1. **数组**

声明：本章节主要以java语言来阐述，以一维数组为例。

* 1. 什么是数组?

在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。

数组(array)是一种最简单的复合数据类型，它是有序数据的集合，数组中的每个元素具有相同的数据类型，可以用一个统一的数组名和不同的下标来确定数组中唯一的元素。根据数组的维度，可以将其分为：一维数组、二维数组和多维数组等。

数组可以理解为，一个装有同种数据类型集合的一个容器，就是指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。数组本身属于一种引用数据类型。

组就是一种用于存储数据的方式，能存数据的地方我们称之为容器，容器能装的东西就是数组的元素。数组可以装任意类型的数据，虽然可以装任意类型的数据，但是定义好的数组只能装一种类型的元素。也就是数组一旦定义，那么里面存储的数据类型也就确定了。

数组有哪些特征：

1. 数组在内存中存储的数据必须是相同的数据类型的数据，并且它们存储在一个连续的空间;
2. 声明一个数组就是在内存空间中划出一串连续的空间;
3. 数组名代表的是连续空间的首地址；例如int a[10]，则a就是该数组的首地址;

通过首地址可以依次访问数组所有元素;

* 1. 如何定义数组？

1.2.1声明数组

Java中声明数组的方式：

DataType[] arr;

DataType []arr;

DataType arr[];

但是以上只是声明了数组，并不能直接使用，因为没有为其分配内存空间，所以此时必须初始化。

1.2.2 初始化数组

初始化数组又叫实例化数组，按照方式可以分为：静态初始化和动态初始化。

静态初始化：由程序猿自己确定每个数组元素的初始值，而数组的长度则由系统自己决定，例如以下两种方式：

int[] a = {1,2,3,4,5}; // DataType[] 数组变量名 = {值, 值, 值, …};

String[] arr3 = new String[]{"1", "a", "a13"}; // DataType[]数组变量名 = DataType[]{值, 值, 值, …};

动态初始化：初始化时由程序猿确定数组的长度，而没一个数组元素的值，则由系统来分配初始值，例如：

int[] b=new int[5]; // DataType[] 数组变量名 = new DataType[长度];

**数组一定要初始化吗？**

答案看上去肯定是一定要，但是我在实践中发现却是不一定需要，要想搞清楚这个问题，我们需要理解数组变量和数组对象的区别。而在认识他们之前需要先了解Java的数据类型。Java的数据类型分为两种，基本数据类型和引用数据类型。

数组变量是一个引用类型的变量，数组变量是指向堆内存当中的数组对象，并不是数组本身。当改变一个数组变量所引用的数组，就可以造成数组长度可变的假象。实际上数组对象本身的长度本没有变化，只是数组变量指向了一个新的数组对象。所以对于数组变量来说，他并不需要初始化，我们常说的初始化其实是初始化数组对象而非数组变量，有时候我们不进行初始化，而让数组变量指向一个有效的数组对象，数组也可以使用，例如：

int[] a = {0,1,2,3,4};

int[] b;

b = a;

1.2.3声明与初始化变量类型

实例变量永远会有默认值。如果没有明确的赋值给实例变量，或者没有调用setter，实例变量仍有默认值。

* integers 0
* floating points 0.0
* Booleans false
* references null
  1. 数组的图形解释
  2. 数组的常用方法
     1. Arrays类中的静态方法

# Arrays类介绍：此类包含用来操作数组（比如排序和搜索）的各种方法，它提供的所有方法均为静态方法。

1. toString方法：将数组按照默认格式输出为字符串；
2. sort方法：将数组进行升序排序；

public static String sort(Array[] array, int fromIndex, int toIndex); // 参数1数组；(参数2,3可以没有) 参数2,3为数组索引位置，用来对指定范围的数据进行排序。

注意：

* 参数array如果是数值默认升序排序；
* 是字符串(英文)按照字母顺序升序排序；
* 是字符串(中文)按照字符串的编码数字升序排序；
* 如果是自定义类型，那么自定义的类必须有comparable或者Comparator接口的支持；

1. binarySearch方法：使用二分法查找元素在数组中的索引位置；

public static int binarySearch(Array[] array, int fromIndex, int toIndex, data key);

// 参数1数组； 参数2,3为数组索引位置(参数2,3可以没有)，用来对指定范围的数据进行排序；参数4为要查找的值；

注意：

使用二分搜索法来搜索指定数组，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）。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。

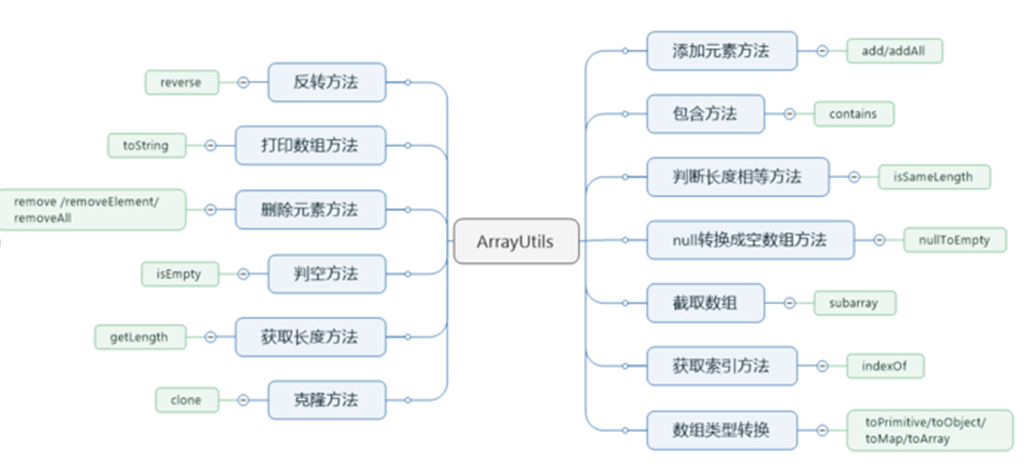
1. fill方法：指定位置进行数值填充；

public static void fill((Array[] array, int fromIndex, int toIndex，data val);

// 参数1数组；参数2,3为数组索引位置(参数2,3可以没有)；参数4是需要填充的值；

1. equals方法：比较数组中元素值是否相等；
   * 1. ArrayUtils类中的静态方法

ArrayUtils类是Apache提供的专门用来处理数组的工具类，提供很多有用的方法，下面是其一个方法蓝图：



简单介绍一些常用方法：

1）添加方法

add(T[] array,int index,T val);

2）克隆方法

clone(T[] array);

3）包含方法

contains(T[] array,T val); // 判断数组array是否包含val的值；返回类型为boolean；

4）获取索引方法

// 从正序开始搜索，搜到则返回当前的Index，否则返回-1；

indexOf(T[] array,data val, int startIndex);

// 参数1是数组；参数2是具体要查找的值；参数3可以指定起始搜索位置（参数3可以没有）；

5）删除元素方法

remove(T[] array,int index); //删除指定下标的元素；

removeElement(T[] array, T element); //删除第一次出现的元素；

removeAll(T[] array,int... indices); // 删除所有指定下标的元素；

removeElements(T[] array,T... values); //删除所有出现的指定元素；

* + 1. 代码实现

以上一些常用方法的代码实现如下：

|  |
| --- |
| **package** com.violet.test.array;  **import** org.apache.commons.lang3.ArrayUtils;  **import** java.util.ArrayList; **import** java.util.Arrays;  **public class** ArrayDemo1 {  **public static void** main(String[] args) {  *// 数组声明* String[] arr1 = **new** String[5];  String[] arr2 = {**"a"**, **"e"**, **"c"**, **"d"**, **"b"**};  **int**[] arr3 = **new int**[]{1, 2, 3, 4, 5};   *// Arrays常用静态方法  // 输出字符串：Arrays.toString方法* System.***out***.println(Arrays.*toString*(arr3)); *// [1, 2, 3, 4, 5]   // 排序：Arrays.sort方法* Arrays.*sort*(arr2);  System.***out***.println(Arrays.*toString*(arr2)); *// [a, b, c, d, e]   // 二分查找：Arrays.binarySearch方法* System.***out***.println(Arrays.*binarySearch*(arr3, 4)); *// 3   // 数组填充：Arrays.fill方法* Arrays.*fill*(arr3, 1, 3, 6);  System.***out***.println(Arrays.*toString*(arr3)); *// [1, 6, 6, 4, 5]   // 数组转ArrayList* String[] strArr1 = {**"a"**, **"b"**, **"c"**, **"d"**, **"e"**};  ArrayList<String> arrList = **new** ArrayList<String>(Arrays.*asList*(strArr1));  *// ArrayList转数组* String[] strArr2 = **new** String[arrList.size()];  arrList.toArray(strArr2);   *// ArrayUtils常用方法  // 添加：ArrayUtils.add方法* **int**[] arr4 = ArrayUtils.*add*(arr3, 1, 10);  System.***out***.println(Arrays.*toString*(arr4));*// [1, 10, 6, 6, 4, 5]   // 克隆：ArrayUtils.clone方法* String[] arr5 = ArrayUtils.*clone*(arr2);  System.***out***.println(Arrays.*toString*(arr5)); *// [a, b, c, d, e]   // 包含方法:ArrayUtils.contains()* System.***out***.println(ArrayUtils.*contains*(arr5, **"c"**));*// true   // 获取索引方法:ArrayUtils.indexOf()* System.***out***.println(ArrayUtils.*indexOf*(arr5, **"c"**)); *// 2* System.***out***.println(ArrayUtils.*indexOf*(arr5, **"k"**)); *// -1   // 删除元素：ArrayUtils.remove* String[] removed = ArrayUtils.*remove*(arr5,3 );  System.***out***.println(Arrays.*toString*(removed));*// [a, b, c, e]* String[] removed2 = ArrayUtils.*removeElement*(arr5, **"b"**);  System.***out***.println(Arrays.*toString*(removed2));*// [a, c, d, e]* } } |

* 1. 数组的常见误区或问题

1. 数组长度一旦声明，便不可改变，不可追加。
2. NullPointerException空指针异常。

原因：引用类型变量没有指向任何对象，而访问了对象的属性或者是调用了对象的方法。

1. ArrayIndexOutOfBoundsException 数组越界异常。

原因：访问了不存在的索引值。数组arr的合法错误范围是[0, arr.length-1]，当你访问这之外的索引时会报这个错。

* 1. 解题技巧
  2. 数组的优缺点

1. 优点

* 定义简单，而且访问很方便；
* 能存储大量数据；
* 随机访问的效率很高 ；复杂度为O(1)；根据下标访问元素效率高（元素连续分配空间）。

2） 缺点

* 数组的元素的类型必须一致；
* 数组的空间必须是连续的，这就造成数组在内存中分配空间时必须找到一块连续的内存空间。所以数组不可能定义的太大，因为内存中不可能有那么多大的连续的内存空间；
* 根据内容查找元素效率比较低，需要逐个比较；
* 删除、插入元素效率比较低，需要移动大量的元素；
* 数组定长，不能自动扩容，不适合动态存储；