String[] args) {
        // 数组扩容

        // 原数组
        int[] oldArr = { 1, 3, 46, 22, 11 };

        // 1.定义一个新数组，长度比原数组的长度多1，用于扩容
        int[] newArr = new int[oldArr.length + 1];

        // 2.数组拷贝
        for (int i = 0; i < oldArr.length; i++) {
            //数组拷贝，将原来数组的元素拷贝到新数组中
            newArr[i] = oldArr[i];
        }

        // 3.将原数组的名称变量指向新数组
        oldArr = newArr;

        System.out.println("数组长度="+oldArr.length);
        //遍历数组
        for (int i = 0; i < oldArr.length; i++) {
            //最后一个元素的值是默认值0
            System.out.println(oldArr[i]);
        }
    }

}
```

这里我们使用`newArr[i] = oldArr[i];这样的语句`，将旧数组中的元素拷贝到新数组中

3.2 缩容代码

以下代码是进行数组缩容的案例，供大家参考：

```java
/**
 * @author ——哥Sun
 * QQ：2312119590
 * CSDN、掘金、知乎找我哦
 */
public class Demo06 {

    public static void main(String[] args) {
        // 数组缩容

        //定义一个原数组
        int[] oldArr = {1,3,46,22,11};

        //1.定义一个新数组，新数组的长度比原数组长度少1个
        int[] newArr = new int[oldArr.length-1];

        //2.进行数组拷贝，将旧数组中的元素拷贝到新数组中
        for (int i = 0; i < newArr.length; i++) {
            newArr[i] = oldArr[i];
        }

        //3.将原数组的名称变量指向新数组
        oldArr = newArr;

        for (int i = 0; i < newArr.length; i++) {
            System.out.println(oldArr[i]);
        }
    }

}
```

三. 数组拷贝

壹哥在给大家讲解数组扩容时，涉及到了数组中数据元素的拷贝复制。那么除了上面的拷贝方式之外，数组还有哪些拷贝方式呢?

1. 拷贝方式

在Java中，数组的拷贝主要有三种实现方式：

> **1. 通过循环语句，将原数组中的各个元素拷贝到新数组中(即数组扩容案例中使用的方法)；**

2. System类提供的数组拷贝方法；
3. Arrays类提供的数组拷贝方法。

接下来壹哥就设计几个案例，来给大家展示这几种方式都是怎么进行数组拷贝的。因为第一种数组拷贝方式，我们已经在数组扩容的案例中给大家演示了，这里就不再重复展示相关代码了。

2. System.arraycopy方法

2.1 简介

System.arraycopy()是Java提供的一个本地静态方法，用于将数据元素从源数组复制到目标数组。

```java
public static native void arraycopy(Object src,int srcPos,Object dest,int destPos,int length);
```

arraycopy()方法有5个核心参数，其含义如下：

- src: 源数组，即被复制的旧数组；
- srcPos: 源数组中开始复制的索引位置；
- dest: 目标数组，即要复制到的新数组；
- destPos: 要复制到目标数组中的索引位置；
- length: 要复制的元素个数。

另外我们还要注意，arraycopy()方法在有些情况下有可能会发生如下异常：

- NullPointerException: if source or destination array is null.NullPointerException ： 如果源或目标数组为null时，就会产生NullPointerException异常。
- ArrayStoreException: if the source and destination array type doesn't match or they are not array.ArrayStoreException ： 如果源和目标数组类型不匹配或不是数组，会产生该异常；
- ArrayIndexOutOfBoundsException: if the data overflow occurs because of index values or they are negative.ArrayIndexOutOfBoundsException ： 如果由于索引值导致数据溢出，或它们为负数时会产生该异常。

2.2 案例

我们先来看看下面这个数组拷贝的案例：

Java | 复制代码

```java
/**
 * @author ——哥Sun
 * QQ：2312119590
 * CSDN、掘金、知乎找我哦
 */
public class Demo07 {

    public static void main(String[] args) {
        // 数组拷贝

        //1.源数组
        int[] srcArr = {1,3,46,22,11};

        //2.目标数组
        int[] destArr = new int[srcArr.length + 5];

        /**
        * src:原数组
        * srcPos：原数组的起始拷贝位置
        * dest：目标数组
        * destPos：目标数组的起始拷贝位置
        * length：拷贝的长度
        */
        //3.调用arraycopy方法进行复制
        System.arraycopy(srcArr, 1, destArr, 3, 4);

        //对新数组进行遍历
        for (int i = 0; i < destArr.length; i++) {
            System.out.print(destArr[i]+"\t");
        }
    }

}
```

3. Arrays.copyOf方法

3.1 简介

Arrays.copyOf()可以复制数组中指定范围的元素。该方法会返回一个新的数组对象，且改变新数组中的元素值，不会影响原来的数组。我们还可以利用Arrays.toString方法将赋值后的数组输出。

该方法支持的参数可以是long、float、double、int、boolean、byte、Object等类型的数组。

Java | 复制代码

```java
public static int[] copyOf(int[] original, int newLength);
```

copyOf()方法有2个核心参数，其含义如下：

> - **original**: 源数组，即被复制的旧数组；
> - **newLength**: 表示新数组的长度。如果新数组的长度超过源数组的长度，会采用数组元素类型的默认值。

3.2 案例

以下是Arrays.copyOf()方法的实现案例：

```java
/**
 * @author ——哥Sun
 * QQ：2312119590
 * CSDN、掘金、知乎找我哦
 */
public class Demo08 {

    public static void main(String[] args) {
        // 数组拷贝

        //1.源数组
        int[] srcArr = {1,3,46,22,11};

        /**
         * original：原数组
         * newLength：新数组的长度
         * 返回值：返回新数组
         */
        //2.调用copyOf方法进行数组拷贝
        int[] destArr = Arrays.copyOf(srcArr, srcArr.length+1);

        //3.遍历新数组
        for (int i = 0; i < srcArr.length; i++) {
            System.out.print(destArr[i]+"\t");
        }
    }

}
```

----------正片已结束，来根事后烟----------

四. 结语

至此，**壹哥**就把数组的扩容、缩容及拷贝等内容给大家介绍完毕了，现在你明白数组的扩容原理了吗？在下一篇文章中，**壹哥**会继续给大家讲解数组的排序、查找等算法，内容非常重要哦，敬请继续关注吧。

另外如果你独自学习觉得有很多困难，可以加入**壹哥**的学习互助群，大家一起交流学习。

五. 今日作业

1. 第一题

将班级中5个学员的名称录入到数组中，再拷贝到更大的一个数组中输出。