数组的定义与使用

> ### 本节目标
>
> 1. 数组基本概念
> 2. 二维数组
> 3. 数组与方法互操作
> 4. Java对数组的支持
> 5. 数组案例
> 6. 对象数组

数组指的就是一组相关类型的变量集合，并且这些变量可以按照统一的方式进行操作。

1. 基本概念

1.1 动态初始化

数组是引用数据类型，有内存分配问题。

> - 数组动态初始化（声明并开辟数组）：

```
数据类型[] 数组名称 = new 数据类型 [长度];
```

当数组开辟空间之后，就可以采用如下方式进行操作：

> 1. 数组的访问通过索引完成。即：“数组名称[索引]”，注意：数组索引从0开始，因此可以采用的索引范围就是0~索引-1；假设现在开辟了3个空间的数组，那么可以使用的索引是：0、1、2。如果访问超过索引访问，那么会产生"java.lang.ArrayIndexOutOfBoundsException"异常信息。
> 2. 当数组采用动态初始化开辟空间之后，数组之中的每个元素都是该数据类型的默认值；
> 3. 数组本身是一个有序的集合操作，所以对于数组的内容操作往往采用循环的模式完成。（数组是一个有限的集合，采用for循环）
> 4. 在Java中有一种动态取得数组长度的方法：数组名称.length;

范例：定义一个int型数组

```
public class ArrayDemo{
    public static void main(String[] args) {
        int[] x = new int[3] ; // 开辟了一个长度为3的数组
        System.out.println(x.length) ;
        x[0] = 1 ; // 数组第一个元素
        x[1] = 2 ; // 数组第二个元素
        x[2] = 3 ; // 数组第三个元素
        for (int i = 0; i<x.length ; i++) {
            System.out.println(x[i]) ; // 通过循环控制索引下标更改
        }
    }
}
```

数组本身除了声明并开辟空间之外还有另外一种开辟模式。

范例：分步进行数组实例化

```
int[] x = null ;
x = new int[3] ;
```

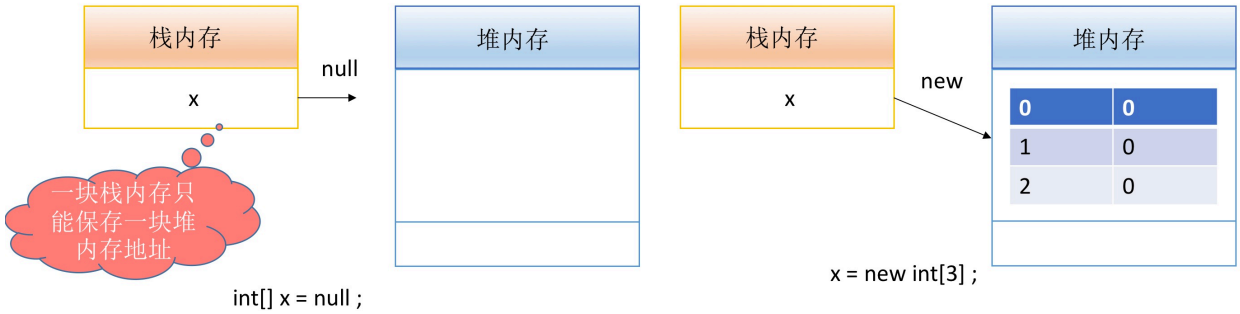
数组属于引用数据类型，因此在使用之前一定要开辟空间（实例化），否则就会产生NullPoninterException

1.2 引用传递

数组作为引用数据类型，也一定可以发生引用传递。在这之前，我们先研究一下数组的空间开辟。

范例：观察一个简单程序

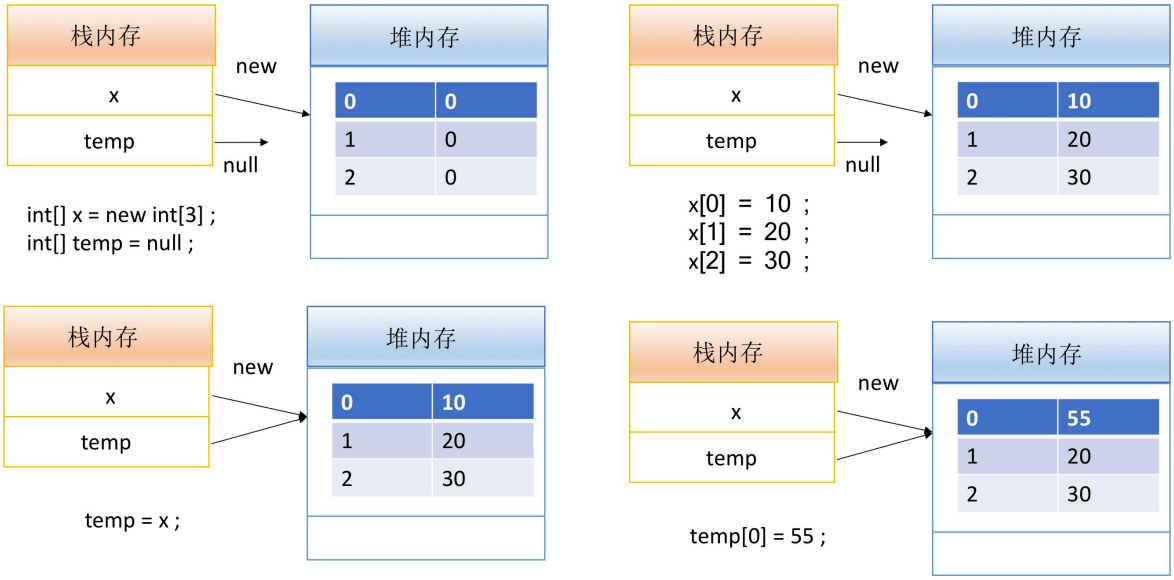
```
public class ArrayDemo{
    public static void main(String[] args) {
        int[] x = null ;
        x = new int[3] ;
        System.out.println(x.length) ;
        x[0] = 10 ; // 数组第一个元素
        x[1] = 20 ; // 数组第二个元素
        x[2] = 30 ; // 数组第三个元素
        for (int i = 0; i<x.length ; i++) {
            System.out.println(x[i]) ; // 通过循环控制索引下标更改
        }
    }
}
```



引用传递空间：同一块堆内存空间可以被不同的栈内存所指向。

范例：多个栈内存指向相同的堆内存

```
public class ArrayDemo{
    public static void main(String[] args) {
        int[] x = null ;
        int[] temp = null ; // 声明对象
        x = new int[3] ;
        System.out.println(x.length) ;
        x[0] = 1 ; // 数组第一个元素
        x[1] = 2 ; // 数组第二个元素
        x[2] = 3 ; // 数组第三个元素
        for (int i = 0; i<x.length ; i++) {
            System.out.println(x[i]) ; // 通过循环控制索引下标更改
        }
        temp = x ; //如果要发生引用传递，不要出现[]
        temp[0] = 55 ; // 修改数据
        System.out.println(x[0]) ;
    }
}
```



引用传递的内存分析都是类似的：同一块堆被不同的栈内存指向。

1.3 数组静态初始化

在之前的数组定义都有一个明显特点：数组首先先开辟内存空间，而后再使用索引进行内容的设置，这种定义数组的方式称为动态初始化；而如果希望数组在定义的同时可以设置内容，那么就可以采用静态初始化。

数组的静态初始化语法一共分为以下两种(推荐使用第二种完整格式)：

简化格式	完整格式
数据类型[] 数组名称 = {值，值，....}	数据类型[] 数组名称 = new 数据类型[] {值，值，....}

范例：采用静态初始化定义数组

```
public class ArrayDemo{
    public static void main(String[] args) {
        int[] x = {1,2,5,55,555,223,45545,666465,6443} ; // 静态初始化定义数组
        System.out.println(x.length) ;
        for (int i = 0; i<x.length ; i++) {
            System.out.println(x[i]) ;
        }
    }
}
```

在开发之中，对于静态数组初始化强烈推荐完整格式，这样可以轻松使用匿名数组这一概念。

范例：观察匿名数组

```
public class ArrayDemo{
    public static void main(String[] args) {
        System.out.println(new int[]
{1,2,5,55,555,223,45545,666465,6443}.length) ; // 匿名数组
    }
}
```

以后使用静态方式定义数组的时候，请使用完整格式

数组最大的缺陷：长度固定（存在越界问题）

2. 二维数组（了解）

在之前所使用的数组，使用一个索引就可以访问。这样的数组像一个数据行的概念。

索引	0	1	2	3	4	5
内容	23	1	2	3	4	5

现在通过一个索引就可以取得唯一的一个记录，这样的数组称为一维数组。

而二维数组本质上指的就是一个行列的集合。换言之，如果要获得一个数据，既需要行索引，也需要列索引。

索引 （上行，下列）	1	2	3	4
0	23	1	2	3
1	5	6	7	8

在上述的结构中，如果要确定一个数据，则使用结构为`数组名称[行索引][列索引]`这样的结构就是一个表的结构。

那么对于二维数组的定义有两种声明格式：

- 动态初始化：

```
数据类型[][] 对象数组 = new 数据类型[行个数][列个数] ;
```

- 静态初始化：

```
数据类型[][] 对象数组 = new 数据类型[][]{{值，值，..},{值，值，..},..} ;
```

数组的数组就是二维数组

范例：定义一个二维数组

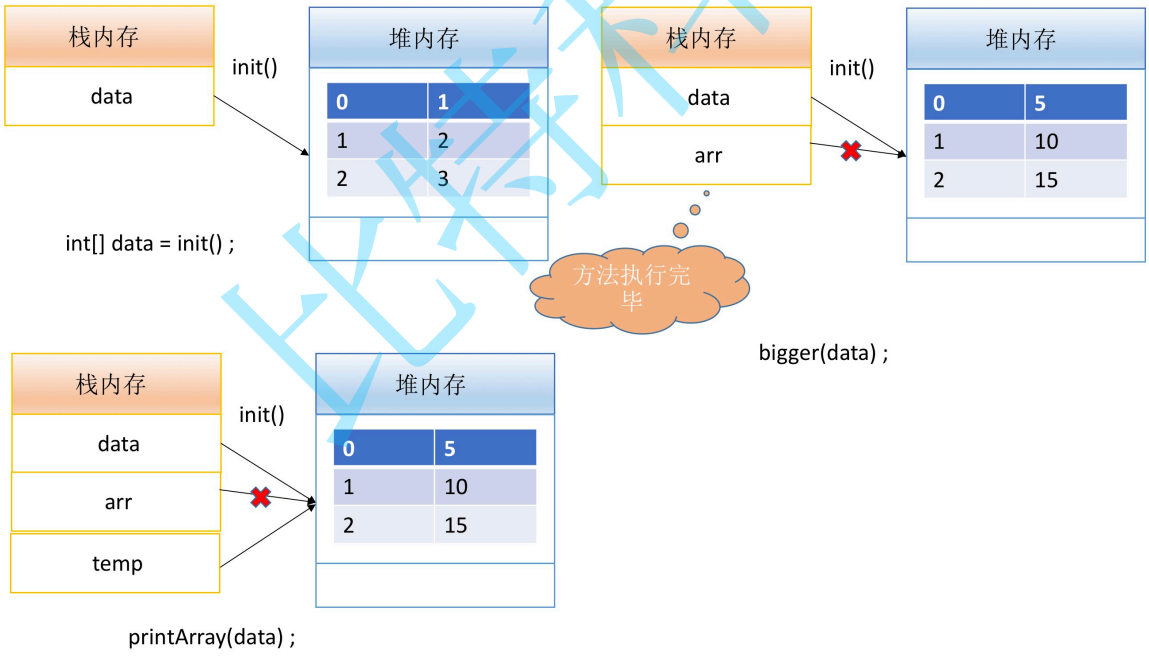
```
public class ArrayDemo{
```

```
        public static void main(String[] args) {
                // 数组并不是等列数组
                int[][] data = new int[][] {
                        {1,2,3},{4,5},{6,7,8,9}
                };
                // 在进行输出的时候一定要使用双重循环
                for (int x = 0; x<data.length ; x++) {
                        for (int y = 0; y<data[x].length ; y ++) {
                                System.out.println("data{"+x+"]
["+y+"]="+data[x][y]+"、") ;
                        }
                        System.out.println();
                }
        }
}
```

开发之中，出现二维数组的概率非常低，了解概念即可。主要用于对付笔试题和数据结构。

3. 数组与方法互操作（重点）

数组是引用数据类型，所有引用数据类型都可以为其设置多个栈内存指向。所以在进行数组操作的时候，也可以将其通过方法进行处理。



3.1 方法接收数组

```
public class ArrayDemo{
        public static void main(String[] args) {
                int[] data = new int[] {1,2,3,4,5} ;
                printArray(data) ; // 其效果等价于 int[] temp = data ;
        }
        public static void printArray(int[] temp) {
                for (int i = 0 ; i<temp.length ; i++) {
                        System.out.println(temp[i]) ;
                }
        }
}
```

3.2 方法返回数组

```
public class ArrayDemo{
        public static void main(String[] args) {
                int[] data = init() ;
                printArray(data) ;
        }
        // 定义一个返回数组的方法
        public static int[] init(){
                return new int[] {1,2,3,4,5} ; // 匿名数组
        }
        public static void printArray(int[] temp) {
                for (int i = 0 ; i<temp.length ; i++) {
                        System.out.println(temp[i]) ;
                }
        }
}
```

现在的数组上发生了引用传递，那么就意味着方法在接收数组后也可以修改数组。

3.3 方法修改数组

```
public class ArrayDemo{
        public static void main(String[] args) {
                int[] data = init() ;
                bigger(data) ;
                printArray(data) ;
        }
        // 定义一个返回数组的方法
        public static int[] init(){
                return new int[] {1,2,3,4,5} ; // 匿名数组
        }
        // 将数组中每个元素的值扩大5倍
        public static void bigger(int[] arr){
                for (int i =0 ; i<arr.length ; i++) {
```

```
                        arr[i]*=5 ; // 每个元素扩大5倍
                }
        }
        public static void printArray(int[] temp) {
                for (int i = 0 ; i<temp.length ; i++) {
                        System.out.println(temp[i]) ;
                }
        }
}
```

4. Java对数组的支持

在JavaSE的类库中提供有对数组操作的支持。

4.1 实现数组排序

Java类库中数组排序操作如：`java.util.Arrays.sort(arrayName);`

```
public class ArrayDemo{
        public static void main(String[] args) {
                int[] intData = new int[]{1,65,55,23,100} ;
                char[] charData = new char[]{'z','a','c','b'} ;
                java.util.Arrays.sort(intData) ;
                java.util.Arrays.sort(charData) ;
                printArray(intData) ;
                printArray(charData) ;
        }
        public static void printArray(int[] temp) {
                for (int i = 0 ; i<temp.length ; i++) {
                        System.out.println(temp[i]) ;
                }
                System.out.println() ;
        }
        // 重载
        public static void printArray(char[] temp) {
                for (int i = 0 ; i<temp.length ; i++) {
                        System.out.println(temp[i]) ;
                }
                System.out.println() ;
        }
}
```

只要是*基本数据类型*的数组，sort方法都可以进行排序处理(升序处理)。**内部使用双轴快速排序**。

4.2 实现数组拷贝

数组部分拷贝：指的是将一个数组的部分内容替换掉另一个数组的部分内容（必须是连续的）

- 数组拷贝：

```
System.arraycopy(源数组名称，源数组开始点，目标数组名称，目标数组开始点，拷贝长度);
```

目标数组A：`1、2、3、4、5、6、7、8、9` 源数组B：`11、22、33、44、55、66、77、88、99` 替换后：`1、55、66、77、5、6、7、8、9`

```
public class ArrayDemo{
        public static void main(String[] args) {
                int[] dataA = new int[]{1,2,3,4,5,6,7,8,9} ;
                int[] dataB = new int[]{11,22,33,44,55,66,77,88,99} ;
                System.arraycopy(dataB,4,dataA,1,3) ;
                printArray(dataA) ;
        }
        public static void printArray(int[] temp) {
                for (int i = 0 ; i<temp.length ; i++) {
                        System.out.println(temp[i]) ;
                }
                System.out.println() ;
        }
}
```

- 数组拷贝:`java.util.Arrays.copyOf(源数组名称,新数组长度)`

范例:使用数组拷贝

```
import java.util.Arrays;
public class Test {
    public static void main(String[] args) {
        int[] original = new int[]{1,3,5,7,9};
        int[] result = Arrays.copyOf(original,10);
        for (int temp : result) {
            System.out.println(temp);
        }
    }
}
```

Java类集框架(动态数组)就采用的此方法来动态扩容。

5. 数组案例

5.1 数据统计

要求：给你一个数组，要求可以统计出该数组的最大值、最小值、平均值、综合。

0	1	2	3	4	5	6	7	8
1	4	3	4	55	77	6	9	8

范例：观察简单实现

```
public class Test {
    public static void main(String[] args) {
       int[] data = new int[]{1,4,3,4,55,77,6,9,8} ;
       int max = data[0] ; // 假定第一个元素为最大值
       int min = data[0] ; // 假定第一个元素为最小值
       int sum = data[0] ;
       for (int i = 1 ; i< data.length; i++){
           sum += data[i] ;
           if (data[i]>max){
               max = data[i] ;
           }
           if (data[i]<min){
               min = data[i] ;
           }
       }
       System.out.println("最大值为:" +max);
       System.out.println("最小值为:"+min);
       System.out.println("总和为：+sum);
       System.out.println("平均值为："+(double)sum/data.length);
    }
}
```

以上代码有一个最大的问题：主方法代码量太多。主方法相当于客户端调用，里面的代码应该越简单越好。

范例：通过方法与数组互操作简化主方法代码

```
public class Test {
    public static void main(String[] args) {
       processData(new int[]{1,4,3,4,55,77,6,9,8});
    }
    public static void processData(int[] temp){
        double[] result = new double[4] ;
        result[0] = temp[0] ; // result[0] means the max of the array
        result[1] = temp[0] ; // result[1] means the min of the array
        result[2] = temp[0] ; // result[2] means the sum of the array
        result[3] = temp[0] ; // result[3] means the avg of tha array
        for (int i =0 ; i < temp.length;i++){
            result[2] += temp[i] ;
            if (temp[i]>result[0]){
                result[0] = temp[i] ;
            }
            if (temp[i]<result[1]){
                result[1] = temp[i] ;
            }
        }
        result[3] = result[2]/temp.length ;
```

```
        System.out.println("最大值为:" +result[0]);
        System.out.println("最小值为:"+result[1]);
        System.out.println("总和为："+result[2]);
        System.out.println("平均值为："+result[3]);
    }
}
```

6. 对象数组

之前所定义的数组都属于基本类型的数组，对象数组往往是以引用数据类型为主的定义，例如：类、接口。

6.1 动态初始化

语法：

```
类名称[] 对象数组名称 = new 类名称[长度] ;
```

范例：观察对象数组的动态初始化

```
class Person {
        private String name ;
        private int age ;
        public Person(String name, int age) {
                this.name = name ;
                this.age = age ;
        }
        public void getInfo() {
                System.out.println("姓名：+this.name+",年龄：+this.age) ;
        }

}
public class Test {
        public static void main(String[] args){
                Person[] per = new Person[3] ; // 数组动态初始化，每个元素都是其对应数据类型的默认值
                per[0] = new Person("张三",1) ;
                per[1] = new Person("李四",2) ;
                per[2] = new Person("王五",3) ;
                for (int x = 0 ; x < per.length  ; x++) {
                        per[x].getInfo() ;
                }
        }
}
```

6.2 静态初始化

范例：观察对象数组静态初始化

```
class Person {
        private String name ;
        private int age ;
        public Person(String name, int age) {
                this.name = name ;
                this.age = age ;
        }
        public void getInfo() {
                System.out.println("姓名：" +this.name+",年龄：" +this.age) ;
        }

}
public class Test {
        public static void main(String[] args){
                Person[] per = new Person[] {
                        new Person("张三",1) ,
                        new Person("李四",2) ,
                        new Person("王五",3)
                } ; // 对象数组静态初始化
                for (int x = 0 ; x < per.length  ; x++) {
                        per[x].getInfo() ;
                }
        }
}
```

对象数组保存的内容要比普通数据类型要多，以后开发之中用到的很多。