题目描述：

为了提升数据传输的效率，会对传输的报文进行压缩处理。输入一个压缩后的报文，请返回它解压后的原始报文。

压缩规则：n[str]，表示方括号内部的 str 正好重复 n 次。注意 n 为正整数（0 < n <= 100），str 只包含小写英文字母，不考虑异常情况。

输入描述：

输入压缩后的报文：

1）不考虑无效的输入，报文没有额外的空格，方括号总是符合格式要求的；

2）原始报文不包含数字，所有的数字只表示重复的次数 n ，例如不会出现像 5b 或 3[8] 的输入；

输出描述：

解压后的原始报文

注：

1）原始报文长度不会超过 1000，不考虑异常的情况

示例 1

输入：

```
3[k]2[mn]
```

输出：

```
kkkmnmn
```

说明：

k 重复 3 次，mn 重复 2 次，最终得到 kkkmnmn

示例 2

输入：

```
3[m2[c]]
```

输出：

```
mccmccmcc
```

说明：

m2[c] 解压缩后为 mcc，重复三次为 mccmccmcc

```
import java.util.Scanner;
import java.util.Stack;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String str = scanner.nextLine();

        Stack<Character> stack = new Stack<>();



        for(int i = 0 ; i < str.length(); i++){
            char c = str.charAt(i);
            if(c == ']'){
                StringBuilder temp = new StringBuilder();
                while(!stack.isEmpty() && '[' != stack.peek()){
                    temp.insert(0, stack.pop());
                }
                //遇到了"["
                stack.pop();

                //找数字
                String nums = "";
                while (!stack.isEmpty() && (stack.peek() >= '0' && stack.peek() <= '9')){
                    nums = stack.pop() + nums;
                }

                int times = Integer.parseInt(nums);
                for(int k = 0 ; k < times; k++){
                    //插入到栈中
                    for(int j = 0 ; j < temp.length(); j++){
                        stack.add(temp.charAt(j));
                    }
```

```
                }

            }else{
                stack.add(c);
            }
        }

        StringBuilder res = new StringBuilder();
        while(!stack.isEmpty() ){
            res.insert(0, stack.pop());
        }
        System.out.println(res);
    }
}
```