The set [1,2,3,…,*n*] contains a total of *n*! unique permutations.

By listing and labeling all of the permutations in order,  
We get the following sequence (ie, for *n* = 3):

1. "123"
2. "132"
3. "213"
4. "231"
5. "312"
6. "321"

Given *n* and *k*, return the *k*th permutation sequence.

**Note:** Given *n* will be between 1 and 9 inclusive.

题目，我们按照顺序进行排列组合，给出k，你需要给出第k个的排列组合。例如k=1，你返回123.

我们注意观察这个整体的数列，1开头的个数其实是由后边两个数组的排列组合的个数决定的，也就是（n-1）！=2\*1=2，依次2，3相同。1，固定，然后找第二位，2开头的由后边的排序组合个数决定的也就是（n-2）！=1\*1=1，到3为0！=1.同理别的也是。

当k=5的时候，我们怎么才能拼凑出该排列组合呢？首先1开头的两个，2开头的两个。3之前就会由4个，2个(n-1)!,3是第三个(n-1)!。也就是k/(n-1)!=2，k里边包含两个最开头不同的排列组合，那么肯定为第三个。（k/(n-1)!）+1=2+1=3刚好是排列组合的按顺序开头的数字，但是我们存储的时候3是存储再第二位的，所以直接用k/(n-1)!=2当作索引去找就可以了。然后我们找第二位，第二位的k为刨除之前掠过整体的组合，也就是减去1和2开头的数据。K=k%(n-1)!。下一个的位数个数为(n-2)!，重复。

注意我们每次找完一个数字之后，需要做的是删除该数字，因为排列组合中不能有重复。

public static String getPermutation(int n, int k) {

int[] fact = new int[n+1];//保存阶乘的数组,容量为n+1，因为0的阶乘需要也算进去

fact[0]=1;//0的阶乘等于1

List<Integer> nums = new ArrayList<>();//保存数字序列

int sum = 1;//用于计算阶乘

for(int i = 1;i<=n;i++){

sum \*= i;

fact[i]=sum;

nums.add(i);

}

k--;//因为保存的位置是从0开始的，所以需要k-1才能够正确访问到该位置的排列组合

StringBuffer sb = new StringBuffer();//用于保存k索引出的排列组合

for(int i = 1;i<=n;i++){

int index = k/fact[n-i];//算出i位置的索引

sb.append(nums.get(index));//加入到sb中

nums.remove(index);//从原序列中移除，因为是不重复的排列组合

k %= fact[n-i];

}

return sb.toString();

}