

玩牌高手

题目描述：

给定一个长度为 n 的整型数组，表示一个选手在 n 轮内可选择的牌面分数。选手基于规则选牌，请计算所有轮结束后其可以获得的最高总分数。选择规则如下：

1、在每轮里选手可以选择获取该轮牌面，则其总分数加上该轮牌面分数，为其新的总分数。

2、选手也可不选择本轮牌面直接跳到下一轮，此时将当前总分数还原为 3 轮前的总分数，

若当前轮次小于等于 3（即在第 1、2、3 轮选择跳过轮次），则总分数置为 0。

3、选手的初始总分数为 0，且必须依次参加每一轮。

输入描述：

第一行为一个小写逗号分割的字符串，表示 n 轮的牌面分数， $1 \leq n \leq 20$ 。

分数值为整数， $-100 \leq \text{分数值} \leq 100$ 。

不考虑格式问题。

输出描述：

所有轮结束后选手获得的最高总分数。

示例 1

输入：

1, -5, -6, 4, 3, 6, -2

输出：

11

说明：

总共有 7 轮牌面。

第一轮选择该轮牌面，总分数为 1。

第二轮不选择该轮牌面，总分数还原为 0。

第三轮不选择该轮牌面，总分数还原为 0。

第四轮选择该轮牌面，总分数为 **4**。

第五轮选择该轮牌面，总分数为 **7**。

第六轮选择该轮牌面，总分数为 **13**。

第七轮如果不选择该轮牌面，则总分数还原到 **3** 轮 **1** 前分数，即第四轮的总分数 **4**，如果选择该轮牌面，总分数为 **11**，所以选择该轮牌面。

因此，最终的最高总分为 **11**。

```
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);
```

```
        String str = in.nextLine();
```

```
        String [] strs = str.split(",");
```

```
        int length = strs.length;
```

```
        Map<Integer, Integer> map = new HashMap();
```

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

```
            int value = Integer.valueOf(strs[i]);
```

```
            if (value >= 0) {
```

```
                int nowValue = i == 0 ? value : map.get(i - 1) + value;
```

```
                int lastThreeValue = i - 3 < 0 ? 0 : map.get(i - 3);
```

```
                if (nowValue >= lastThreeValue) {
```

```
                    map.put(i, nowValue);
```

```
                } else {
```

```
                    map.put(i, lastThreeValue);
```

```
                }
```

```
            } else {
```

```
                if (i - 3 < 0) {
```

```
                    int nowValue = i == 0 ? value : map.get(i - 1) + value;
```

```
                    if (nowValue >= 0) {
```

```
                        map.put(i, nowValue);
```

```
                    } else {
```

```
                        map.put(i, 0);
```

```
                    }
```

```
        } else {  
            int lastThreeValue = map.getDefault(i - 3, 0);  
            int nowValue = value + map.get(i - 1);  
            if (nowValue >= lastThreeValue) {  
                map.put(i, nowValue);  
            } else {  
                map.put(i, lastThreeValue);  
            }  
        }
```

```
    }  
}  
}  
System.out.println(map.get(length - 1));  
}  
}
```