简述

java.util.Arrays 类是 JDK 提供的一个工具类，用来处理数组的各种方法，而且每个方法基本上都是静态方法，能直接通过类名Arrays调用

与List相关

asList

```
public static <T> List<T> asList(T... a) {
    return new ArrayList<>(a);
}
```

下面每一项都是重点

1. 这里返回的ArrayList 是 java.util.Arrays.ArrayList， 非 java.util.ArrayList

2. 这里的list是一个固定长度的数组，**只能对其进行查看或者修改，但是不能进行添加或者删除操作，否则会抛出异常**

```
List<String> list = Arrays.asList("a", "b", "c");
list.add("d");

Exception in thread "main" java.lang.UnsupportedOperationException
    at java.util.AbstractList.add(AbstractList.java:148)
```

3. 引用类型的数组和基本类型的数组区别

```
String[] str = {"a","b","c"};
List<String> listStr = Arrays.asList(str);
System.out.println(listStr.size());//3

int[] i = {1,2,3};
List<int[]> listI = Arrays.asList(i);//注意这里List参数为 int[] ，而不是 int
System.out.println(listI.size());//1

Integer[] in = {1,2,3};
 List<Integer> listIn = Arrays.asList(in);//这里参数为int的包装类Integer，所以集合长度为3
 System.out.println(listIn.size());//3
```

4. 返回的列表ArrayList里面的元素都是引用，不是独立出来的对象

```
String[] str = {"a","b","c"};
List<String> listStr = Arrays.asList(str);
//执行更新操作前
System.out.println(Arrays.toString(str));//[a, b, c]
listStr.set(0, "d");//将第一个元素a改为d
//执行更新操作后
System.out.println(Arrays.toString(str));//[d, b, c]
```

5. 已知数组数据，如何快速获取一个可进行增删改查的列表List

```
String[] str = {"a", "b", "c"};
List<String> listStr = new ArrayList<>(Arrays.asList(str));
listStr.add("d");
System.out.println(listStr.size());//4
```

ps: 第 5 点中的ArrayList是 java.util.ArrayList

排序相关

sort

1. 默认排序

```
int[] array = {1,3,6,7,4,2};
Arrays.sort(array);
System.out.println(Arrays.toString(array));
输出 : [1, 2, 3, 4, 6, 7]
```

2. 局部排序

```
int[] array = {1,3,6,7,4,2};
Arrays.sort(array,2,5);
System.out.println(Arrays.toString(array));
输出:  [1, 3, 4, 6, 7, 2]
说明：包含开始下标，不包含结束下标
```

3. 改变排序规则的两种方式

- 排序对象实现Comparable接口
- 传入参数包含 排序方式Comparator接口

parallelSort

使用同上，区别在于支持并行计算

查找相关

binarySearch

二分查找、可以指定查找范围、查找不到返回 -1

注意：需要数组有序

数组复制

copyOf

拷贝数组元素。底层采用 System.arraycopy() 实现，这是一个native方法

可以指定拷贝范围

数组比较

hashCode

循环计算数组中每一个元素的hashcode，然后合并

equal

循环每一个元素，挨个对比，equals

deepHashCode

循环计算数组每一个元素的hashcode，然后合并，区别 hashCode在于，这里是多维度，深层次的

deepEqual

循环每一个元素，挨个对比，区别 equals在于，这里是多维度，深层次的

其他方法

fill

该系列方法用于给数组赋值，并能指定某个范围赋值

```
//给a数组所有元素赋值 val
public static void fill(int[] a, int val) {
    for (int i = 0, len = a.length; i < len; i++)
        a[i] = val;
}

//给从 fromIndex 开始的下标，toIndex-1结尾的下标都赋值 val,左闭右开
public static void fill(int[] a, int fromIndex, int toIndex, int val) {
    rangeCheck(a.length, fromIndex, toIndex);//判断范围是否合理
     for (int i = fromIndex; i < toIndex; i++)
         a[i] = val;
}
```

toString 和 deepToString

toString 用来打印一维数组的元素，而 deepToString 用来打印多层次嵌套的数组元素

Stream

将数组转为流

parallelPrefix

此重载方法以累积方式对输入数组的每个元素执行操作

```
使用提供的函数并行地累积给定数组的每个元素。例如，如果数组最初保持[2,1,0,3]并且操作执行加法，
则返回时阵列保持[2,3,3,6]。对于大型数组，并行前缀计算通常比顺序循环更有效，效率更高
```

```
int[] arr2=new int[]{1,8,51,13,46,11,22};
Arrays.parallelPrefix(arr2, new IntBinaryOperator() {
    @Override
    public int applyAsInt(int left, int right) {
        //left代表数组中前一个索引处的元素，计算第一个元素时，left为1
        //right代表数组中当前索引处的元素
        return left*right;
    }
});
 System.out.println(Arrays.toString(arr2));

 输出 ：[1, 8, 408, 5304, 243984, 2683824, 59044128]
```

parallelSetAll

按照规则生成数组

```
int[] array = new int[10];
Arrays.parallelSetAll(array, i -> i);
System.out.println(Arrays.toString(array));

```

```
输出：[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
int[] arr3=new int[5];
Arrays.parallelSetAll(arr3, new IntUnaryOperator() {
    @Override
    public int applyAsInt(int operand) {
        //operand代表正在计算的元素的索引
        return operand*5;
    }
});
System.out.println(Arrays.toString(arr3));

输出：[0, 5, 10, 15, 20]
```

spliterator

Spliterator是将一个stream进行对半平分的操作类

Arrays.parallelSetAll 和 IntStream.range可以生成一个指定长度Int的Stream