数组概述

关于数组我们可以把它看作是一个类型的所有数据的一个集合，并用一个数组下标来区分或指定每一个数，例如一个足球队通常会有几十个人，但是我们来认识他们的时候首先会把他们看作是某某对的成员，然后再利用他们的号码来区分每一个队员，这时候，球队就是一个数组，而号码就是数组的下标，当我们指明是几号队员的时候就找到了这个队员。 同样在编程中，如果我们有一组相同数据类型的数据，例如有10个数字，这时候如果我们要用变量来存放它们的话，就要分别使用10个变量，而且要记住这10个变量的名字，这会十分的麻烦，这时候我们就可以用一个数组变量来存放他们，例如在VB中我们就可以使用dim a(9) as integer（注意：数组的下标是从0开始的，所以10个数的话，下标就是9,a(0)=1）。 使用数组会让程序变的简单，而且避免了定义多个变量的麻烦。

数组的定义：

- 数组是相同类型数据的有序集合.
- 数组描述的是相同类型的若干个数据,按照一定的先后次序排列组合而成。
- 其中,每一个数据称作一个数组元素,每个数组元素可以通过一个下标来访问它们.

数组的四个基本特点：

1. 其长度是确定的。数组一旦被创建，它的大小就是不可以改变的。
2. 其元素必须是相同类型,不允许出现混合类型。
3. 数组中的元素可以是任何数据类型，包括基本类型和引用类型。
4. 数组变量属引用类型，数组也可以看成是对象，数组中的每个元素相当于该对象的成员变量。数组本身就是对象，Java中对象是在堆中的，因此数组无论保存原始类型还是其他对象类型，**数组对象本身是在堆中的。**

数组声明创建

1、声明数组

首先必须声明数组变量，才能在程序中使用数组。下面是声明数组变量的语法：

```
dataType[] arrayRefVar;   // 首选的方法
或
dataType arrayRefVar[];  // 效果相同，但不是首选方法
```

建议使用 **dataType[] arrayRefVar** 的声明风格声明数组变量。 dataType arrayRefVar[] 风格是来自 C/C++ 语言 ， 在Java中采用是为了让 C/C++ 程序员能够快速理解java语言。

```
double[] myList;         // 首选的方法
或
double myList[];         //  效果相同，但不是首选方法
```

2、创建数组

Java语言使用new操作符来创建数组，语法如下：

```
arrayRefVar = new dataType[arraySize];
```


狂神社群笔记资料，禁止外传，本人QQ：24736743

上面的语法语句做了两件事：

- 一、使用 dataType[arraySize] 创建了一个数组。
- 二、把新创建的数组的引用赋值给变量 arrayRefVar。

数组变量的声明，和创建数组可以用一条语句完成，如下所示：

```
dataType[] arrayRefVar = new dataType[arraySize];
```

数组的元素是通过索引访问的。数组索引从 0 开始，所以索引值从 0 到 arrayRefVar.length-1。

获取数组长度：

```
arrays.length
```

【演示创建一个数组，并赋值，进行访问】

```
public static void main(String[] args) {
    //1.声明一个数组
    int[] myList = null;
    //2.创建一个数组
    myList = new int[10];
    //3.像数组中存值
    myList[0] = 1;
    myList[1] = 2;
    myList[2] = 3;
    myList[3] = 4;
    myList[4] = 5;
    myList[5] = 6;
    myList[6] = 7;
    myList[7] = 8;
    myList[8] = 9;
    myList[9] = 10;
    // 计算所有元素的总和
    double total = 0;
    for (int i = 0; i < myList.length; i++) {
        total += myList[i];
    }
    System.out.println("总和为：  " + total);
}
```

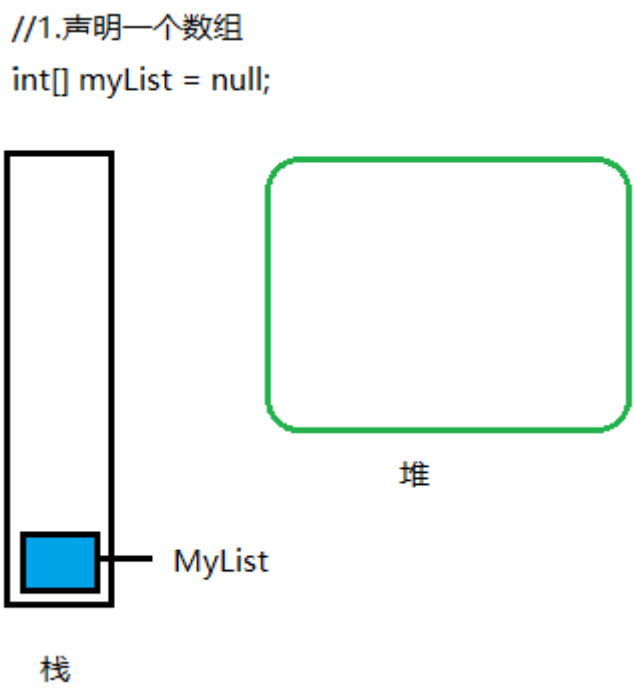
3、内存分析

Java内存分析：

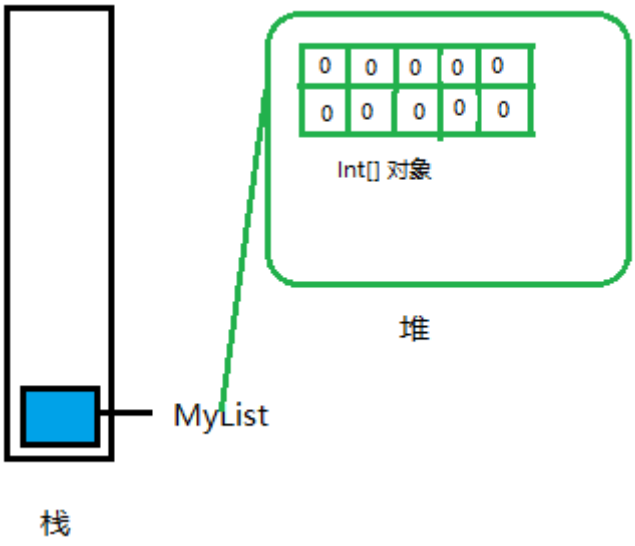
狂神社群笔记资料，禁止外传，本人QQ：24736743

1. 声明的时候并没有实例化任何对象，只有在实例化数组对象时，JVM才分配空间，这时才与长度有关。因此，声明数组时不能指定其长度(数组中元素的个数)，例如： int a[5]; //非法
2. 声明一个数组的时候并没有数组被真正的创建。
3. 构造一个数组，必须指定长度

```
1 //1.声明一个数组
2 int[] myList = null;
```

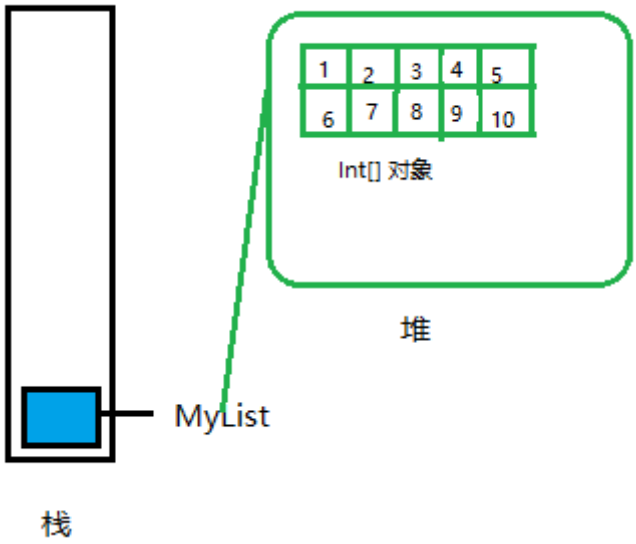



```
1 //2.创建一个数组
2 myList = new int[10];
```



狂神社群笔记资料，禁止外传，本人QQ：24736743

```
//3.像数组中存值
myList[0] = 1;
myList[1] = 2;
myList[2] = 3;
myList[3] = 4;
myList[4] = 5;
myList[5] = 6;
myList[6] = 7;
myList[7] = 8;
myList[8] = 9;
myList[9] = 10;
```



4、三种初始化

静态初始化

除了用new关键字来产生数组以外,还可以直接在定义数组的同时就为数组元素分配空间并赋值。

```
int[] a = {1,2,3};
Man[] mans = {new Man(1,1),new Man(2,2)};
```

动态初始化

数组定义、为数组元素分配空间、赋值的操作、分开进行。

```
int[] a = new int[2];
a[0]=1;
a[1]=2;
```

数组的默认初始化

数组是引用类型，它的元素相当于类的实例变量，因此数组一经分配空间，其中的每个元素也被按照实例变量同样的方式被隐式初始化。

```java
public static void main(String[] args) {
    int[] a=new int[2];
    boolean[] b = new boolean[2];
    String[] s = new String[2];
    System.out.println(a[0]+":"+a[1]); //0,0
    System.out.println(b[0]+":"+b[1]);  //false,false
    System.out.println(s[0]+":"+s[1]); //null, null
}
```

5、数组边界

下标的合法区间：[0, length-1]，如果越界就会报错；

```java
public static void main(String[] args) {
    int[] a=new int[2];
    System.out.println(a[2]);
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 2
    at com.kuang.chapter3.Demo03.main(Demo03.java:6)
```

ArrayIndexOutOfBoundsException : 数组下标越界异常！

6、小结

数组是相同数据类型(数据类型可以为任意类型)的有序集合

数组也是对象。数组元素相当于对象的成员变量(详情请见内存图)

数组长度的确定的，不可变的。如果越界，则报：ArrayIndexOutofBounds

数组使用

数组的元素类型和数组的大小都是确定的，所以当处理数组元素时候，我们通常使用基本循环或者 For-Each 循环。

【该实例完整地展示了如何创建、初始化和操纵数组】

```java
public class TestArray {
   public static void main(String[] args) {
      double[] myList = {1.9, 2.9, 3.4, 3.5};

      // 打印所有数组元素
      for (int i = 0; i < myList.length; i++) {
         System.out.println(myList[i] + " ");
      }
      // 计算所有元素的总和
      double total = 0;
      for (int i = 0; i < myList.length; i++) {
         total += myList[i];
      }
      System.out.println("Total is " + total);
      // 查找最大元素
      double max = myList[0];
      for (int i = 1; i < myList.length; i++) {
          if (myList[i] > max) {
              max = myList[i];
```

```
            }
        }
        System.out.println("Max is " + max);
    }
}
```

1、For-Each 循环

JDK 1.5 引进了一种新的循环类型，被称为 For-Each 循环或者加强型循环，它能在不使用下标的情况下遍历数组。

语法格式如下：

```
for(type element: array){
    System.out.println(element);
}
```

【示例】

```
public static void main(String[] args) {
    double[] myList = {1.9, 2.9, 3.4, 3.5};

    // 打印所有数组元素
    for (double element: myList) {
        System.out.println(element);
    }
}
```

2、数组作方法入参

数组可以作为参数传递给方法。

例如，下面的例子就是一个打印 int 数组中元素的方法：

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

3、数组作返回值

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];

  for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {
    result[j] = list[i];
  }
  return result;
}
```

以上实例中 result 数组作为函数的返回值。

多维数组

多维数组可以看成是数组的数组，比如二维数组就是一个特殊的一维数组，其每一个元素都是一个一维数组。

多维数组的动态初始化（以二维数组为例）

直接为每一维分配空间，格式如下：

```
type[][] typeName = new type[typeLength1][typeLength2];
```

type 可以为基本数据类型和复合数据类型，arraylenght1 和 arraylenght2 必须为正整数，arraylenght1 为行数，arraylenght2 为列数。

比如定义一个二维数组：

```
int a[][] = new int[2][5];
```

解析：二维数组 a 可以看成一个两行三列的数组。

多维数组的引用（以二维数组为例）

对二维数组中的每个元素，引用方式为 **arrayName[index1] [index2]**，例如：

num[1] [0];

其实二维甚至多维数组十分好理解，我们把两个或者多个值当做定位就好。

原来的数组就是一条线，我们知道一个位置就好

二维就是一个面，两点确定一个位置

三维呢，就需要三个点来确定

。。。

依次理解即可！

获取数组长度：

a.length获取的二维数组第一维数组的长度，a[0].length才是获取第二维第一个数组长度。

Arrays 类

数组的工具类java.util.Arrays

由于数组对象本身并没有什么方法可以供我们调用,但API中提供了一个工具类Arrays供我们使用,从而可以对数据对象进行一些基本的操作。

文档简介：

此类包含用来操作数组（比如排序和搜索）的各种方法。此类还包含一个允许将数组作为列表来查看的静态工厂。

除非特别注明，否则如果指定数组引用为 null，则此类中的方法都会抛出 NullPointerException。

Arrays类中的方法都是static修饰的静态方法,在使用的时候可以直接使用类名进行调用,而"不用"使用对象来调用(注意:是"不用" 而不是 "不能")

java.util.Arrays 类能方便地操作数组. 使用之前需要导包！

具有以下常用功能：

- 给数组赋值：通过 fill 方法。
- 对数组排序：通过 sort 方法,按升序。
- 比较数组：通过 equals 方法比较数组中元素值是否相等。

- 查找数组元素：通过 binarySearch 方法能对排序好的数组进行二分查找法操作。

具体说明请查看下表：

序号	方法和说明
1	**public static int binarySearch(Object[] a, Object key)** 用二分查找算法在给定数组中搜索给定值的对象(Byte,Int,double等)。数组在调用前必须排序好的。如果查找值包含在数组中，则返回搜索键的索引；否则返回 (-(*插入点*) - 1)。
2	**public static boolean equals(long[] a, long[] a2)** 如果两个指定的 long 型数组彼此*相等*，则返回 true。如果两个数组包含相同数量的元素，并且两个数组中的所有相应元素对都是相等的，则认为这两个数组是相等的。换句话说，如果两个数组以相同顺序包含相同的元素，则两个数组是相等的。同样的方法适用于所有的其他基本数据类型（Byte，short，Int等）。
3	**public static void fill(int[] a, int val)** 将指定的 int 值分配给指定 int 型数组指定范围中的每个元素。同样的方法适用于所有的其他基本数据类型（Byte，short，Int等）。
4	**public static void sort(Object[] a)** 对指定对象数组根据其元素的自然顺序进行升序排列。同样的方法适用于所有的其他基本数据类型（Byte，short，Int等）。

1、打印数组

```
public static void main(String[] args) {
    int[] a = {1,2};
    System.out.println(a);  //[I@1b6d3586
    System.out.println(Arrays.toString(a));  //[1, 2]
}
```

2、数组排序

对指定的 int 型数组按数字升序进行排序

```
public static void main(String[] args) {
    int[] a = {1,2,323,23,543,12,59};
    System.out.println(Arrays.toString(a));
    Arrays.sort(a);
    System.out.println(Arrays.toString(a));
}
```

3、二分法查找

在数组中查找指定元素并返回其下标

注意：使用二分搜索法来搜索指定的数组，以获得指定的值。必须在进行此调用之前对数组进行排序(通过sort方法等)。如果没有对数组进行排序，则结果是不确定的。

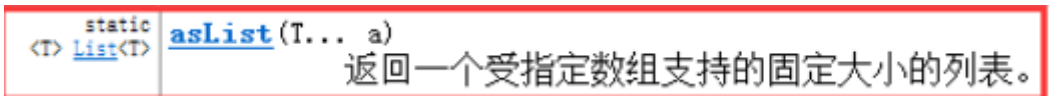
如果数组包含多个带有指定值的元素，则无法保证找到的是哪一个。

```
public static void main(String[] args) {
    int[] a = {1,2,323,23,543,12,59};
    Arrays.sort(a);    //使用二分法查找，必须先对数组进行排序
    System.out.println("该元素的索引："+Arrays.binarySearch(a, 12));
}
```

4、元素填充

```
public static void main(String[] args) {
    int[] a = {1,2,323,23,543,12,59};
    Arrays.sort(a);   //使用二分法查找，必须先对数组进行排序
    Arrays.fill(a, 2, 4, 100);  //将2到4索引的元素替换为100
    System.out.println(Arrays.toString(a));
}
```

5、数组转换为List集合




```
int[] a = {3,5,1,9,7};
List<int[]> list = Arrays.asList(a);
```

常见排序算法

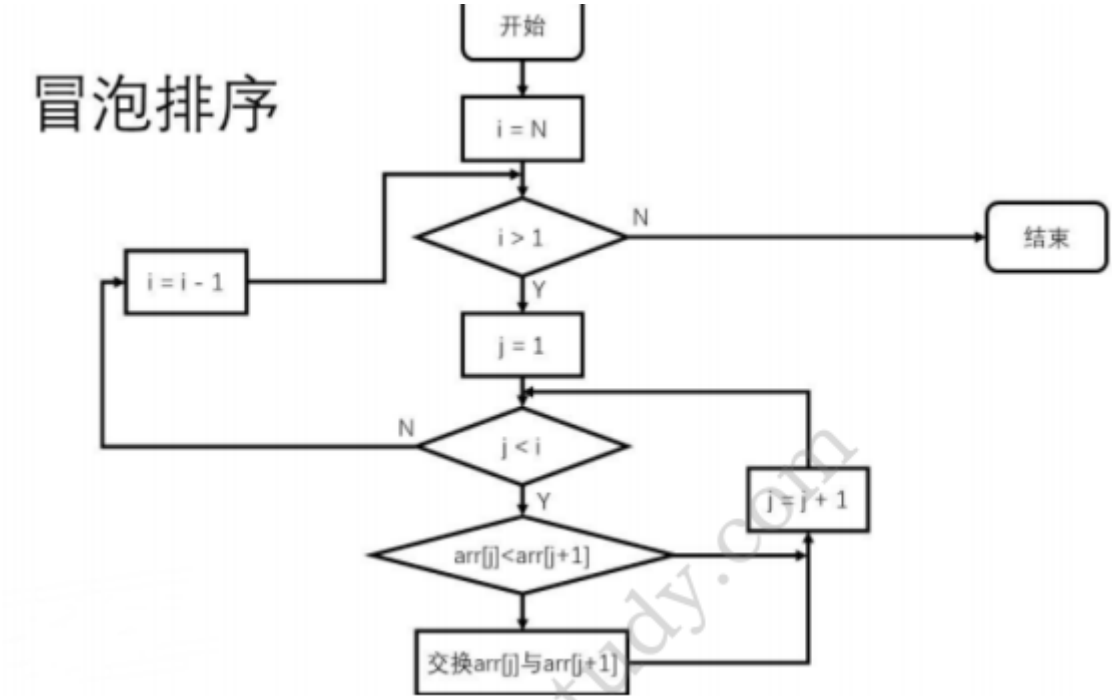
1、冒泡排序

【请写出冒泡排序代码】
冒泡排序（Bubble Sort），是一种计算机科学领域的较简单的排序算法。
它重复地走访过要排序的元素列，依次比较两个相邻的元素，如果他们的顺序（如从大到小、首字母从A到Z）错误
就把他们交换过来。走访元素的工作是重复地进行直到没有相邻元素需要交换，也就是说该元素列已经排序完成。
这个算法的名字由来是因为越大的元素会经由交换慢慢“浮”到数列的顶端（升序或降序排列），就如同碳酸饮料中二氧化碳的气泡最终会上浮到顶端一样，故名“冒泡排序”。

冒泡排序算法的原理如下：

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素做同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的
   数。
3. 针对所有的元素重复以上的步骤，除了最后一个。
4. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

狂神社群笔记资料，禁止外传，本人QQ：24736743




```java
class Bubble {
    public int[] sort(int[] array) {
        int temp = 0;
        // 外层循环，它决定一共走几趟 //-1为了防止溢出
        for (int i = 0; i < array.length - 1; i++) {
            int flag = 0; //通过符号位可以减少无谓的比较，如果已经有序了，就退出循环
            //内层循环，它决定每趟走一次
            for (int j = 0; j < array.length - i - 1; j++) {
                //如果后一个大于前一个,则换位
                if (array[j + 1] > array[j]) {
                    temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                    flag = 1;
                }
            }
            if (flag == 0) {
                break;
            }
        }
        return array;
    }

    public static void main(String[] args) {
        Bubble bubble = new Bubble();
        int[] array = {2, 5, 1, 6, 4, 9, 8, 5, 3, 1, 2, 0};
        int[] sort = bubble.sort(array);
        for (int num : sort) {
            System.out.print(num + "\t");
        }
    }
}
```

2、选择排序

【请写出选择排序的代码】

狂神社群笔记资料，禁止外传，本人QQ：24736743

选择排序（Selection sort）是一种简单直观的排序算法。它的工作原理是每一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，然后，再从剩余未排序元素中继续寻找最小（大）元素，然后放到排序序列的末尾。以此类推，直到全部待排序的数据元素排完。 选择排序是不稳定的排序方法。

```
class SelectSort{
    public int[] sort(int arr[]) {
        int temp = 0;
        for (int i = 0; i < arr.length - 1; i++) {// 认为目前的数就是最小的，记录最小数的下标
            int minIndex = i;
            for (int j = i + 1; j < arr.length; j++) {
                if (arr[minIndex] > arr[j]) {// 修改最小值的下标
                    minIndex = j;
                }
            }// 当退出for就找到这次的最小值,就需要交换位置了
            if (i != minIndex) {//交换当前值和找到的最小值的位置
                temp = arr[i];
                arr[i] = arr[minIndex];
                arr[minIndex] = temp;
            }
        }
        return arr;
    }

    public static void main(String[] args) {
        SelectSort selectSort = new SelectSort();
        int[] array = {2, 5, 1, 6, 4, 9, 8, 5, 3, 1, 2, 0};
        int[] sort = selectSort.sort(array);
        for (int num : sort) {
            System.out.print(num + "\t");
        }
    }
}
```

稀疏数组

https://blog.csdn.net/baolingye/article/details/99943083

狂神社群笔记资料，禁止外传，本人QQ：24736743