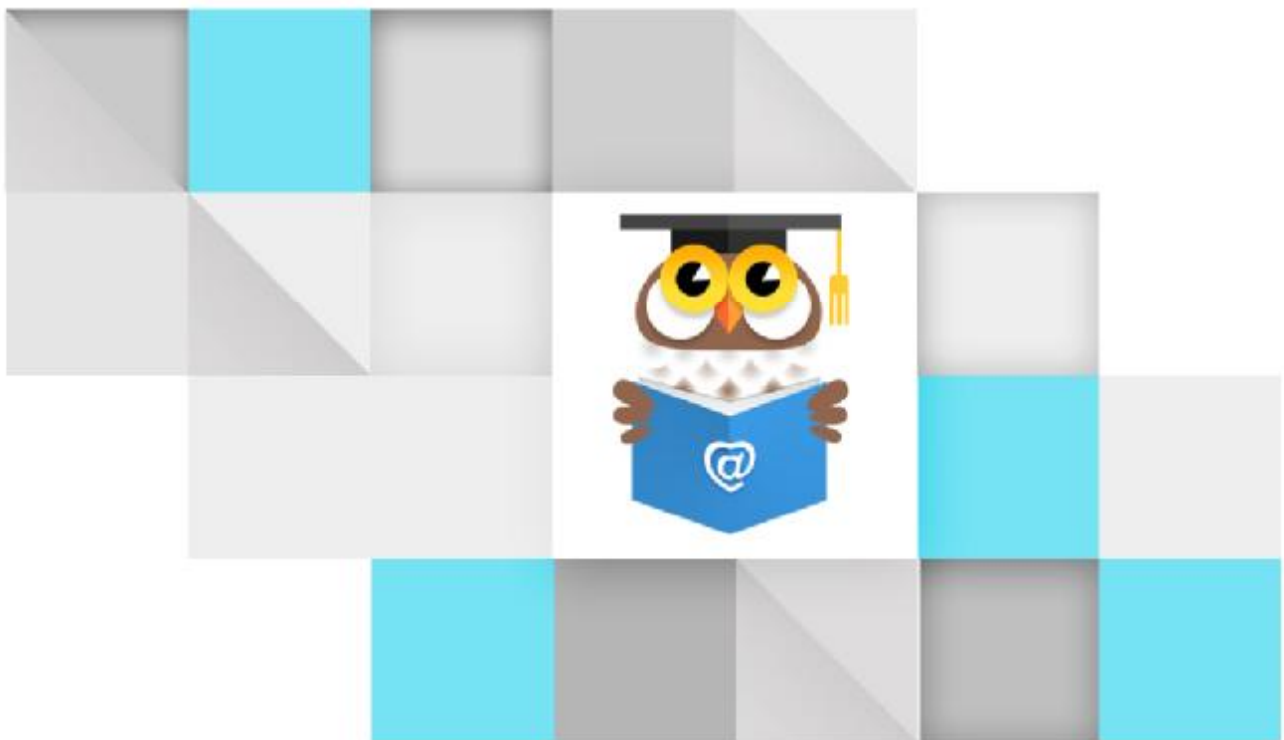


# 消灭泡泡糖 ( Java )

## 实训指导手册

### 实训场景 004 – 获得待消除的泡泡糖



**Campus** Solution Group

## 目 录

一、任务编号：PRJ-BU2-JAVA-004 .....	1
1、实训技能 .....	1
2、涉及知识点 .....	1
3、实现效果 .....	1
4、场景说明 .....	2
5、快速开始 .....	4
6、任务 1 – 泡泡糖克隆函数 .....	5
7、任务 2 – 查询某个泡泡糖左侧同色泡泡糖 .....	7
8、任务 3 – 查询某个泡泡糖右侧同色泡泡糖 .....	13
9、任务 4 – 查询某个泡泡糖顶部同色泡泡糖 .....	16
10、任务 5 – 查询某个泡泡糖底部同色泡泡糖 .....	19
11、场景总结 .....	22

## 一、任务编号：PRJ-BU2-JAVA-004

### 1、实训技能

I Java 面向对象编程技能

### 2、涉及知识点

I 类的成员变量

I 类的成员方法

I static 成员方法

### 3、实现效果



图 3-1

## 4、场景说明

### 1、业务说明：

本场景主要用于查找某个【被点击】的泡泡糖四周是否存在【同色】泡泡糖。将获得的所有【同色】泡泡糖存储于集合中，我们将该集合称为：“待消除泡泡糖” 集合



图 4-1

### 2、实现思路：

- 2-1. 当界面某个泡泡糖【被点击】时，游戏界面会调用服务类中的tobeClearedStars方法。
- 2-2. tobeClearedStars方法负责检查【完整泡泡糖列表】。
- 2-3. 检查时，需将【被点击】泡泡糖四周【同色】泡泡糖保存入“待消除泡泡糖” 集合中。

### 3、核心组件介绍：

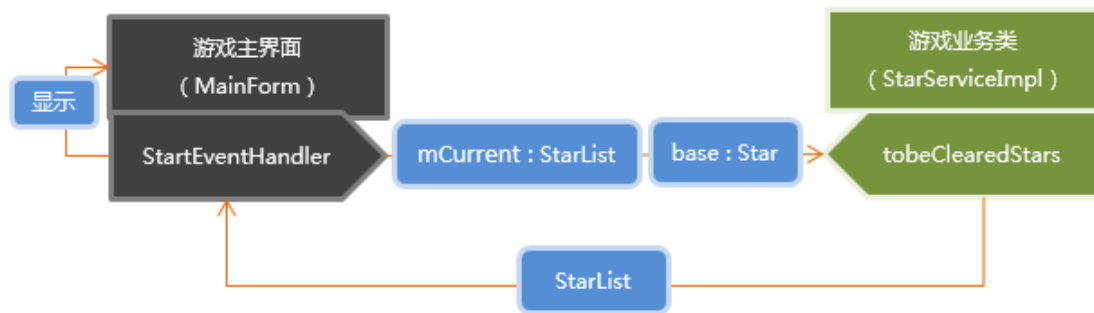


图 4-2

### 3-1. MainForm - 游戏界面类（本场景无需实现）：

负责游戏数据显示、响应用户在界面上的各类操作。

### 3-2. StarEventHandler – 鼠标事件处理类（本场景无需实现）：

负责处理鼠标在游戏界面的各类点击事件。

### 3-3. StarServiceImpl - 游戏业务类：

负责游戏相关逻辑计算，例如：泡泡糖移动、消除、分数计算等操作。

### 3-4. StarServiceImpl类中的tobeClearedStars：

该方法主要通过调用lookupByPath方法查找【被点击】泡泡糖四周【同色】泡泡糖。

### 3-5. tobeClearedStars的【Star base】参数 – 【被点击】的泡泡糖对象：

当前场景描述为：被用户点击的泡泡糖，由页面传递给服务类。

### 3-6. tobeClearedStars的【StarList mCurrent】参数 – 界面【完整的泡泡糖列表】：

当前场景描述为：界面10 \* 10泡泡糖矩阵对应的【完整的泡泡糖列表】，在未执行消除操作前，该集合始终保持着100个随机颜色的泡泡糖。

该集合由场景PRJ-BU2-JAVA-003实现的createStars函数创建。

### 3-7. tobeClearedStars的StarList返回值 - 【待消除泡泡糖】集合：

用于保持所有“待消除泡泡糖”的集合。

### 3-8. StarServiceImpl类中的lookupByPath：

该方法主要负责检查【被点击】泡泡糖的左、右、上、下四格内存放的泡泡糖是否为【同色】：

3-8.1. 若是，则以当前【被搜索到】的泡泡糖为基准，再次调用lookupByPath方法作进一步的查找。

3-8.2. 若不是，则从其他方向执行搜索操作。

#### 4、了解更多：

请参考《消灭泡泡糖 - 需求说明文档》

#### 5、前置条件：

5-1. 前置场景：PRJ-BU2-JAVA-003 – 随机显示泡泡糖

5-2. 必备知识与技能：

5-2.1. Java基本语法（变量、变量类型、运算符、for循环、if条件判断）。

5-2.2. Java面向对象编程技能（工具类、引入包、递归调用）。

## 5、快速开始

### 1、开发环境：

1-1. Oracle JDK8.x 以上版本

1-2. Eclipse Luna ( 4.4.x ) 以上版本

1-3. 工程包：PRJ\_BU2\_JAVA\_004

### 2、进入开发环境：

详见SPOC平台上《PRJ-BU2-JAVA-004 前置任务：进入开发环境》



图 5-1

## 6、任务 1 – 泡泡糖克隆函数

### 1、任务描述：

1-1. 【克隆对象】：意指以某个对象的属性与属性值为基础，创建一个全新的对象，其结构、属性、属性值与源对象完全一致。

1-2. 【克隆对象】和对象引用存在本质差异，【克隆对象】（图6-1）是创建了一个全新对象，它拥有独立的存储空间和引用对象；而对象引用（图6-2）则是多个引用对象访问同一块存储空间。



图 6-1

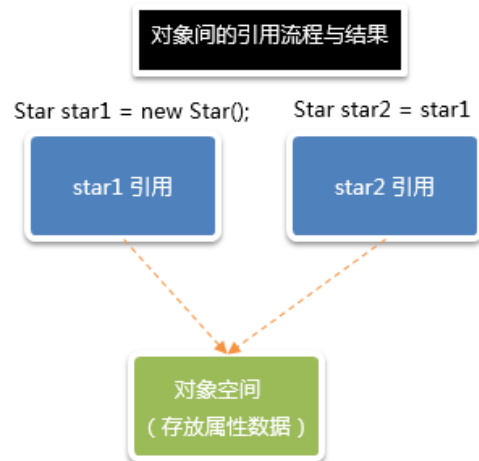


图 6-2

1-3. 当前场景需要检索游戏界面上【完整泡泡糖集合】（A）中与【被点击】泡泡糖同色的对象，并将同色泡泡糖存储于【待消除泡泡糖集合】（B）中。

1-4. 从1-3可知，为了实现泡泡糖消除功能，我们需要将满足消除条件的【同色】泡泡糖从A集合“搬运”到B集合，为保证泡泡糖在A和B两个集合中独立存在，我们需要在“搬运”前，对被“搬运”的泡泡糖进行克隆操作。

## 2、推荐步骤：

2-1. 定位到工具类：cn.campsg.practical.bubble.util.StarsUtil

2-2. 新建一个克隆泡泡糖的方法clone

2-2.1. 方法为公共静态函数

2-2.2. 入参为待克隆的Star对象。

2-3. 新建一个Star对象，该对象为源对象的克隆体。

2-4. 将源对象的行坐标、列坐标、类型依次赋值给新建的Star对象。

### + 提示

- 1) 不能将原Star对象的postion属性直接赋值给新建的Star对象，否则相当于两个Star对象共用了同一个Position，失去了克隆的意义。



2) 为解决以上问题，我们需要为新建的Star对象创建一个Position，再将源Star对象中行值与列值设置给新建的Position对象。

2-5. 返回克隆对象

### 3、验证与测试：

3-1. 在当前StarsUtil类中创建一个main函数，作为测试的程序入口。

3-2. 新建一个Star对象，作为源泡泡糖对象，并设置坐标为(5,5)，颜色为红色。

3-3. 调用clone方法得到一个克隆对象。

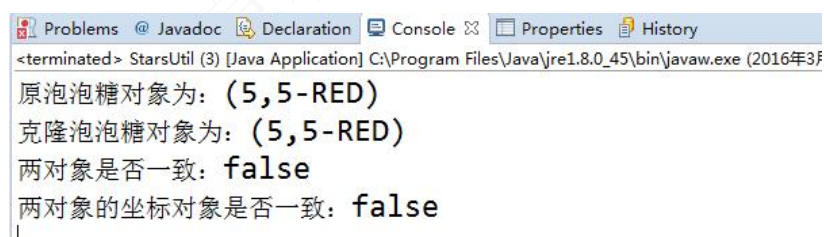
3-4. 向控制台输出源对象内容（直接输出泡泡糖对象即可）。

3-5. 向控制台输出克隆对象内容（直接输出泡泡糖对象即可）。

3-6. 使用equals方法对两泡泡糖进行比较并显示在控制台，用于测试两对象是否相同。

3-7. 使用equals方法对两泡泡糖的position属性对象进行比较并显示在控制台，用于测试Position属性对象是否相同。

3-8. 观察控制台是否显示以下测试内容：



```
<terminated> StarsUtil (3) [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2016年3月25日 14:14:14)
原泡泡糖对象为: (5,5-RED)
克隆泡泡糖对象为: (5,5-RED)
两对象是否一致: false
两对象的坐标对象是否一致: false
```

图 6-1

## 7、任务 2 – 查询某个泡泡糖左侧同色泡泡糖

### 1、任务描述：

检索游戏界面【完整泡泡糖集合】，找出【被点击】泡泡糖【左侧】相邻的【同色】泡泡糖。

1-1. 从【被点击】的泡泡糖s1开始，查找左边一格是否存在与s1【同色】的泡泡糖s2。

1-1.1. 如果有,则从s2开始查找左边一格是否存在和s2【同色】的泡泡糖s3,以此类推。

1-1.2. 找到【同色】泡泡糖后将其视作“待消除泡泡糖”,经【克隆】后保存于“待消除泡泡糖”集合(StarList)中。

1-1.3. 如果无,则停止【同色】泡泡糖查找操作。

1-1.4. 如果超出10 \* 10泡泡糖矩阵的左侧边界(列号小于0),则停止【同色】泡泡糖查找操作。

1-1.5. 实现流程见下图(黄色为【被点击】泡泡糖;蓝色为【左侧同色】泡泡糖)

0,0	0,1	0,2	0,3	0,4	0,5
1,0	1,1	1,2	1,3	1,4	1,5
2,0	2,1	2,2	2,3	2,4	2,5
3,0	3,1	3,2	3,3	3,4	3,5
4,0	4,1	4,2	4,3	4,4	4,5

图 7-1

1-2. lookupByPath函数负责业务流程的实现,该函数共包含三个参数,他们的含义分别为:

1-2.1. Star base: 界面被点击的泡泡糖(左侧同色泡泡糖判断基准对象)。

1-2.2. StarList sList: 界面【完整的泡泡糖列表】(10 \* 10的泡泡糖矩阵)。

1-2.3. StarList clearStars: 【待消除的泡泡糖集合】。

```
private void lookupByPath(  
    Star base, StarList sList, StarList clearStars) {  
  
}  

```

## 2、推荐步骤:

2-1. 场景定位:

2-1.1. 找到业务类: cn.campsg.practical.bubble.service.StarServiceImpl

2-1.2. 定位到方法lookupByPath的任务2注释处。

## 2-2. 获取lookupByPath函数的泡泡糖参数base的信息

2-2.1. 获取行号。

2-2.2. 获取列号。

2-2.3. 获取类型。

2-3. 新建一个用于保存【同色】泡泡糖的Star对象。

2-4. 开始判断【被点击】泡泡糖左侧的【同色】泡泡糖（按图7-2流程实现）。

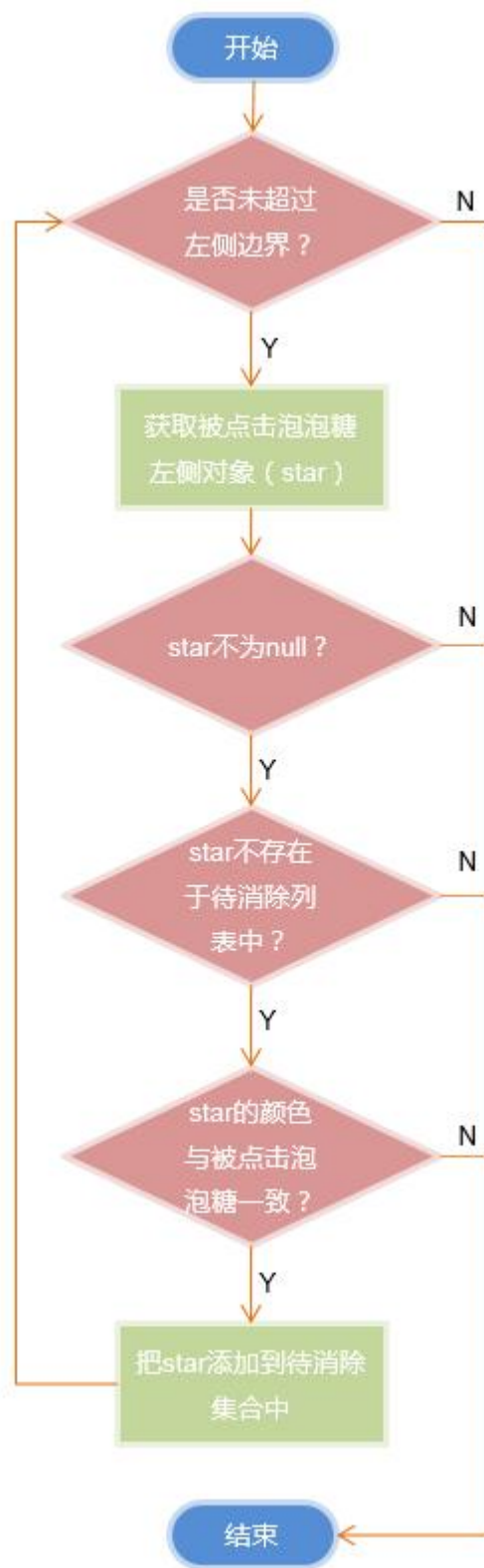


图 7-2

+ 提示

- 1) 【被点击】泡泡糖的左侧泡泡糖坐标：
  - 1-1) 列值 = 【被点击】泡泡糖的【列值 - 1】。
  - 1-2) 行值 = 【被点击】泡泡糖的【行值】。
- 2) 左边界判断：泡泡糖的【列值 - 1】大于等于0表示未超过左边界，允许继续判断。反之，泡泡糖的【列值 - 1】小于0表示超过左边界，无需继续判断。
- 3) 从【完整的泡泡糖集合】中查找泡泡糖：可以直接调用StarList对象中的lookup方法，并将待判断的泡泡糖行值、列值传递给该方法，获取坐标对应的Star对象。
- 4) 判断是否为“待消除泡泡糖”：必须满足三个关键条件：
  - 4-1) 待判断泡泡糖不能为null。
  - 4-2) 待判断泡泡糖不存在于“待消除泡泡糖”集合中（防止重复存储）。判断是否存在于“待消除泡泡糖”集合中，可以直接调用StarList对象的自定义函数：existed，该方法返回true表示存在，false表示不存在。
  - 4-3) 待判断泡泡糖的颜色（类型）与【被点击】泡泡糖的颜色一致。
- 5) 将泡泡糖加入“待消除泡泡糖”集合：可以直接调用StarList对象中的add方法实现。
- 6) **继续往左边查找：如已知【被点击】泡泡糖的左侧为【同色】泡泡糖，那么请再次调用lookupByPath，并将当前判断所得的【同色】泡泡糖作为参数传入（函数递归调用）。**

## 2-5. 在当前类中，实现接口的toBeClearedStars方法

- 2-5.1. 新建一个StarList对象，用于存放所有“待消除泡泡糖”。
- 2-5.2. 由于【被点击】的泡泡糖必属于“待消除泡泡糖”，因此请把base参数添加到待消除列表中。
- 2-5.3. 执行lookupByPath方法，以【被点击】的泡泡糖为基准，查找该泡泡糖左侧是否存在【同色】的泡泡糖。

2-5.4. 如执行完lookupByPath方法后，“待消除泡泡糖”列表中仍然只有一个泡泡糖，则表示查询过程未发现任何【同色】泡泡糖，则清空“待消除泡泡糖”列表。

**+ 提示**

- 1) 使用size方法可以获取StarList对象中包含多少泡泡糖对象。
- 2) 使用clear方法可以清除StarList中的所有泡泡糖对象。

2-5.5. 返回待消除列表

3、验证与测试：

3-1. 定位函数入口所在类：cn.campsg.practical.bubble.MainClass，并运行项目工程

3-2. 当前项目已提供了测试用泡泡糖矩阵。

3-3. 点击界面指定的泡泡糖，观察是否能消除【被点击】泡泡糖左侧的对象，如图所示：



图 7-3

## 8、任务 3 – 查询某个泡泡糖右侧同色泡泡糖

### 1、任务描述：

1-1. 任务2已经实现了：以【被点击】泡泡糖为基准向【左侧】递归判断【同色】泡泡糖的功能。

1-2. 当前任务在任务2的基础上，实现向【右侧】递归判断【同色】泡泡糖的功能。

### 2、推荐步骤：

2-1. 定位到业务类：cn.campsg.practical.bubble.service.StarServiceImpl

2-1.1. 定位到方法lookupByPath的任务3注释处

2-2. 开始判断【被点击】泡泡糖右侧的【同色】泡泡糖（按图8-1流程实现）。

2-2.1. 实现流程基于任务2代码基础之上。

2-2.2. 实现思路与任务2基本类似，只需做局部代码的修正即可。

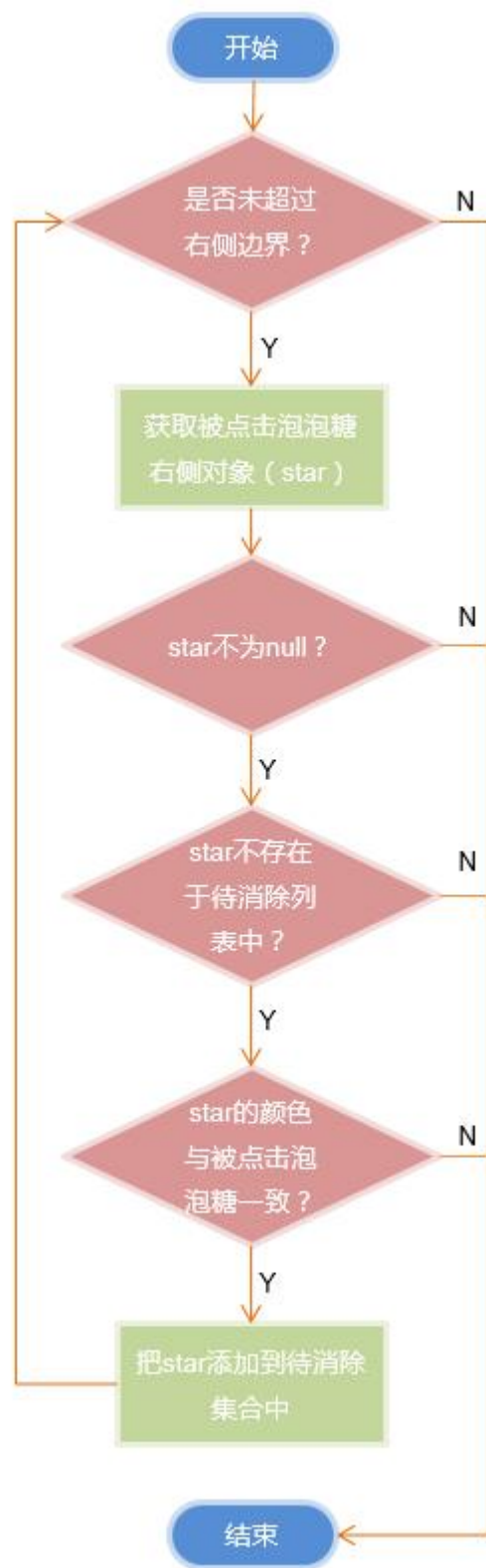


图 8-1



**+ 提示**

- 1) 【被点击】泡泡糖的右侧泡泡糖坐标：
  - 1-1) 列值 = 【被点击】泡泡糖的【列值 + 1】。
  - 1-2) 行值 = 【被点击】泡泡糖的【行值】。
- 2) 右边界判断：泡泡糖的【列值 + 1】小于最大列值 ( MAX\_COLUMN\_SIZE ) 表示未超过右边界，允许继续判断。反之， 泡泡糖的【列值 + 1】大于等于最大列值 ( MAX\_COLUMN\_SIZE ) 表示超过右边界，无需继续判断。
- 3) **继续往右侧边查找：如已知【被点击】泡泡糖的右侧为【同色】泡泡糖，那么请再次调用lookupByPath ,并将当前判断所得的【同色】泡泡糖作为参数传入( 函数递归调用 )。**

**3、验证与测试：**

- 3-1. 定位函数入口所在类：cn.campsg.practical.bubble.MainClass，并运行该项目工程。
- 3-2. 目前已经完成了查找左右两侧【同色】泡泡糖的功能，点击界面上指定的泡泡糖，观察能否消除一整行【同色】泡泡糖：



## 9、任务 4 – 查询某个泡泡糖顶部同色泡泡糖

### 1、任务描述：

1-1. 任务2与任务3分别实现了：以【被点击】泡泡糖为基准向【左侧】与【右侧】递归判断【同色】泡泡糖的功能。

1-2. 当前任务在任务2和任务3的基础上，实现向【顶部】递归判断【同色】泡泡糖的功能。

### 2、推荐步骤：

2-1. 定位到业务类：cn.campsg.practical.bubble.service.StarServiceImpl。

2-1.1. 定位到方法lookupByPath的任务4注释处

2-2. 开始判断【被点击】泡泡糖顶部的【同色】泡泡糖（按图9-1流程实现）。

2-2.1. 实现流程基于任务2代码基础之上。

2-2.2. 实现思路与任务2基本类似，只需做局部代码的修正即可。

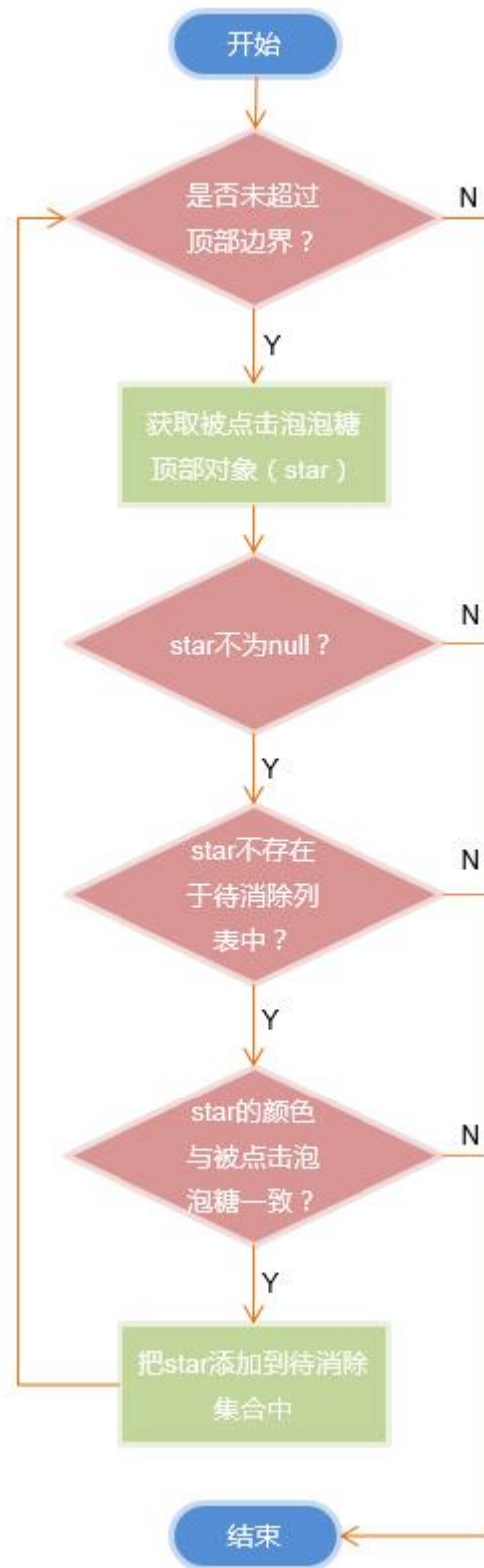


图 9-1

+ 提示

- 1) 【被点击】泡泡糖的顶部泡泡糖坐标：
  - 1-1) 列值 = 【被点击】泡泡糖的【列值】。
  - 1-2) 行值 = 【被点击】泡泡糖的【行值 - 1】。
- 2) 顶部边界判断：泡泡糖的【行值 - 1】大于等于0表示未超过顶部边界，允许继续判断。  
反之，泡泡糖的【行值 - 1】小于0表示超过顶部边界，无需继续判断。
- 3) **继续往顶部查找：如已知【被点击】泡泡糖的顶部为【同色】泡泡糖，那么请再次调用 `lookupByPath`，并将当前判断所得的【同色】泡泡糖作为参数传入（函数递归调用）。**

3、验证与测试：

3-1. 定位函数入口所在类：`cn.campsg.practical.bubble.MainClass`，并运行该项目工程。

3-2. 目前已经完成了查找左、右和上三侧【同色】泡泡糖的功能，点击界面上指定的泡泡糖，观察能否消除【被点击】泡泡糖所在行以及之上的所有【同色】泡泡糖：



## 10、任务 5 – 查询某个泡泡糖底部同色泡泡糖

### 1、任务描述：

1-1. 任务2、任务3、任务4分别实现了：以【被点击】泡泡糖为基准向【左侧】、【右侧】、

【顶部】递归判断【同色】泡泡糖的功能。

1-2. 当前任务在任务2、任务3、任务4的基础上，实现向【底部】递归判断【同色】泡泡糖的功能。

### 2、推荐步骤：

2-1. 定位到业务类：cn.campsg.practical.bubble.service.StarServiceImpl。

2-1.1. 定位到方法lookupByPath的任务5注释处

2-2. 开始判断【被点击】泡泡糖底部的【同色】泡泡糖（按图10-1流程实现）。

2-2.1. 实现流程基于任务2代码基础之上。

2-2.2. 实现思路与任务2基本类似，只需做局部代码的修正即可。

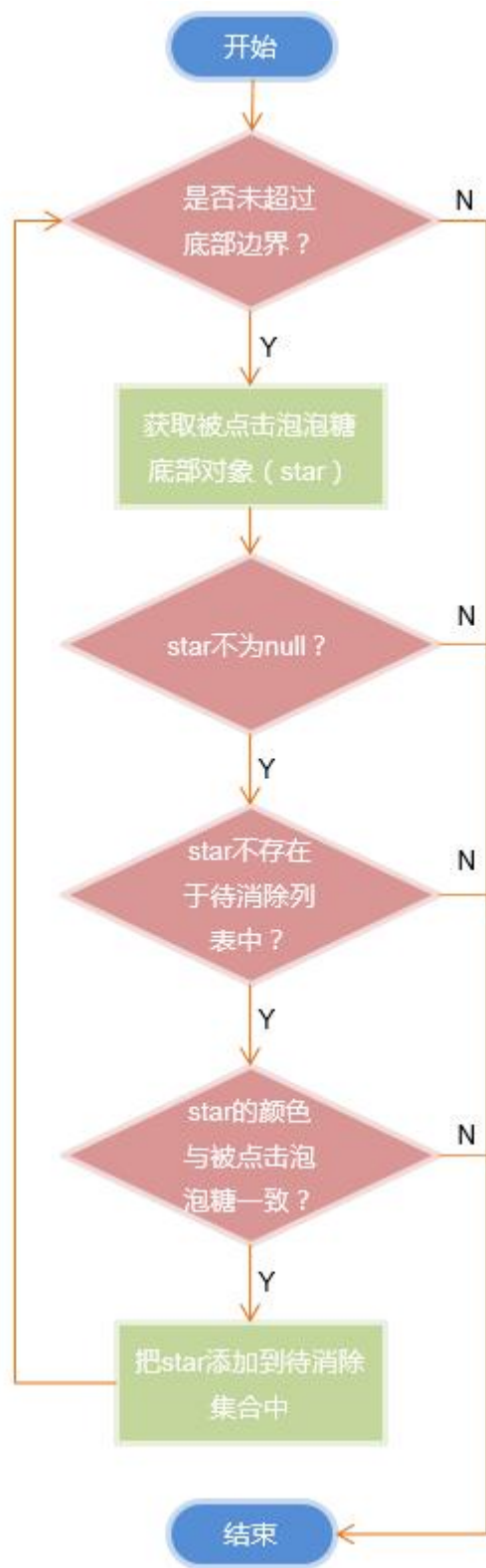


图 10-1

+ 提示

1) 【被点击】泡泡糖的底部泡泡糖坐标：

1-1) 列值 = 【被点击】泡泡糖的【列值】。

1-2) 行值 = 【被点击】泡泡糖的【行值 + 1】。

2) 底部边界判断：泡泡糖的【行值 + 1】小于最大行值 ( MAX\_ROW\_SIZE ) 表示未超过底部边界,允许继续判断。反之, 泡泡糖的【行值 + 1】大于等于最大行值( MAX\_ROW\_SIZE ) 表示超过底部边界, 无需继续判断。

3) **继续往底部查找：如已知【被点击】泡泡糖的底部为【同色】泡泡糖，那么请再次调用 lookupByPath，并将当前判断所得的【同色】泡泡糖作为参数传入（函数递归调用）。**

3、验证与测试：

3-1. 定位函数入口所在类：cn.campsg.practical.bubble.MainClass，并运行该项目工程。

3-2. 目前已经完成了四个方向查找的功能，点击界面上指定的泡泡糖，观察能否消除当前泡泡糖四个方向的所有同色泡泡糖：





## 11、场景总结

**Q1. 在您的项目中是否使用过递归函数？请谈谈递归函数的运用场景。**

1. 递归函数是指函数对其本身的重复调用，递归函数在使用时需设计明确的“调用点”和“退出点”。

1-1. 调用点：何时对函数本身执行调用操作，一般调用都是在判断下执行，防止死循环。

1-2. 退出点：递归过程退出点，一般都是调用点条件不满足的情况下，执行退出点。

2. 当前场景中，通过搜索同类"泡泡糖"执行递归调用，函数名：`lookupByPath`

2-1. 调用点：用户点击的"泡泡糖"左侧、右侧、上方、下方存在同类"泡泡糖"。

2-2. 退出点：当前"泡泡糖"左侧、右侧、上方、下方无任何同类"泡泡糖"。

**Q2. clone 函数如何使用？使用时有哪些注意事项？**

1. clone 函数是对某个对象完全复制（注意：不是引用），克隆后的对象具有以下特性：

1-1. 与原始对象具有相同的属性值。

1-2. 与原始对象逻辑结构完全一致。

1-3. 与原始对象具有不同的内存地址（完全就是两个对象）。

2. clone 与引用的区别：

2-1. 引用特性：



```
Star s1 = new Star();
```

```
Star s2 = s1;
```

以上代码：s1 和 s2 指代的是同一块内存空间，无论操作的 s1 还是 s2 的属性都是对同一个空间做修改。

2-2. 克隆特性：

```
Star s1 = new Star();
```

```
Star s2 = StarUtils.clone(s1);
```

以上代码：s1 和 s2 指代的是两块完全不同的内存空间。