面向对象程序设计 lab4 实验报告

实验内容：

本次 lab 是在 lab2，lab3 的基础上完整实现的扫雷小游戏。
Lab2 实现了扫雷地图的生成及对于地图的打印。
Lab3 实现了扫雷的具体操作的实现及和用户交互的要求。
Lab4 对 lab2 和 lab3 的代码进行了大规模的重构。

1. 完善了扫雷的具体功能即“当点击的位置周边地雷数量为零时，可以自动扩大显示的范围”。
2. 增加了右键标记的选项实现对于地雷位置的标记显示。
3. 增加了在不退出程序的前提下再次开始下一局扫雷的功能

实现思路：

本 lab 由 Main.java，PrintMap.java，CommmandControl.java，SweepMine.java，MakeMap.java 五个文件组成，各自包含一个类，实现相对应的功能，实现及具体代码解析在各函数具体实现部分详解。
Main class 是整个项目的起始，按照顺序调用各个类及具体函数实现扫雷的功能。
CommandControl class 实现控制台与用户的交互功能，并定义类实例变量存储关于地雷地图的相关信息。
MakeMap class 实现对于指定大小和地雷数量的地图的产生与初始化。
SweepMine class 实现一次命令执行扫雷的操作。
PrintMap class 实现对于地图的不同打印方式。

主要变量意义解释：

int row:用于存储地图的行数。
int column:用于存储地图的列数。
int num:用于存储地图中地雷的数量。
char [][]innerMap:存储 0-8 和*这十个字符，分别表示周围地雷的数量以及地雷本身。
Boolean [][] outerMap:表示各位置是否已经被用户翻开，true 表示没有被用户翻开，false 表示已经被用户翻开，此位置对用户可见。
Boolean [][] markMap:用于实现用户的右键标记的功能，判断各个位置是否被用户标记，true 表示已经被用户标记为地雷，false 表示此位置没有被用户标记为地雷。
Boolean [][] visited:：用于实现当点击的位置周边地雷数量为零时，可以自动扩大显示的范围的功能，表示在递归遍历时附近的某位置是否已经被遍历过。true 表示已经遍历过，false 表示没有遍历过。

各函数具体实现

CommandControl class：

1. **isSweepOut 方法：**
   - 功能：遍历地图，如果发现有非地雷方块未被揭示，则游戏未结束，返回 false；否则返回 true，表示游戏结束。

```
public static boolean isSweepOut(char
[][]innerMap,boolean[][]outerMap)
{
    for(int i=0;i<innerMap.length;i++)
    {
        for(int j=0;j<innerMap[0].length;j++)
        {
            if(innerMap[i][j]!='*'&&outerMap[i][j])
                return false;
        }
    }
    return true;
}
```

2. **oneGame 方法：**
   - 功能：通过循环进行游戏直到游戏结束。在每一轮中，根据用户输入的操作执行相应的动作，如排雷或标记地雷，然后根据游戏状态判断是否继续游戏。

```
public void oneGame(MakeMap makeMap)
{
    int flag=1;
    while (!CommandControl.isSweepOut(makeMap.innerMap,
makeMap.outerMap)) {
        System.out.println("左键排雷则输入 1，右键标记则输入 2");
        int judgeNum = scn.nextInt();
        System.out.printf("请输入 1-%d 1-%d 来确定要操作的坐标\n",
makeMap.row, makeMap.column);
        int inputRow = scn.nextInt();  int inputColumn =
scn.nextInt();
        if (judgeNum == 1)
        {
            if (SweepMine.SweepOneMine(makeMap.innerMap,
makeMap.outerMap, makeMap.markMap, makeMap.visited,inputRow-
1, inputColumn - 1))
            {
                System.out.println("踩中地雷，游戏结束");
                PrintMap.printMap(makeMap.innerMap);
                flag = 0;
```

```
                break;
            }
            else
            {
                PrintMap.printMap(makeMap.innerMap,
makeMap.outerMap, makeMap.markMap);
            }
        }
        else if(judgeNum==2)
        {
            SweepMine.markOneMine(makeMap.markMap,
inputRow,inputColumn);
            PrintMap.printMap(makeMap.innerMap,
makeMap.outerMap, makeMap.markMap);
        }
        else
        {

        }
    }
    if (flag == 1)
        System.out.println("恭喜，扫雷完成");
}
```

3. **inputData 方法：**
    - 功能：通过 Scanner 类从控制台获取用户输入的行数、列数和地雷数量。

```
public void inputData()
{

    scn=new Scanner(System.in);
    System.out.println("请依次输入行、列以及地雷数”);
    row=scn.nextInt();
    column=scn.nextInt();
    num=scn.nextInt();
}
```

4. **continueOrNot 方法**：
    - 功能：从控制台获取用户输入的整数，如果输入不为 0，则表示用户想要继续游戏，返回 true；否则返回 false，表示用户不想继续游戏。

```
public boolean continueOrNot()
{
    System.out.println("是否继续游戏（1/0)");
    int input=scn.nextInt();
```

```
    return input != 0;
}
```

-

MakeMap class：

1. **构造函数 MakeMap(CommandControl cmd)：**
   - 功能：将 MakeMap 类的行数、列数和地雷数量初始化为传入 CommandControl 对象的相应属性值。

```
public MakeMap(CommandControl cmd)
{
    this.row=cmd.row;
    this.column=cmd.column;
    this.num=cmd.num;
}
```

2. **makeMap 方法：**
   - 功能：利用 Random 类生成随机坐标，将地雷随机放置在地图上，然后遍历地图中每个非地雷方块，计算周围的地雷数量，并将其填入地图中。

```
public void makeMap()
{

    Random generator=new Random();
    for(int i=0;i<num;)
    {
        int m1=generator.nextInt(innerMap.length),m2=
generator.nextInt(innerMap[0].length);
        if(innerMap[m1][m2]!='*')
        {
            innerMap[m1][m2]='*';
            i++;
        }
    }
    for(int i=0;i<innerMap.length;i++)
    {
        for(int j=0;j<innerMap[0].length;j++)
        {
            int val=0;
            if(innerMap[i][j]!='*')
            {
                for(int k=Math.max(i-
1,0);k<=Math.min(i+1,innerMap.length-1);k++){
                    for(int t=Math.max(j-
1,0);t<=Math.min(j+1,innerMap[0].length-1);t++){
                        if(innerMap[k][t]=='*'){
```

```
                        val++;
                    }
                }
            }
            innerMap[i][j]=(char)(val+'0');
        }
    }
}
```

3. **initBoolMap 方法：**
    - 功能：根据地图的行数和列数初始化 innerMap、outerMap、markMap 和 visited 四个布尔数组，并分别将其初始值设置为表示全部被覆盖中、未标记地雷、未访问状态。

```
public  void initBoolMap()
{
    innerMap=new char[row][column];
    outerMap=new boolean[row][column];
    markMap=new boolean[row][column];
    visited=new boolean[row][column];
    for (boolean[] booleans : outerMap)
    {
        Arrays.fill(booleans, true);//表示全部被覆盖中
    }
    for(boolean[]booleans:markMap)
    {
        Arrays.fill(booleans,false);
    }
    for(boolean []booleans:visited)
    {
        Arrays.fill(booleans,false);
    }
}
```

SweepMine class：

1. **SweepOneMine 方法：**
    - 功能：首先检查指定位置是否为地雷，如果是地雷则游戏失败，返回 true；如果不是地雷，则调用 subSweep 方法揭示该位置及其周围方块，返回 false 表示游戏继续进行。

```
public static boolean SweepOneMine(char
[][]innerMap,boolean[][]outerMap,boolean[][]markMap,boolean
[][]visited,int row,int column)
{
    //未防止重复点击、越界等的输入
```

```
    if(innerMap[row][column]=='*')
    {
        return true;
    }
    else
    {
        subSweep(innerMap,outerMap,markMap,visited,row,column);
        return false;
    }
}//true----bomb,false----continue
```

2. **subSweep 方法：**
    - 功能：首先检查是否为标记的地雷，如果是则返回；然后将当前位置的外部地图状态设置为未覆盖，并标记为已访问。接着检查当前位置的数字，如果为 0，则递归调用 subSweep 方法揭示周围的方块。

```
public static void
subSweep(char[][]innerMap,boolean[][]outerMap,boolean[][]markM
ap,boolean [][]visited,int row,int column)
{
    if(markMap[row][column])
    {
        return;
    }

    if(innerMap[row][column]!='*')
    {
        outerMap[row][column]=false;
        visited[row][column]=true;
    }
    else return;
    if(innerMap[row][column]=='0')
    {
        if(row>0&&!visited[row-1][column])
            subSweep(innerMap,outerMap,markMap,visited,row-
1,column);
        if(row<innerMap.length-1&&!visited[row+1][column])

subSweep(innerMap,outerMap,markMap,visited,row+1,column);
        if(column>0&&!visited[row][column-1])

subSweep(innerMap,outerMap,markMap,visited,row,column-1);
        if(column<innerMap[0].length-
1&&!visited[row][column+1])
```

```
subSweep(innerMap,outerMap,markMap,visited,row,column+1);
        if(row<innerMap.length-
1&&column>0&&!visited[row+1][column-1])

subSweep(innerMap,outerMap,markMap,visited,row+1,column-1);
        if(row>0&&column<innerMap[0].length-1&&!visited[row-
1][column+1])
            subSweep(innerMap,outerMap,markMap,visited,row-
1,column+1);
        if(row>0&&column>0&&!visited[row-1][column-1])
            subSweep(innerMap, outerMap, markMap, visited, row-
1, column-1);
        if(row< innerMap.length-1&&column<innerMap[0].length-
1&&!visited[row+1][column+1])
            subSweep(innerMap, outerMap, markMap, visited,
row+1, column+1);
    }
}
```

3. markOneMine 方法：
    - 功能：根据传入的坐标位置将对应位置的标记状态取反，即标记为地雷或取消地雷标记。

```
public static void markOneMine(boolean[][]markMap,int row,int
column)
{
    markMap[row][column]= !markMap[row][column];

}
```

PrintMap class：

1. printMap(char[][] a, boolean[][] b, boolean[][] c) 方法：
    - 功能：遍历内部地图 a，对于每个位置，如果 c[i][j] 为 true，则打印 "$ "，表示标记的地雷；如果 b[i][j] 为 true，则打印 "# "，表示未揭示的区域；否则打印 a[i][j] 的值，即数字或空格，表示揭示的区域。

```
public static void printMap(char[][]a,boolean [][]b,boolean
[][]c)
{
    for(int i=0;i<a.length;i++)
    {
        for(int j=0;j<a[0].length;j++)
        {
            if(c[i][j])
```

```
            {
                System.out.println("$ ");
                continue;
            }
            if(b[i][j])
            {
                System.out.print("# ");
            }
            else
            {
                System.out.print(a[i][j]+" ");
            }

        }
        System.out.println();
    }

}
```

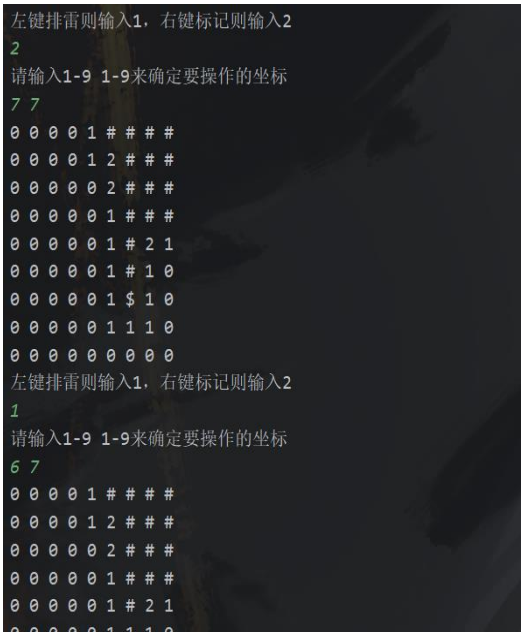
2. printMap(char[][] a) 方法：
    - 功能：遍历内部地图 a，对于每一行直接打印该行的内容，每个元素之间用空格分隔，行与行之间换行输出。

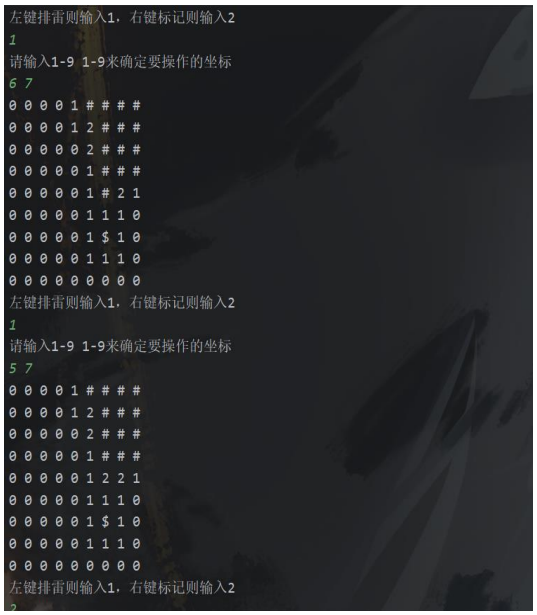
```
public static void printMap(char[][]a)
{
    for(char[]arr:a)
    {
        for(char i:arr)
        {
            System.out.print(i+" ");
        }
        System.out.println();
    }

}
```

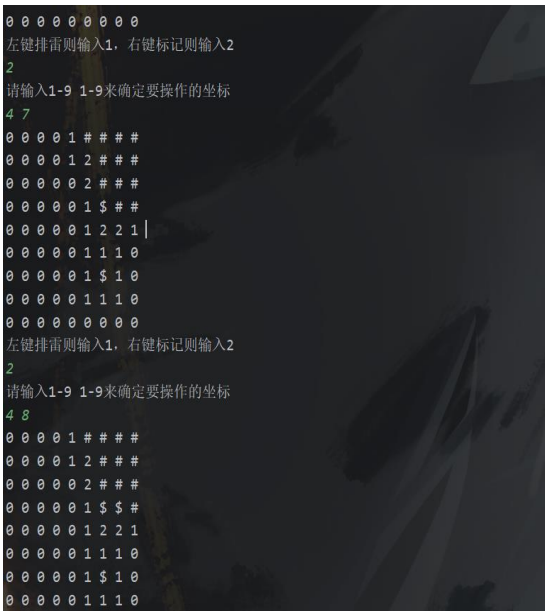
具体运行效果：

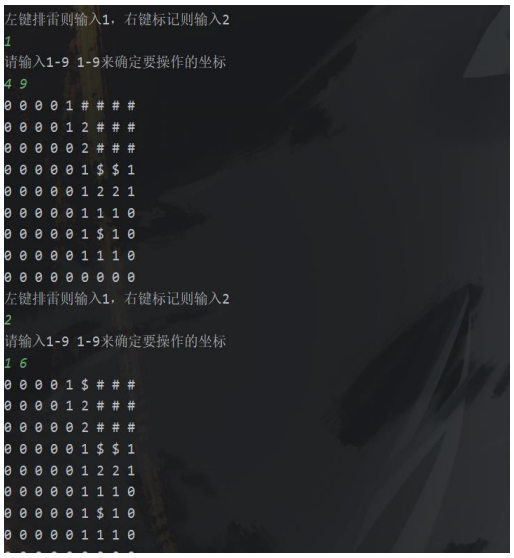
下根据各截图来解释运行的具体情况。
首先打印出提示输入地图初始化的相关信息，初始化地图。然后提示用户下一步是进行排雷操作还是进行标记操作，最后输入要操作的坐标。在用户输入完毕后，紧接着打印出用户端可见的地图样式。具体体现看下面几幅图片即可

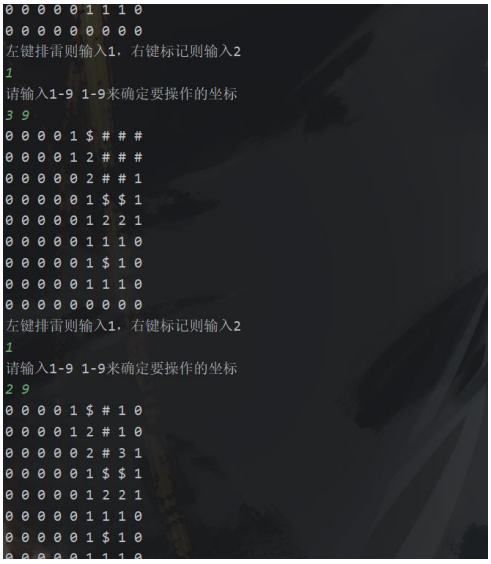

请依次输入行、列以及地雷数
9 9 5
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
# # # # # # # # #
左键排雷则输入1，右键标记则输入2
1
请输入1-9 1-9来确定要操作的坐标
5 5
0 0 0 0 1 # # # #
0 0 0 0 1 2 # # #
0 0 0 0 0 2 # # #
0 0 0 0 0 1 # # #
0 0 0 0 0 1 # 2 1
0 0 0 0 0 1 # 1 0
0 0 0 0 0 1 # 1 0
0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0

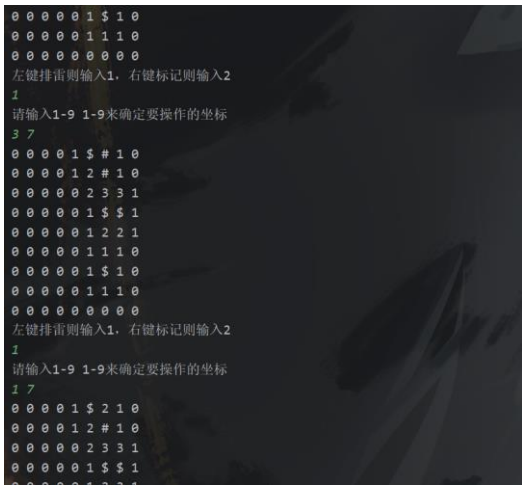

左键排雷则输入1，右键标记则输入2
2
请输入1-9 1-9来确定要操作的坐标
7 7
0 0 0 0 1 # # # #
0 0 0 0 1 2 # # #
0 0 0 0 0 2 # # #
0 0 0 0 0 1 # # #
0 0 0 0 0 1 # 2 1
0 0 0 0 0 1 # 1 0
0 0 0 0 0 1 $ 1 0
0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0
左键排雷则输入1，右键标记则输入2
1
请输入1-9 1-9来确定要操作的坐标
6 7
0 0 0 0 1 # # # #
0 0 0 0 1 2 # # #
0 0 0 0 0 2 # # #
0 0 0 0 0 1 # # #
0 0 0 0 0 1 # 2 1

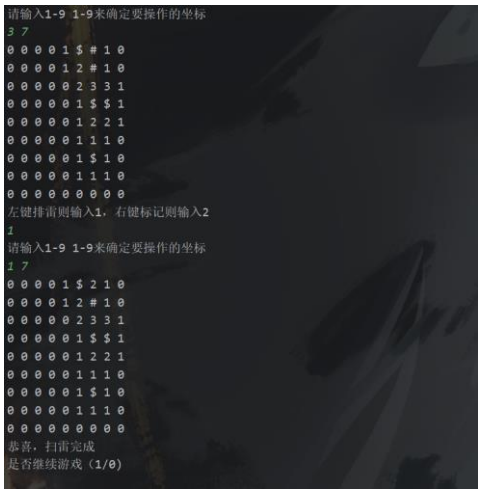

左键排雷则输入1，右键标记则输入2
1
请输入1-9 1-9来确定要操作的坐标
6 7
0 0 0 0 1 # # # #
0 0 0 0 1 2 # # #
0 0 0 0 0 2 # # #
0 0 0 0 0 1 # # #
0 0 0 0 0 1 # 2 1
0 0 0 0 0 1 1 1 0
0 0 0 0 0 1 $ 1 0
0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0
左键排雷则输入1，右键标记则输入2
1
请输入1-9 1-9来确定要操作的坐标
5 7
0 0 0 0 1 # # # #
0 0 0 0 1 2 # # #
0 0 0 0 0 2 # # #
0 0 0 0 0 1 # # #
0 0 0 0 0 1 2 2 1
0 0 0 0 0 1 1 1 0
0 0 0 0 0 1 $ 1 0
0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0
左键排雷则输入1，右键标记则输入2


0 0 0 0 0 0 0 0 0
左键排雷则输入1，右键标记则输入2
2
请输入1-9 1-9来确定要操作的坐标
4 7
0 0 0 0 1 # # # #
0 0 0 0 1 2 # # #
0 0 0 0 0 2 # # #
0 0 0 0 0 1 $ # #
0 0 0 0 0 1 2 2 1
0 0 0 0 0 1 1 1 0
0 0 0 0 0 1 $ 1 0
0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0
左键排雷则输入1，右键标记则输入2
2
请输入1-9 1-9来确定要操作的坐标
4 8
0 0 0 0 1 # # # #
0 0 0 0 1 2 # # #
0 0 0 0 0 2 # # #
0 0 0 0 0 1 $ $ #
0 0 0 0 0 1 2 2 1
0 0 0 0 0 1 1 1 0
0 0 0 0 0 1 $ 1 0
0 0 0 0 0 1 1 1 0



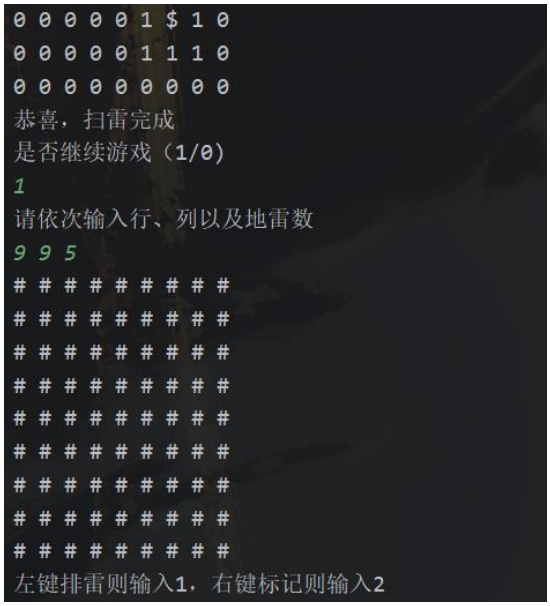


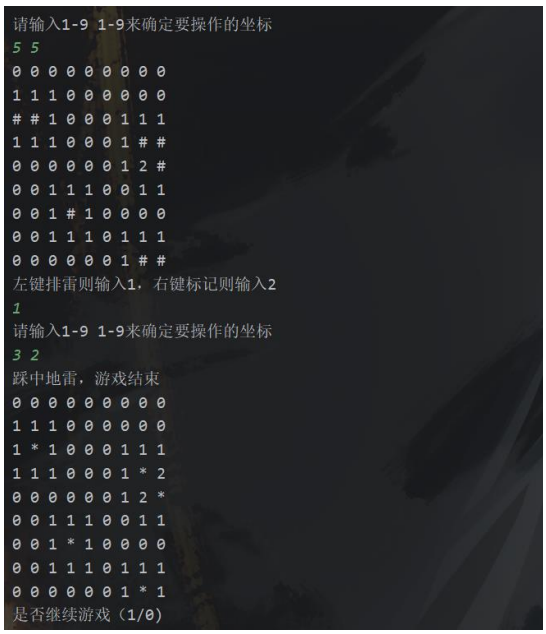




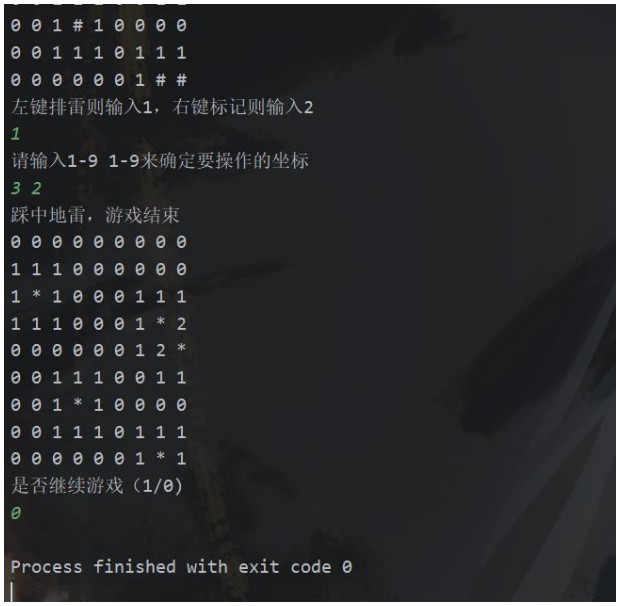
在检测到剩下没有被翻开的全部为地雷后，程序提示“恭喜，扫雷完成”，表明已经完成游戏，然后提示“是否继续游戏”输入 1 即为继续下一盘游戏，0 为退出游戏。

选择继续游戏后，下面展示扫雷失败的图示





当碰到地雷时，会提示“踩中地雷，游戏结束”的提示语，并且打印出实际的无遮挡的地雷图，再次提示是否继续游戏。



输入 0，游戏结束，程序结束。

Lab 总结：

1. 遇到的难点及解决方案

A．在生成地图时，为判断每个位置的周围地雷的数量，需要遍历周边，最初忽略边界情况导致数组越界。解决：加设判断条件挑出边界情况单独分析，防止越界。

B．在扫雷的实际功能实现时，需要进行对于所点击的点周围点的迭代判断，最初无从下手。解决：使用递归函数，反复递归值为‘0’的点的周围，实现功能。

C．在重构代码时，经常出现变量和方法有效作用域导致的无法在其他类或方法中访问的问题。解决：调整好变量定义时的类型，或者将变量作为类的实例变量，使得该类下的方法可以访问该变量。

2. 实验收获

1. 在这几个 lab 中，从最简单的生成地图开始，逐渐增加功能，最终实现了扫雷这个小游戏，从这个逐步实现的过程中，我初步体会到了一个小的项目开发的模式，从一个又一个功能的叠加，到最后的重构与整合，完成了这个小项目的开发实践。
2. 使用 java 进行开发，更加熟练的掌握了 java 这门语言的语法特性，并且在最后的重构整合过程中体会到了 Java 面向对象和 c 面向过程的不同，初步学习面向对象的语言特性。
3. 在最后 lab4 优化代码时，充分体会到最初接触这个 lab 时所写的代码的粗糙，自身去优化代码，并且在这个优化的过程中提高了自己的编码能力。