/*Rotate an array of n elements to the right by k steps.

For example, with n = 7 and k = 3, the array [1,2,3,4,5,6,7] is rotated to [5,6,7,1,2,3,4].

Note: Try to come up as many solutions as you can, there are at least 3 different ways to solve this problem.
*/

-
- 思想：
- (1) 方法一（允许分配额外空间,更快）： 首先对输入k进行取余操作，然后开辟一个长度为k的新数组，复制原数组的后k个元素到新数组中，再将原数组的前 length - k个元素向后复制移动k步，再将新数组的元素复制到原数组的前k部分

```
    int numsLength = nums.length;
    k = k % numsLength;

    int[] helper = new int[k];

    System.arraycopy(nums, numsLength - k, helper, 0, k);
    System.arraycopy(nums, 0, nums, k, numsLength - k);
    System.arraycopy(helper, 0, nums, 0, k); */
```

- (2) 方法二（不分配额外空间，相对慢些）： 首先对k进行取余操作，然后翻转整个数组，然后翻转数组的前k部分，然后翻转数组的后 length - k 部分

```
public void rotate(int[] nums, int k) {



    // 方法二：不需要开辟新空间，相对慢点
    int numsLength = nums.length;
    k = k % numsLength;

    reverse(nums, 0, numsLength - 1);
    reverse(nums, 0, k - 1);
    reverse(nums, k, numsLength - 1);
}

private void reverse(int[] nums, int startIndex, int endIndex) {

    while (startIndex < endIndex){
        int temp = nums[startIndex];
        nums[startIndex] = nums[endIndex];
        nums[endIndex] = temp;

        startIndex++;
        endIndex--;
    }
}
```

- (3) System.arraycopy是native方法，作用是复制数组的一部分，第一个参数代表原数组，第二个参数代表原数组的起始复制下标（包括），第三个参数代表目标数组，第四个参数代表目标数组的起始复制下标（包括），第五个参数代表复制元素个数
- (4) System.arraycopy可以从把当前数组的一部分，移动到当前数组的另一部分