Lab4 扫雷高级版本

1. 主要功能实现
2. 恶意输入防范
3. 程序开始时对地雷图数据的恶意输入防范
4. 总述

创建Function类中的一个对象newGame。提示用户输入地雷图数据,不合法时提示用户并让其重新输入,用do-while结构,循环条件为(!newGame.isValidLoad),正确输入地雷图数据时结束循环。

b.正确格式

isValidLoad赋初值为true。用正则表达式判断,若不对,提示用户”Not right form,try again!”,将isValidLoad赋值为false,继续循环,直到用户输入正确数据。

文本

描述已自动生成

(左图展示未按照正确格式输入地雷图数据的效果)

c.合理数值

数值不合理的情况分为两种:行列设置为0或过大,炸弹数过多或为0。用if语句分别讨论,满足其中一种情况,则将isValidLoad赋值为false。继续循环,直到用户输入正确数据。

文本

描述已自动生成(左图显示输入不合理地雷图数据的效果)

2.实时打印标记地雷数量

正确读入地雷图数据后,依次调用Function类中createArray和printGameInterface方法打印游戏界面。printGameInterface方法中输出一行语句提示剩余的标记次数,剩余标记次数由变量leftBomb记录,该变量在load中赋初值为bombs。

\*注意:leftBomb如何记录剩余标记数量,3(3)c中讨论。

3.游戏中各类命令实现总述

(1)循环设置

进行游戏时需要输入多次命令,故采用while循环进行游戏过程,循环条件(!newGame.isVictory & !newGame.isFail & ! newGame.startOver)。在循环开始处用nextLine读入字符串输入并赋给字符串变量command,commandType方法判断命令类型。

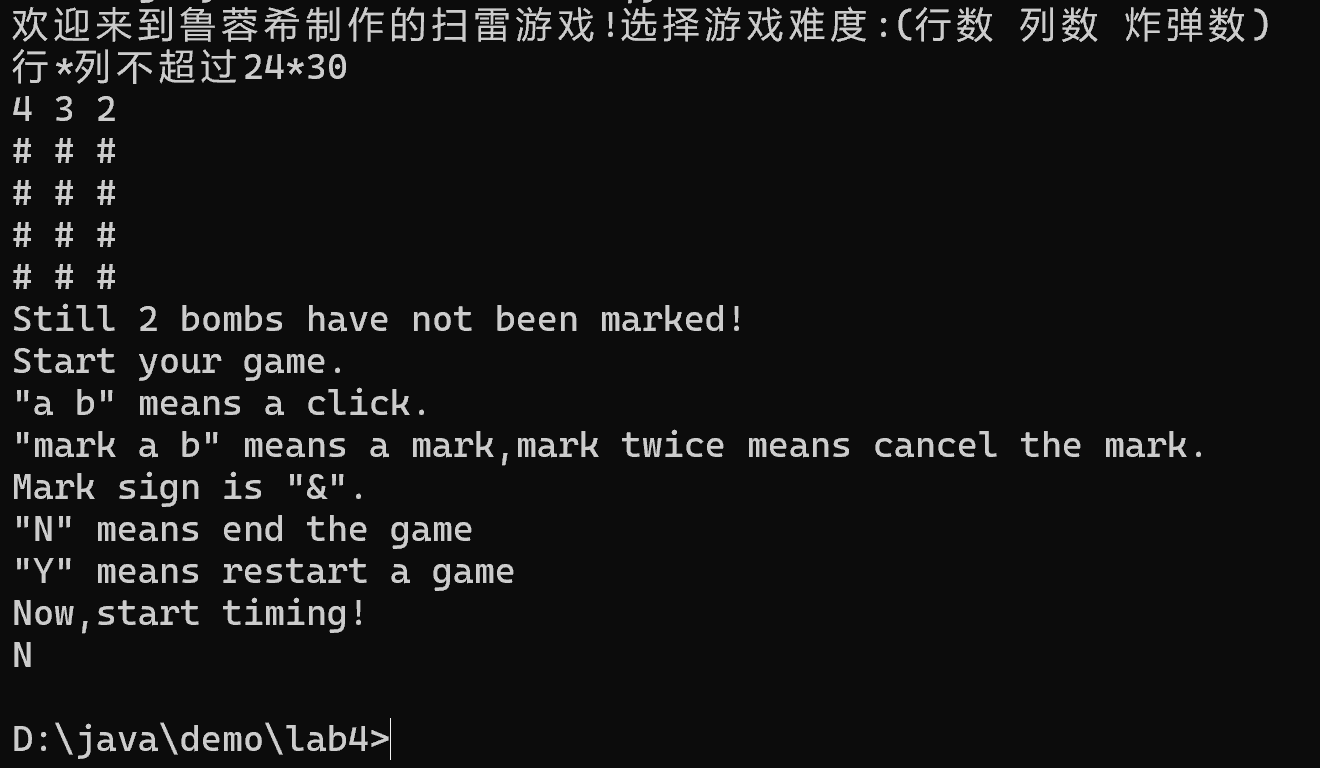
\*注意:关于变量isVictory,isFail,startOver如何改变将在后文展开。

(2)游戏重载或结束

生成地雷图后,命令”Y”重新开始,命令”N”结束程序。commandType方法用if语句判断是否为Y/N命令。若是,调用endOrStartOver方法。

\*注意:命令Y/N可以在游戏中和游戏结束后输入。此处讨论在游戏中输入的情况, endOrStartOver方法细节在后文展开。

文本

描述已自动生成

(左图展示游戏中输入Y重新开始的效果,右图展示游戏中输入N 结束程序的效果)

(3)标记

a.标记口令及符号

口令形式为”mark 整数 整数”,用”&”符号在地图中表示被标记的位置。

b.识别命令并判断是否合法

commanType方法中startsWith(“mark ”)识别标记命令,若是,调用mark方法。mark方法用正则表达式判断格式,if语句判断是否合理(标记点不在地图内,标记已打开的坐标和标记次数已用完仍尝试标记未标记坐标)。若不合法,根据情况给出提示,并重新读取命令。

文本

描述已自动生成(左图分别展示2种不合格式的标记,不在地图内的标记,已打开的标记和标记次数用完但继续标记的效果)

c.标记与取消标记

用If语句分为未标记和已标记2种情况。未曾标记则更新符号为&,剩余标记数leftBomb减一,继续判断是否为地雷,若是,rightMark加一。已标记则更新符号为#,剩余标记数leftBomb加一,继续判断是否为地雷,若是,rightMark减一。最后, 调用printGameInterface打印。赋值isVictory为i == rightClick | bombs == rightMark。

文本

描述已自动生成(左图运行结果展示取消标记的功能)

(4)点击

a.识别命令并判断是否合法

commandType方法中输入均不属于以上命令时,调用click方法。首先正则表达式判断格式,然后判断值是否合理。若不合理,提示用户并让其重新输入。

(下图运行效果分别展示输入2种不合格式的点击,地图外的点击和已打开处的点击的结果)

文本

描述已自动生成

1. 开局踩雷不会导致游戏失败

正确输入点击命令后, 用if-else语句分为invincibility是否为真两部分。为真则提示用户”This is a bomb!”并标记,赋值invincibility为false,若不为,打印地雷图并提示游戏失败。

(下图展示开局踩雷不会导致游戏失败,而是自动标记,之后踩雷将导致游戏失败)

文本

描述已自动生成

c.递归打开

click中坐标周围地雷数为0时,揭开该处,并调用open方法。open方法首先遍历以目标点为中心的九宫格,过程中,揭开未被打开的位置,i加一,若某位置同样为0,用open方法递归打开。

文本

描述已自动生成

4.重载与结束

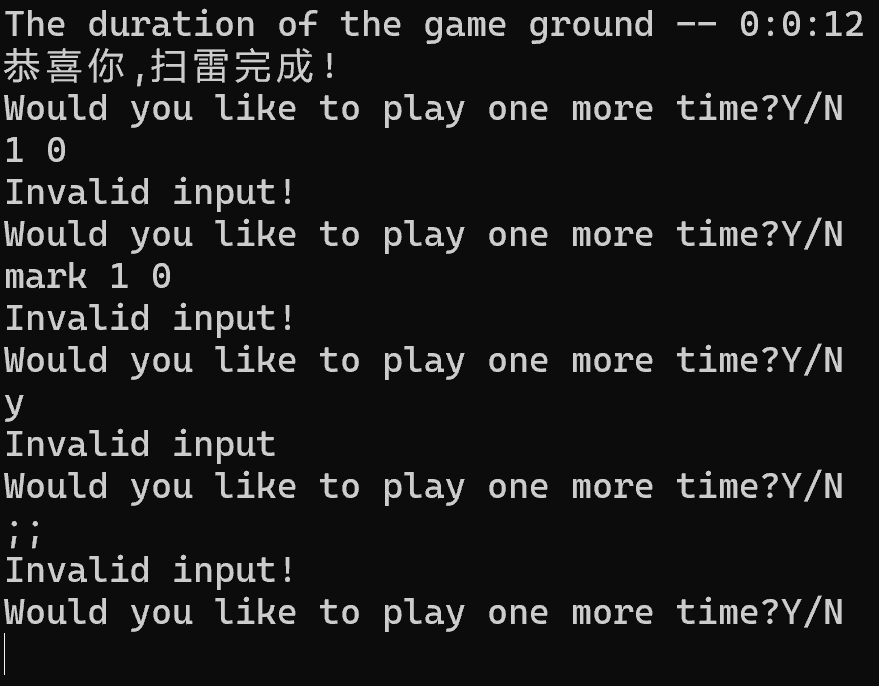
startOver初值为false,endOrStartOver方法中先用正则表达式判断是否为一个字符,再判断若是N命令,System.exit()结束程序,Y命令,则赋值startOver为true。游戏未使用Y/N命令而正常结束时,询问用户”Would you like to play one more game?Y/N”,while结构用于用户重复输入,调用endOrStsrtOver方法,循环条件为!startOver。

文本

描述已自动生成文本

描述已自动生成

(右图为游戏结束后输入Y重新开始,左图为游戏结束后输入N结束程序,下图为输入4种不正确指令)



5.计时器

在生成地雷图后,System.currentTimeMillis()赋值startTime,代表游戏中的循环体结束后, System.currentTimeMillis()赋值endTime。调用timer方法显示本轮用时。

1. 创新及细节
2. 首选地雷

Lab4实验任务1中恶意输入防范可以保证用户首次选择位置不是地雷,意即用户首次输入坐标时不能输入地雷坐标。如此,当用户发现某个坐标在首次不能输入时,便能理解到此处为地雷。因此,对该任务要求做创新,用户可以在首次输入时输入地雷坐标,但系统会提示此处是地雷,并标记。

2.扫雷成功结束游戏条件

Lab3实验要求中规定当用户揭开所有非地雷位置后,提示用户”恭喜你,扫雷完成!”。Lab4加入标记功能后,对扫雷成功的条件判断除揭开所有非地雷位置以外,还可为正确标记所有地雷,当然,游戏给予用户的总标记次数与本轮的炸弹数相等。

3.重载与结束

Lab4实验要求5中要求本轮游戏结束后,可以在不退出程序的情况下,选择新游戏或结束程序。创新为用户不仅可以在本轮游戏结束后如此选择,在生成地雷图后的游戏进行中也可以随时选择重新开始或结束程序。

\*注意:最早也要生成地雷图后才可以用Y/N选择重新开始或结束,在输入地雷图数据时不能。

1. 计时器

一轮游戏结束后显示该局时长。

1. 对象和类

据实验要求定义一个Function类,并创建一个newGame对象。以下为UML类图。

Function

- rows : int

- cols : int

- bombs : int

- rightClick : int

- rightMark : int

- leftBomb : int

- i : int

-startTime : long

-endTime : long

- bombLable : char[][]

- state : boolean[][]

- gameInterface : char[][]

- invincibility : boolean

+ isValidLoad : boolean

+ startOver : boolean

+ isVictory : boolean

+ isFail : boolean

+Function()

+load(s : String):void

+creatArray():void

-printBombLable():void

+printGameInterface():void

+commandType(s : String):void

-click(s:String):void

-open(r:int,c:int):void

-mark(s:String):void

+endOrStartOver(s:String):void

+timer():void

1. 难点及细节回顾

1.Lab2制作地雷图

(1)生成足够地雷

在生成地雷时,有随机生成的地雷位置重复导致地雷不足的问题。为了避免,循环次数为bombs,生成随机地雷的for循环中,每次生成地雷后,将相应位置的布尔数组state元素赋值为真,标记地雷位置,加入if语句判断该处state是否为真,若是,则for循环计数器i减一。

1. 周围地雷数的边界统计问题(Lab2中已讨论)
2. Lab3扫雷初级版本

(1)输入地雷图数据后Enter键对游戏中命令产生的影响

游戏中while循环使用户能不断输入坐标以推进游戏。该循环体开头用nextLine读入用户命令,为防止用户输入地雷图数据后按下的Enter键对游戏命令造成的影响,在循环开始前加入String tab = cmd.nextLine();语句。

1. 总结与收获

1.字符串,字符与整型的互相转换

在游戏中用户输入的命令为字符串,地图为字符数组,坐标值为整型数字。程序中常常涉及三者的相互转换,这使我更加熟悉它们的常见转换方法。

字符串→字符:char ch = s.charAt();

字符→字符串:String s = ch + “”;

字符串→数字:int num = Integer.parseInt(s);

数字→字符串:String s = String.valueOf(num);

字符→数字:int num = (int)(ch - ‘0’);

数字→字符:(非负个位数)char ch = String.valueOf(num).charAt();

2.恶意输入防范

该功能的实现迫使我全面考虑所有非法输入的情况,并主动思考判断顺序,防止出现错误。学会使用基本的正则表达式判断格式。

3.循环

该程序大量使用循环结构,让我更加了解三种循环结构的区别,循环条件的设置,更加熟练地应用循环。

1. 递归打开

open方法具有明显的递归特点,使我主动思考递归方法中的逻辑,以免出现无限递归的情况。