【济南中心】JAVA 编程阶梯：基础篇之第五章

- **数组概述**

概念：

数组是存储同一种数据类型多个元素的集合。也可以看成是一个容器。

数组既可以存储基本数据类型，也可以存储引用数据类型。

应用场景：为了存储同种数据类型的多个值

- **数组定义格式**

格式 1：元素类型[] 数组名 = new 元素类型[元素个数或数组长度];

例：int[] arr = new int[8];

格式 2：元素类型[] 数组名 = new 元素类型[]{元素，元素，……};

例：int []arr = new int[]{3,5,1,7};int []arr ={3,5,7,1};

- **数组的初始化**

Java 中的数组必须先初始化,然后才能使用。

所谓初始化：就是为数组中的数组元素分配内存空间，并为每个数组元素赋值。

数组的初始化分为动态初始化和静态初始化

动态初始化：初始化时只指定数组长度，由系统为数组分配初始值。

格式：数据类型[] 数组名 = new 数据类型[数组的长度];

举例：

int[] arr = newint[3];

解释 :定义了一个 int 类型的数组 ,这个数组中可以存放 3 个 int 类型的值。

静态初始化：初始化时指定每个数组元素的初始值，由系统决定数组长度。

数组的初始化静态初始化及内存图

格式：数据类型[] 数组名 = new 数据类型[]{元素 1,元素 2,...};

简化格式：数据类型[] 数组名 = {元素 1,元素 2,...};

举例：

int[] arr = new int[]{3,7,5,4};

int[] arr = {3,7,5,4};

- **Java 中的内存分配以及栈和堆的区别**

需要在内存中的分配空间。为了提高运算效率 ,就对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据方式和内存管理方式。

内存结构：栈、堆、方法区、本地方法区、寄存器。（java 对内存的划分原因：每片内存处理的方式不同）

栈内存：暂时存储。用于存储局部变量，当数据使用完，所占空间会自动释放。

堆内存：存储数组和对象，通过 new 建立的实例都存放在堆内存中，任何的"引用数据类型"的"值"都存在堆里。。

方法区：静态存储区、构造函数、常量池、线程池，方法字节码

本地方法区：window 系统占用 ，被定义为 native 的方法。

寄存器：存储正要准备交给 CPU 处理的字节码

数组操作的两个常见小问题越界和空指针

ArrayIndexOutOfBoundsException:数组索引越界异常

原因：你访问了不存在的索引。

NullPointerException:空指针异常

原因：数组已经不在指向堆内存了。而你还用数组名去访问元素。

数组的操作：遍历

```
/**
 * 数组遍历：依次输出数组中的每一个元素。 数组的属性:arr.length 数组的长度 数组的最大索引:arr.length - 1;
 */
class Demo {
		public static void main(String[] args) {
			int[] arr = new int[] { 4, 8, 68, 42, 2, 7 };
			print(arr);
		}

		public static void print(int[] arr) {
			for (int i = 0; i < arr.length; i++) {
				System.out.print(arr[i] + " ");
			}
		}

}
```

数组的操作：获取最值

```
/**
 * 数组获取最值(获取数组中的最大值最小值)
 */
class Demo {
		public static void main(String[] args) {
			int[] arr = new int[] { 4, 8, 68, 42, 2, 7 };
			int max = getMax(arr);
			System.out.println(max);
		}

		public static int getMax(int[] arr) {
			int max = arr[0];
			for (int i = 1; i < arr.length; i++) { // 从数组的第二个元素开始遍历
				if (max < arr[i]) { // 如果 max 记录的值小于的数组中的元素
					max = arr[i]; // max 记录住较大的
```

```
                                        }
                                }

                                return max;
                        }

}
```

数组的操作：反转

```
/**
 * 数组遍历：依次输出数组中的每一个元素。 数组的属性:arr.length 数组的长度 数组的最大索引:arr.length - 1;
 */
class Demo {
                public static void main(String[] args) {
                                int[] arr = new int[]{4,8,68,42,2,7};
                                reverseArray(arr);
                                for (int i = 0; i < arr.length; i++) {
                                                System.out.println(arr[i]);
                                }
                }

                public static void reverseArray(int[] arr) {
                                for (int i = 0; i < arr.length / 2; i++) {
int temp = arr[i];
                                                arr[i] = arr[arr.length - 1 - i];
                                                arr[arr.length - 1 - i] = temp;
                                }
                }

}
```

数组的操作：选择查找

```
public class SelectDemo {
                public static void main(String[] args) {
                                int[] arr = { 4 , 9, 32, 12, 6, 12 };
                                printArray(arr);
                                selectSort(arr);
                }
```

```
		// 打印数组的方法。
		public static void printArray(int[] arr) {
			for (int x = 0; x < arr.length; x++) {
				if (x != arr.length - 1)
					System.out.prin
t(arr[x] + ",");
				else
					System.out.prin
tln(arr[x]);
			}
		}
		/*
		 * 数组的排序。 选择排序。
		 */
		public static void selectSort(int[] arr) {
			for (int x = 0; x < arr.length - 1; x++) {
				for (int y = x + 1; y < arr.length;
y++) {
					if (arr[x] >
arr[y]) {
	// int temp = arr[x];
	// arr[x] = arr[y];
	// arr[y] = temp;
	swap(arr, x, y);
					}
				}
			}
		}
}
```

数组的操作：冒泡排序

```
public class BubbleDemo {
		public static void main(String[] args) {
			int[] arr = { 12, 9, 32, 23, 6, 34 };
			printArray(arr);
			bubbleSort(arr);
```

```
                printArray(arr);
        }

        // 打印数组的方法。
        public static void printArray(int[] arr) {
                for (int x = 0; x < arr.length; x++) {
                        if (x != arr.length - 1)
                                System.out.prin
t(arr[x] + ",");
                        else
                                System.out.prin
tln(arr[x]);
                }
// 发现排序方法，位置置换代码重复，进行抽取。
public static void swap(int[] arr, int a, int b) {
int temp = arr[a];
arr[a] = arr[b];[/b]
arr[b] = [/b]temp;
}
        }

        /*
          * 冒泡排序。
          */
        public static void bubbleSort(int[] arr) {
                for (int x = 0; x < arr.length - 1; x++) {
                        for (int y = 0; y < arr.length -
1 - x; y++) {
                                if (arr[y] >
arr[y + 1]) {

  // int temp = arr[y];

  // arr[y] = arr[y+1];

  // arr[y+1] = temp;

  swap(arr, y, y + 1);
                                }
                        }
                }

        // 发现排序方法，位置置换代码重复，进行抽取。
```

```
	public static void swap(int[] arr, int a, int b) {
		int temp = arr[a];
		arr[a] = arr;
		arr = temp;
	}
}
```

数组的操作：二分查找

```
class Demo {

	public static void main(String[] args) {
		int[] arr = { 19, 32, 123, 1,23,34, 12 };

		int index = binarySearch(arr, 23);
		System.out.println("index=" + index);

	}

	// 二分查找。前提：数组必须是有序的。
	/*
	 * 思路： 1，通过角标先获取中间角标上元素。 2，让该元素和要找的数据比较。 3，如果要找的数大了，缩小范围，要找的范围应该是
	 * 中间的角标+1---尾角标。 如果要找的数小了，要找的范围 头角标---中间角标-1； 4，不断如此重复，就可以找到元素对应的角标。
	 */

	public static int binarySearch(int[] arr, int key) {
		// 定义三个变量，记录头角标，尾角标，中间角标。
		int max, min, mid;
		min = 0;
		max = arr.length - 1;

		while (min <= max) {
			mid = (min + max) >> 1;

			if (key > arr[mid])
				min = mid + 1;
			else if (key < arr[mid])
				max = mid - 1;
			else
				return mid;
		}
		return -1;
	}
```

```
}
```

数组的操作：查表法

```
/**
  * 数组查表法(根据键盘录入索引,查找对应星期)
  */
class Demo {
		public static void main(String[] args) {
				char week = getWeek(2);
				System.out.println("星期"+week);
		}

		public static char getWeek(int week) {
				char[] arr = { ' ', '一', '二', '三', '四', '五',
'六', '日' }; // 定义了一张星期表
				return arr[week]; // 通过索引获取表中的元素
		}

}
```

数组的操作：查找

```
/**
  * 数组元素查找(查找指定元素第一次在数组中出现的索引)
  */
class Demo {
		public static void main(String[] args) {
				int[] arr = new int[] { 4, 8, 68, 42, 32, 8,2, 7 };
				int index = getIndex(arr, 2);
				System.out.println(index);
		}

		public static int getIndex(int[] arr, int value) {
				for (int i = 0; i < arr.length; i++) { // 数组的
遍历
						if (arr[i] == value) { // 如果数
组中的元素与查找的元素匹配
								return i;
						}
				}
				return -1;
		}

}
```

- **二维数组概述和格式的讲解**

二维数组概述

二维数组格式

二维数组格式 1 的解释

注意事项

a:以下格式也可以表示二维数组

1:数据类型 数组名[][] = new 数据类型[m][n];

2:数据类型[] 数组名[] = new 数据类型[m][n];

举例：

```
int[][] arr = new int[3][2];
System.out.println(arr);          [[I@e6f7d2    哈希值，二维数组实体。
System.out.println(arr[0]);    [I@3e0ebb      一维数组实体。
System.out.println(arr[0][0]);//0 一维数组中的元素。
```

解释：

- 定义了名称为 arr 的二维数组
- 二维数组中有 3 个一维数组
- 每一个一维数组中有 2 个元素
- 一维数组的名称分别为 arr[0], arr[1], arr[2]
- 给第一个一维数组 1 脚标位赋值为 78 写法是：arr[0][1] = 78；

```
int[][] array = new int[3][];//明确了二维数组的长度，没有明确具体的一维数组。
System.out.println(array);//[[I@3e0ebb
System.out.println(array[0]);//null
System.out.println(array[0][0]);//NullPointerException
```

解释：

- 二维数组中有 3 个一维数组
- 每个一维数组都是默认初始化值 null

- 可以对这个三个一维数组分别进行初始化

  arr[0] = new int[3];

  arr[1] = new int[1];

  arr[2] = new int[2];

遍历二维数组

```
for (int x = 0; x < arr.length; x++) {
                for (int y = 0; y < arr[x].length; y++) {
                                sum += arr[x][y];
                }
}
```