题目：0，1，，，，，n-1这n 个数字排成一个圆圈，从数字0开始每次从这个圆圈中删除第m个数字。求出这个圆圈里剩下的最后一个数字.

例如，0，1，2，3，4这5个数字组成的一个圆圈，从数字0开始每次删除第3个数字，则删除的前四个数字依次是2，0，4，1因此最后剩下的数字是3.

本题就是著名的约瑟夫环的问题。我们介绍两种方法：一种方法是用环形链表模拟圆圈的经典解法，第二种方法是分析每次被删除的数字的规律并直接计算出圆圈中最后剩下的数字。

第一种方法：环形列表

```java
public void lastRemaining(int totalNum,int index){
    //初始化人数
    List<Integer> start = new ArrayList<Integer>();
    for(int i = 0;i< totalNum;i++){
        start.add(i);
    }
    //从第k个开始计数
    int k = 0;
    while(start.size() > 0){
        k = k+ index;
        //第m个人的索引位置
        k = k %(start.size()) -1;
        //判断是否到队尾
        if(k < 0){
           System.out.println(start.get(start.size()-1));
           start.remove(start.size()-1);
           k = 0;
        }else{
         //   System.out.println(start.get(k));
           start.remove(k);
        }
    }
 }
```

因为不判断，remove(k)会越界。

首先我们定义一个关于n和m的方程f(n,m)，表示每次在n个数字0，1，...,n-1中每次删除第m个数字最后剩下的数字。

在这n个数字中，第一个被删除的数字是（m-1)%n.为了简单起见，我们把（m-1)%n记为k，那么删除k之后剩下的n-1个数字为0，1，。。。。k-1，k+1,.....n-1。并且下一次删除从数字k+1,.....n-1,0,1,....k-1。该序列最后剩下的数字也应该是关于n和m的函数。由于这个序列的规律和前面最初的序列不一样（最初的序列是从0开始的连续序列），因此该函数不同于前面的函数，即为f'(n-1,m)。最初序列最后剩下的数字f(n,m)一定是删除一个数字之后的序列最后剩下的数字，即f(n,m)=f'(n-1,m).

接下来我么把剩下的这n-1个数字的序列k+1,....n-1，0，1，,,,,,k-1做一个映射，映射的结果是形成一个从0到n-2的序列

k+1 ------> 0

k+2 ---------> 1

.....

n-1 ----- > n-k-2

0 -------> n-k-1

1 ---------> n-k

.....

k-1 ----------> n-k

我们把映射定义为p，则p(x) = (x-k-1)%n。它表示如果映射前的数字是x,那么映射后的数字是（x-k-1)%n.该映射的逆映射是p-1(x)= (x+k+1)%n.

由于映射之后的序列和最初的序列具有同样的形式，即都是从0开始的连续序列，因此仍然可以用函数f来表示，记为f(n-1,m).根据我们的映射规则，映射之前的序列中最后剩下的数字f'(n-1,m) = p-1[(n-1,m)] = [f(n-1,m)+k+1]%n，把k= (m-1)%n代入f(n,m) = f'(n-1,m) =[f(n-1,m)+m]%n.

经过上面的复杂的分析，我们终于找到一个递归的公示。要得到n个数字的序列中最后剩下的数字，只需要得到n-1个数字的序列和最后剩下的数字，并以此类推。当n-1时，也就是序列中开始只有一个数字0，那么很显然最后剩下的数字就是0.我们把这种关系表示为：

$$f(n,m)=\begin{cases}0 & n=1\\ [f(n-1,m)+m]\%n & n>1\end{cases}$$

```java
int LastRemaining(int n,int m){
        if(n<1||m<1) return -1;
        int last=0;
        for(int i=2;i<=n;i++)
            last=(last+m)%i;
        return last;
    }
```

代码最后为了得到f(n,m).