字母组合

知识点 回溯
时间限制：1s 空间限制：256MB 限定语言：不限

题目描述：

每个数字对应多个字母，对应关系如下：
0：a,b,c　1：d,e,f　2：g,h,i　3：j,k,l　4：m,n,o　5：p,q,r　6：s,t　7：u,v　8：w,x　9：y, z

输入一串数字后，通过数字和字母的对应关系可以得到多个字母字符串（要求按照数字的顺序组合字母字符串）；

屏蔽字符： 屏蔽字符中的所有字母不能同时在输出的字符串出现，如屏蔽字符时abc，则要求字符串中不能同时出现a，b，c，但是允许同时出现a，b；a，c；b，c等；

给定一个数字字符串和一个屏蔽字符串，输出所有可能的字符组合；

例如输入数字字符串78和屏蔽字符串ux，输出结果为uw，vw，vx；数字字符串78，可以得到如下字符串： uw，ux，vw，vx；由于ux是屏蔽字符串，因此排除ux，最终的输出时uw，vw，vx；

输入描述：

第一行输入为一串数字字符串，数字字符串中的数字不允许重复，数字字符串的长度大于0，小于等于5；
第二行输入是屏蔽字符，屏蔽字符的长度一定小于数字字符串的长度，屏蔽字符串中字符不会重复，

输出描述：

输出可能的字符串组合
注：字符串之间使用逗号隔开，最后一个字符串后携带逗号

示例1

输入：

78
ux

输出：

uw,vw,vx,

示例2

输入：

78
x

输出：

uw,vw,

解题思路：

回溯法求出所有可能的字母组合，并筛选出符合要求的（去除不能出现的字符）

```
public class Main{

    public static String noContain;
    public static StringBuffer res = new StringBuffer();
    public static char[][] digits = {
                                                {'a','b','c'},
                                                {'d','e','f'},
                                                {'g','h','i'},
                                                {'j','k','l'},
                                                {'m','n','o'},
                                                {'p','q','r'},
                                                {'s','t'},
                                                {'u','v'},
                                                {'w','x'},
                                                {'y','z'}
                                            };

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
```

```
        String numStr = sc.nextLine();
        noContain = sc.nextLine();          //不包含的字符串

        handle(0, numStr, "");

        System.out.println(res);

    }

    /**
     * 回溯法求出所有可能的字母组合，并筛选出符合要求的（去除不能出现的字符）
     * @param index         字母索引
     * @param numStr        输入的数字字符串
     * @param temp          字母组合
     */
    public static void handle(int index, String numStr, String temp){

        if(temp.length() == numStr.length()){     //字符串的长度等于数字字符串的长度说明已经遍历完成
            if(!temp.contains(noContain)){          //字符串不包含 noContain 的进行拼接
                res.append(temp + ",");
            }
        }else {

            int digitsIndex = numStr.charAt(index) - '0';     //字母的索引
            char[] chars = digits[digitsIndex];
            for( int i=0; i<chars.length; i++){
                handle(index + 1, numStr, temp + chars[i]);
            }

        }
    }

}
```