Java中的Arrays类（数组工具类）

1. **Arrays类**是一个用来**操纵数组的**工具类，存在于**java.util包**中，直接继承于**java.lang.Object类**。**Arrays类不提供构造方法和字段，故不可实例化，不能用来生成对象，也没有必要生成对象，只提供成员方法，且都是静态方法，可以直接利用类名进行调用。如Arrays.sort方法。Arrays类是个直接操作数组的工具类，其中主要的方法就是排序、二分查找方法和toString方法：sort和binarySearch、toString。**

**注意： Arrays类中的方法都是静态方法。**

1. **Arrays类**中的所有方法都会抛出一个空指针异常:**NullPointerException。**
2. **Arrays类中的方法主要有几种:**
3. **sort方法;**
4. **binarySearch方法;**
5. **asList方法（联想到集合Collection中toArray方法）：将数组转换成List集合**
6. **copyOf方法；**
7. **copyOfRange方法**
8. **equals方法；**
9. **fill方法；**
10. **setAll方法；**
11. **stream方法；**
12. **spliterator方法；**
13. **toString方法；**
14. **hashCode方法；**
15. **parallelSort方法；**
16. **parallelSetAll方法；**
17. parallelPrefix**方法；**
18. **从Object类继承的方法。**
19. **search方法**：
20. **二分查找**:

**二分查找法**是先判断区间**中间值**的大小然后**决定选择前一半区间还是后一半区间，默认数组按照大小顺序排好的**。

1. 传入**基本数据类型的数组**，然后在数组中寻找此类型的**某个数据**。八种基本数据类型除了boolean之外，其余七个都有，且每个类型有两个方法，**一个是在整个数组范围内查找，一个是限定起始点和终止点**。binarySearch方法，返回的int值，如果数组中存在要找的值，则按照二分法算法返回此元素的索引，否则返回负数，负数值为**(-(*insertion point*) - 1)**，*insertion point为****应当插入的位置***。

如: char[] cs1 = {'u','u','e','u','u'};

int index = Arrays.binarySearch(cs1,'u');

System.out.println(index);**//返回3**

char[] cs1 = {'u','u','e','u','u'};

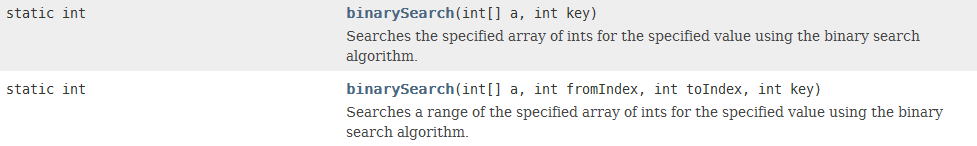
int index = Arrays.binarySearch(cs1,'a');

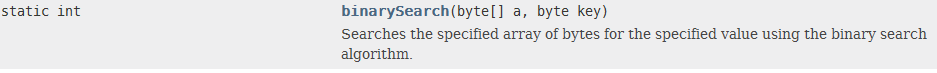
System.out.println(index);**//返回-1**

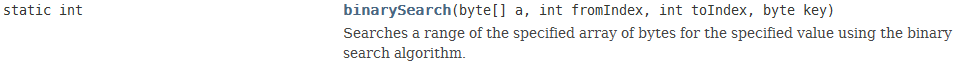
char[] cs1 = {'b','u','e','u','u'};

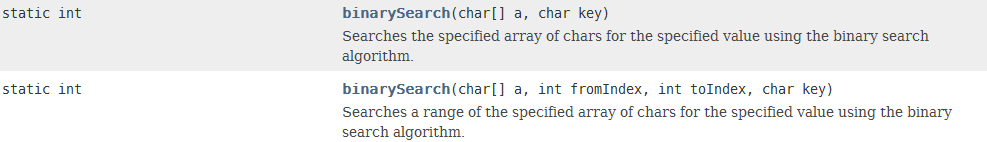
int index = Arrays.binarySearch(cs1,'c');

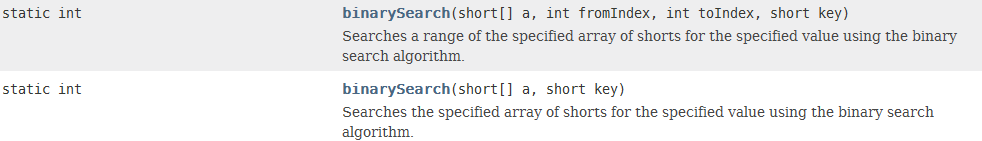
System.out.println(index);**//返回-2**

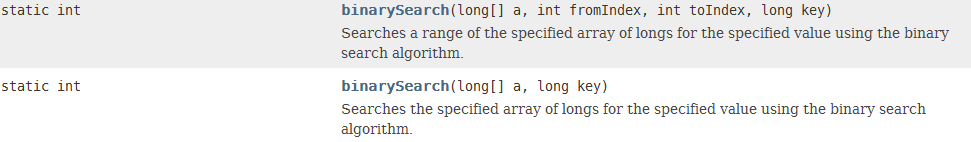


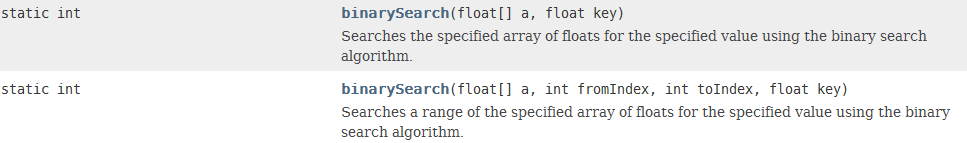


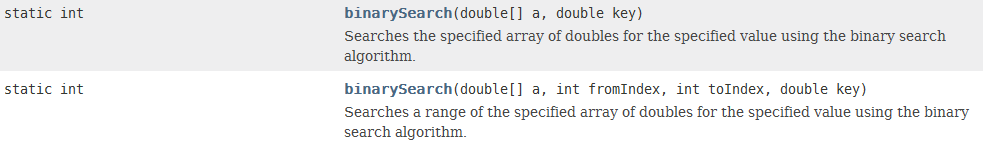




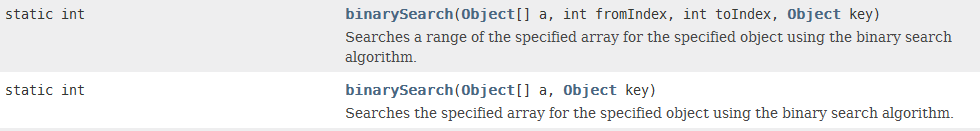




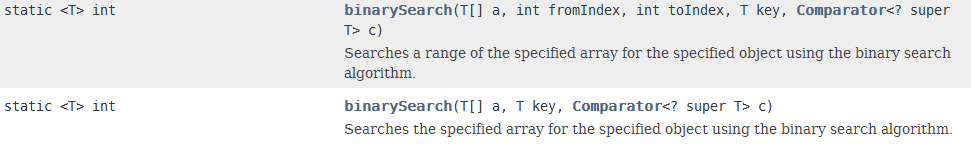




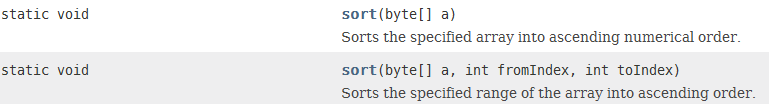
1. 可以接收**Object类型**：**引用数组**。



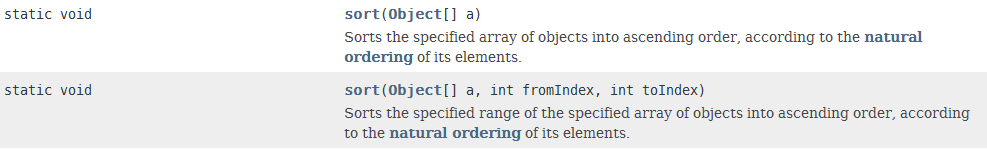
1. **接收T数组：**



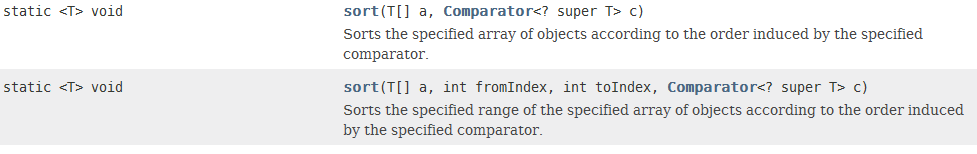
1. **sort方法**:
2. **基本数据类型数组**：除了**boolean数组**之外。



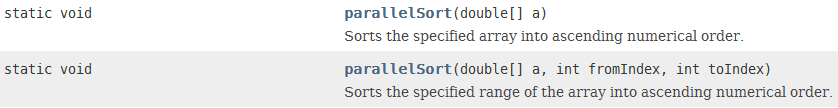
1. Object数组:



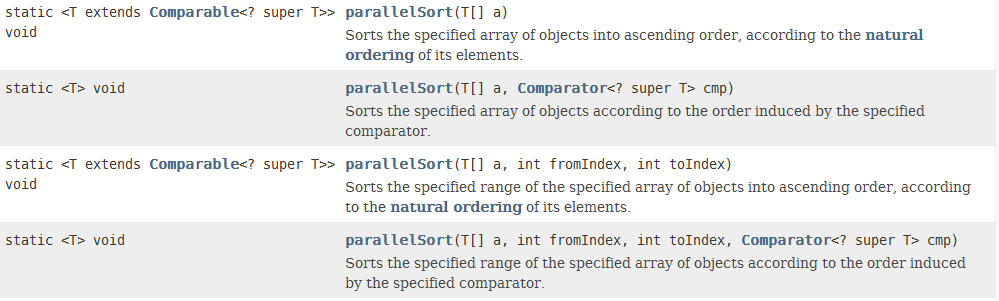
1. T数组：



1. **parallelSort方法：**
2. **基本数据类型数组：每个类型有两个方法。**



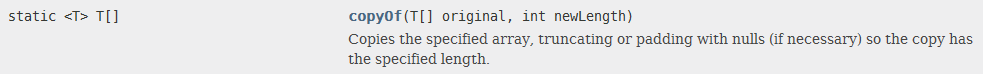
1. **T数组：**



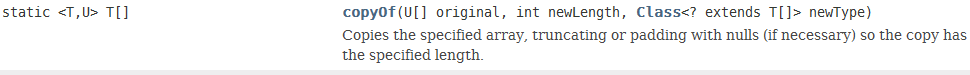
1. **copyOf方法:**
2. **基本数组类型数组:返回一个新的基本数据类型数组。八种基本数据类型数据都有。**



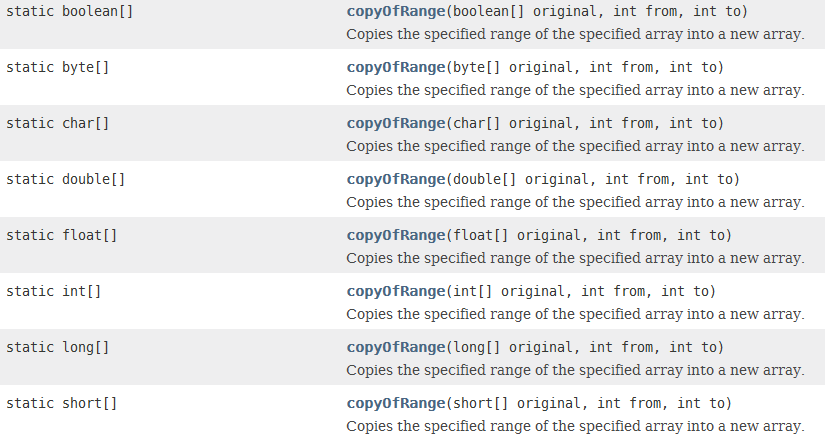
1. **接收T：**



1. **接收U：**



1. **copyOfRange方法**：与copyOf方法极其相似。
2. **基本数据类型数组：**



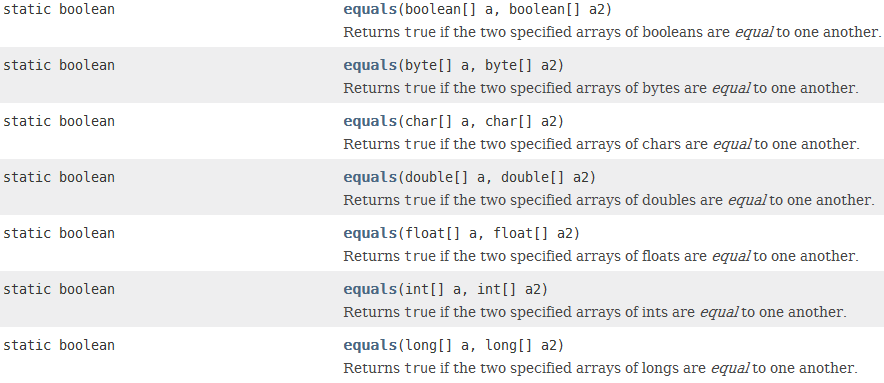
1. **接收T:**



1. **接收U:**



1. **equals方法**:
2. **基本数据类型数组**：





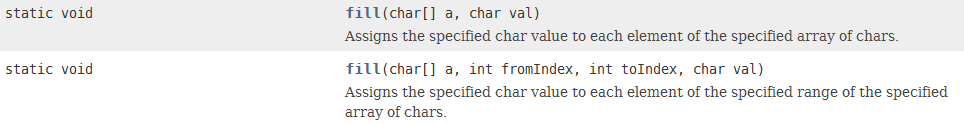
1. **接收Object数组**：



1. **deepEquals方法**：

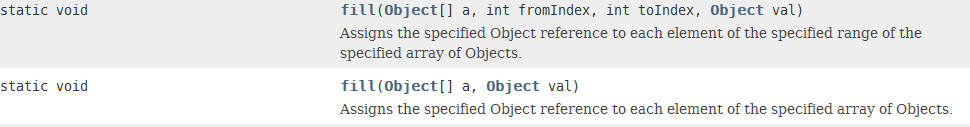


1. **fill方法**：
2. 基本数据类型:八种都有。



。。。。。。

1. 接收Object类;



1. **hashCode方法：**
2. 基本数据类型数组





1. Object数组：

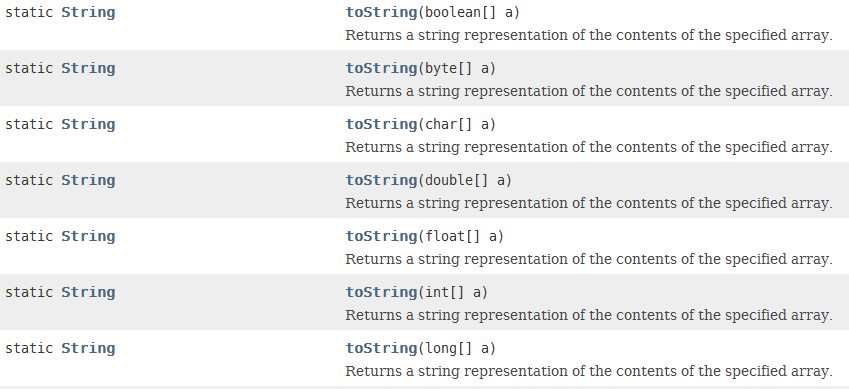


1. deepHashCode方法：



1. **toString方法**：
2. **基本数据类型**：





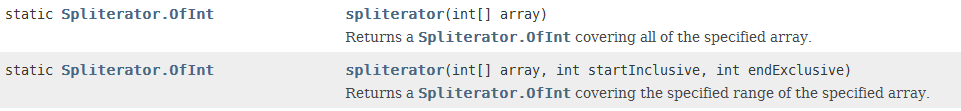
1. **接收Object数组**：



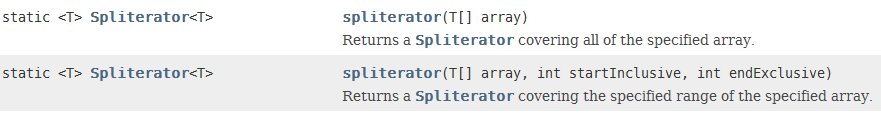
1. **deepToString方法**：



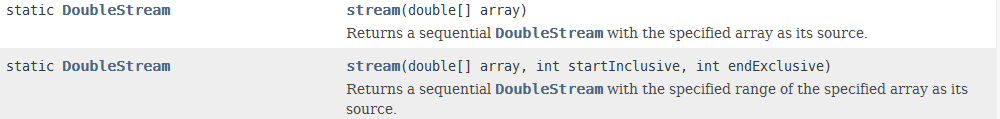
1. spliterator方法：
2. 基本数据类型数组：



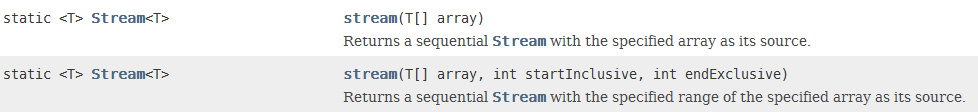
1. T数组：



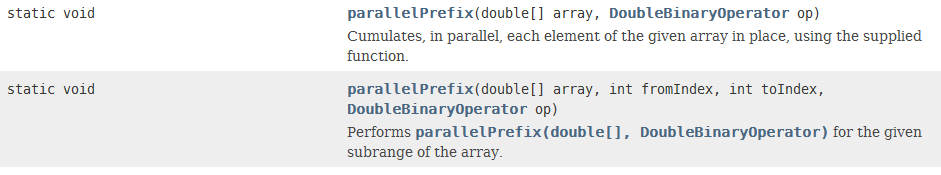
1. **stream方法**：
2. **基本数据类型数组：long、int 、double**

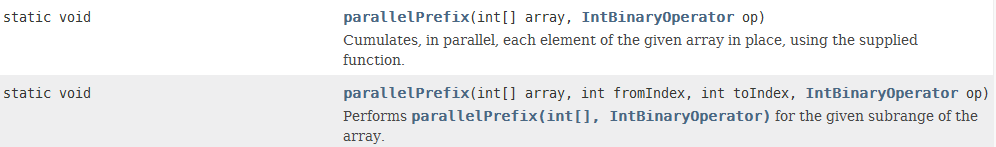


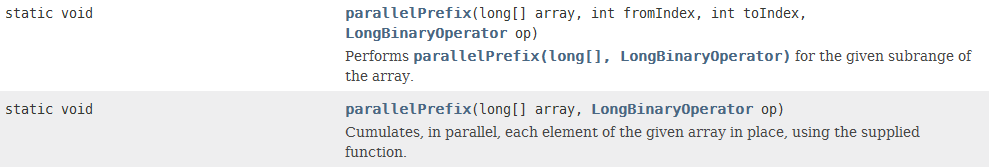
1. **T数组**：



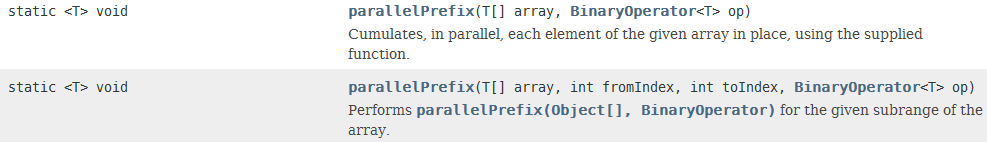
1. parrelPrefix方法:
2. **基本数据类型数组：double、int、long**



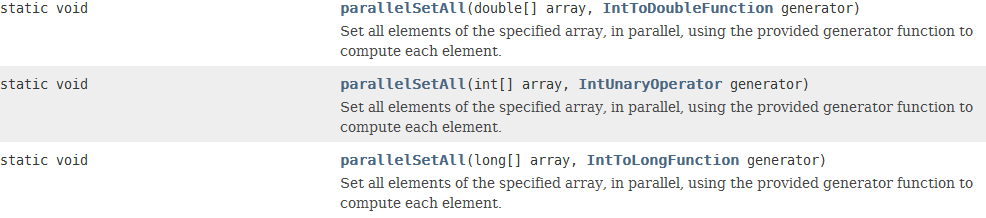




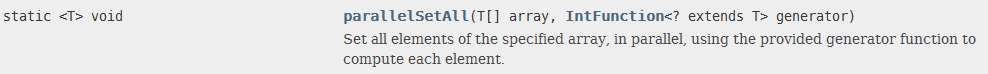
1. **T数组**：



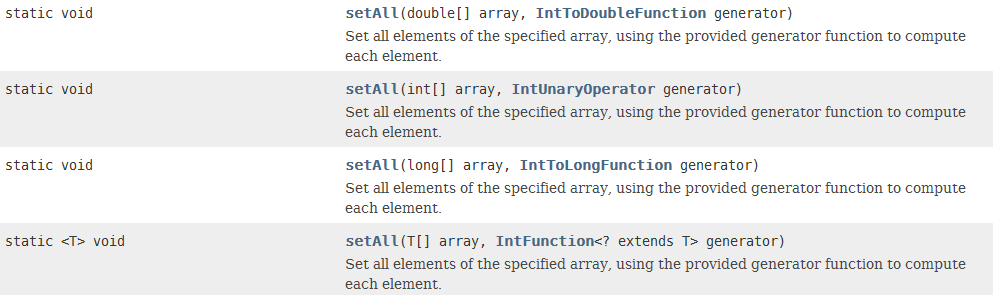
1. parrelSetAll方法：
2. 基本数据类型:



1. T 数组：



1. setAll方法：



1. **asList方法**：是一个**静态方法**，可以将**数组**转换成**List集合**。（数组的工具类中的方法不足，可以转成集合利用**集合中的方法**进行**处理**），可以传入数组名，也可以直接传入元素，会自动封装成一个数组，再进行转换。



1. **集合和数组之间的转换：**
2. **数组转成List集合：利用Arrays类中的asList静态方法。**

**将数组转成集合好处在于:可以利用List集合的方法操作数组中的元素。**



* 1. **如果数组中的元素是基本数据类型，asList方法就会把整个数组封装成一个集合元素存到List集合中去；此时泛型为xxx[] ,如 List<int[]> list = Arrays.asList(a);**

**相应的泛型为int[] 、double[]、float[]、long[]等等。**

**② 如果数组的元素是对象，则asList方法把数组中的每个对象封装成集合元素存到List集合中去，泛型就是相应的类名如String。**

**注意：数组转成的集合，由于数组的长度是不能改变的，此时相应的集合的操作中也不能修改长度，例如add、remove方法都不能使用，会抛出java.lang.NoSupportedOperationException。(不支持操作异常)**

**练习：**

**如: String[] strarr = {"wo","ai","ni"};**

**List<String> list = Arrays.asList(strarr);**

**for(Iterator<String> it = list.iterator();it.hasNext();) {**

**System.out.println(it.next());**

**}//输出结果为：**

**int[] a = {2,3,5,5,4};**

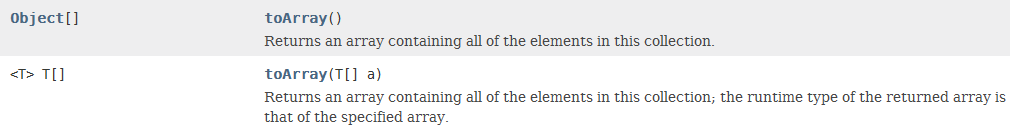
**List<int[]> list = Arrays.asList(a);**

**for(Iterator<int[]> it = list.iterator();it.hasNext();) {**

**System.out.println(it.next());**

**}//输出结果为：**

1. **集合转成数组:利用Collection接口中声明的toArray方法。**
   1. **toArray方法：**



* 1. **第一个方法toArray返回的就是一个数组，大小就是集合中元素的个数。**

**第二个方法可以指定数组类型和数组大小，如果指定的数组大小小于集合中的元素个数，则就会默认返回一个与集合元素大小相同的数组，如果指定的数组的大小大于集合中的元素个数，则就会把数组中多余的元素默认为空。所以，一般都指定数组大小为集合的大小，list.size。**

* 1. **集合转成数组，可以对集合中的元素操作的方法进行限定，但是不允许对其进行增删。**

如:

List<String> list = new ArrayList<String>();

list.add("nihao");

list.add("iloveyou");

String[] str = list.toArray(new String[list.size()]);

for (int i = 0; i < str.length; i++) {

System.out.println(str[i]);

}//结果为

1. Arrays类中的**toString方法**实现原理：（以**int数组**为例）

private static String myToString(int[] a) {

int imax = a.length -1;

if(imax == -1) {

return "[]";

}

StringBuilder sb = new StringBuilder();

sb.append("[");

**for (int i = 0;; i++) {//每次循环是不再判断条件，提高了效率（亮点）**

sb.append(a[i]);

if(i == imax )

return sb.append("]").toString();

sb.append(",");

}

}