Java-排序数组-比赛

题目描述：一个有N个选手参加比赛，选手编号为1~N(3<=N<=100)，有M(3<=M<=10)个评委对选手进行打分。打分规则为每个评委对选手打分，最高分 10 分， 最低分 1 分。 请计算得分最多的 3 位选手的编号。如果得分相同，则得分高分值最多的选手排名靠前(10 分数量相同，则比较 9 分的数量，以此类推，用例中不会出现多个选手 得分完全相同的情况)。

输入描述：第一行为半角逗号分割的两个正整数，第一个数字表示 M(3<=M<=10)个评委，第二个数字表示 N（3<=N<=100）个选手。 第 2 到 M+1 行是半角逗号分割的整数序列，表示评委为每个选手的打分，0 号下标数字表示 1 号选手分数，1 号下标数字表示 2 号选手分数，依次类推。

输出描述：选手前 3 名的编号。 注：若输入为异常，输出-1，如 M、N、打分不在范围内。

示例

示例1
输入：4,5
10,6,9,7,6
9,10,6,7,5
8,10,6,5,10
9,10,8,4,9
输出：2,1,5
说明：第一行代表有4个评委，5个选手参加比赛
矩阵代表是4*5，每个数字是选手的编号，每一行代表一个评委对选手的打分排序，
2号选手得分36分排第1，1号选手36分排第2，5号选手30分(2号10分值有3个，1号10分值只有1个，所以2号排第一)

示例2
输入：2,5
7,3,5,4,2
8,5,4,4,3
输出：-1
说明：只有2个评委，要求最少为3个评委

示例3

输入：4,2
8,5
5,6
10,4
8,9
输出：-1
说明：只有2名选手参加，要求最少为3名

示例4
输入：4,5
11,6,9,7,8
9,10,6,7,8
8,10,6,9,7
9,10,8,6,7
输出：-1
说明：第一个评委给第一个选手打分11，无效分数

```java
import java.util.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNextLine()) { // 注意 while 处理多个 case
            String s = in.nextLine();
            String[] s1 = s.split(",");
            if (s1.length != 2) {
                System.out.println(-1);
                return;
            }
            //评委数量
            int m = Integer.parseInt(s1[0]);
            //选手数量
            int n = Integer.parseInt(s1[1]);
            if (m < 3 || n < 3 || m > 10 || n > 100) {
                System.out.println(-1);
                return;
            }
            List<Athlete> list = new ArrayList<>();
            Map<Integer, Athlete> map = new HashMap<>();
            for (int i = 1; i < n + 1; i++) {
                Athlete athlete = new Athlete();
                athlete.num = i;
                athlete.total = 0;
                athlete.scores  = new ArrayList<>();
                map.put(i, athlete);
                list.add(athlete);
            }

            for (int i = 0; i < m; i++) {
```

```
            String a = in.nextLine();
            String[] a1 = a.split(",");
            for (int j = 1; j <= a1.length; j++) {
                int score = Integer.parseInt(a1[j - 1]);
                if (score < 1 || score > 10) {
                    System.out.println(-1);
                    return;
                }
                map.get(j).scores.add(score);
                map.get(j).total += score;
                map.get(j).count[score]++;
            }
        }
        //System.out.println(list);
        list.sort(new Comparator<Athlete>() {
            @Override
            public int compare(Athlete o1, Athlete o2) {
                if (o1.total != o2.total) {
                    return o2.total - o1.total;
                } else {
                    for (int i = 10; i >= 1; i--) {
                        if (o1.count[i] == o2.count[i]) {
                            continue;
                        } else {
                            return o2.count[i] - o1.count[i];
                        }
                    }
                    return 0;
                }
            }
        });
        System.out.println(list.get(0).num + "," + list.get(1).num + "," + list.get(
                        2).num);
    }
}

    static class Athlete {
        public int num;
        public int total;
        public List<Integer> scores;
        public int[] count = new int[11];
    }
}
```