解压报文题目描述：

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

补充说明：

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.*;

// 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.hasNextInt()) { // 注意 while 处理多个 case
        //     int a = in.nextInt();
        //     int b = in.nextInt();
        //     System.out.println(a + b);
        // }
        String str = in.nextLine();
        String s = decompress(str);
        System.out.println(s);
    }

    public static String decompress(String compressedString) {
        Stack<Integer> countStack = new Stack<>();
    Stack<String> strStack = new Stack<>();
    StringBuilder currentString = new StringBuilder();
    int count = 0;
    for (char c : compressedString.toCharArray()) {
        if (Character.isDigit(c)) {
            count = count * 10 + (c - '0');
        } else if (c == '[') {
            countStack.push(count);
            strStack.push(currentString.toString());
            currentString = new StringBuilder();
            count = 0;
        } else if (c == ']') {
```

```
                int repeatTimes = countStack.pop();
                StringBuilder temp = new StringBuilder(strStack.pop());
                for (int i = 0; i < repeatTimes; i++) {
                    temp.append(currentString);
                }
                currentString = temp;
            } else {
                currentString.append(c);
            }
        }
        return currentString.toString();
    }
}
```