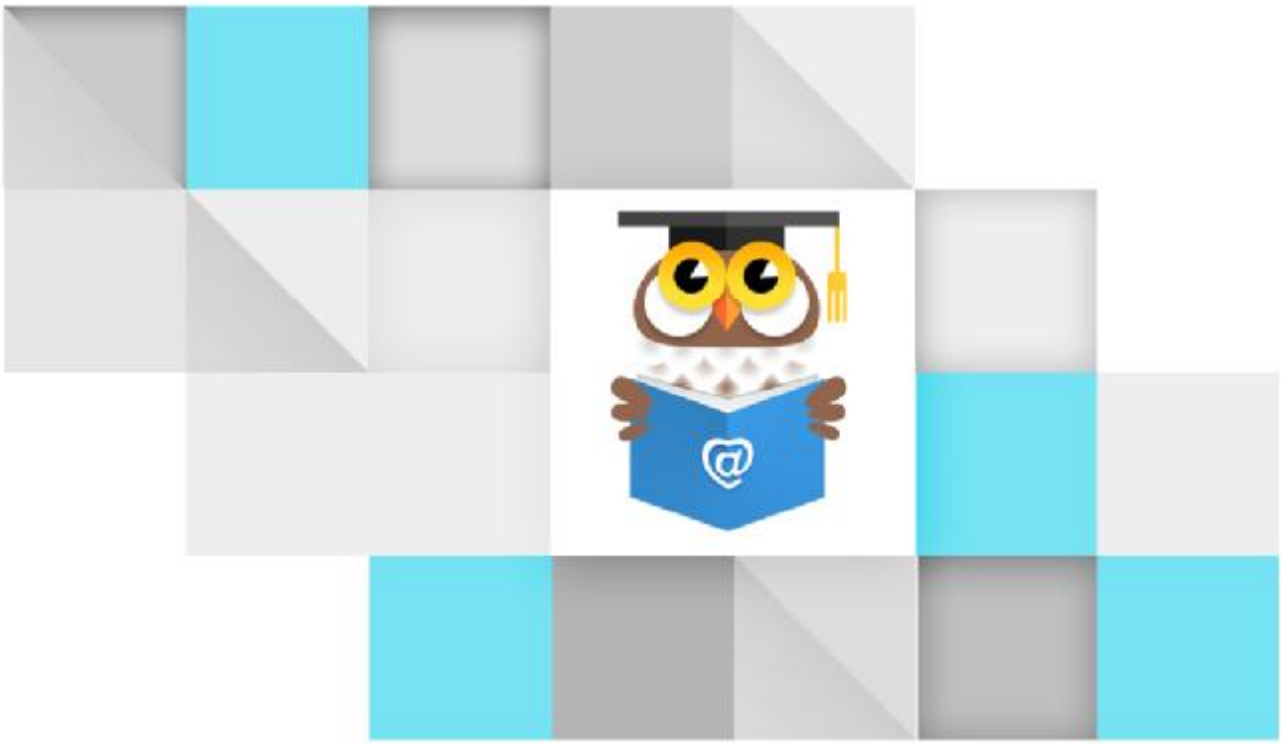
消灭泡泡糖（Java）

实训指导手册

实训场景 010 – 移动垂直方向的泡泡糖（一）



Campus Solution Group

目 录

一、任务编号：PRJ-BU2-JAVA-010

1、实训技能

- Java API 运用技能

2、涉及知识点

- List 实现类的特点
- List 中的方法
- List 的应用

3、实现效果

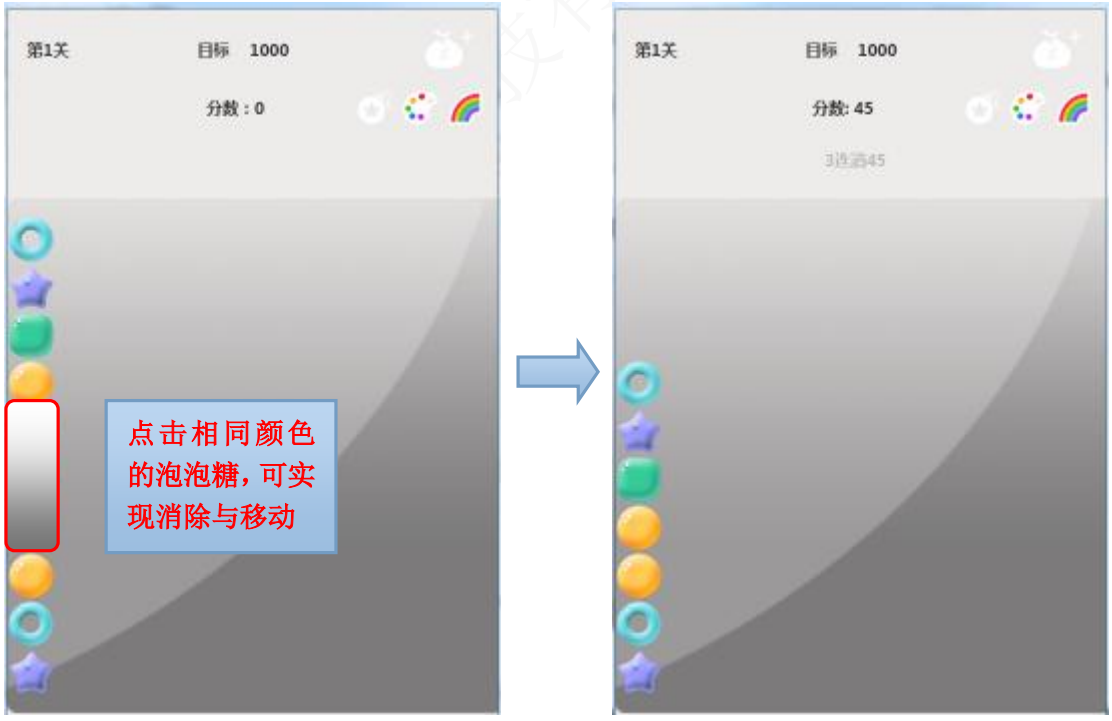


图 3-1

4、场景说明

1、业务说明：

本场景主要用于实现游戏界面上单列泡泡糖消除与垂直移动“填补空隙”的功能。

本场景在**PRJ-BU2-JAVA-005**场景的基础上实现了单个泡泡糖移动步长和泡泡糖移动个数自动计算的效果。

2、实现思路：

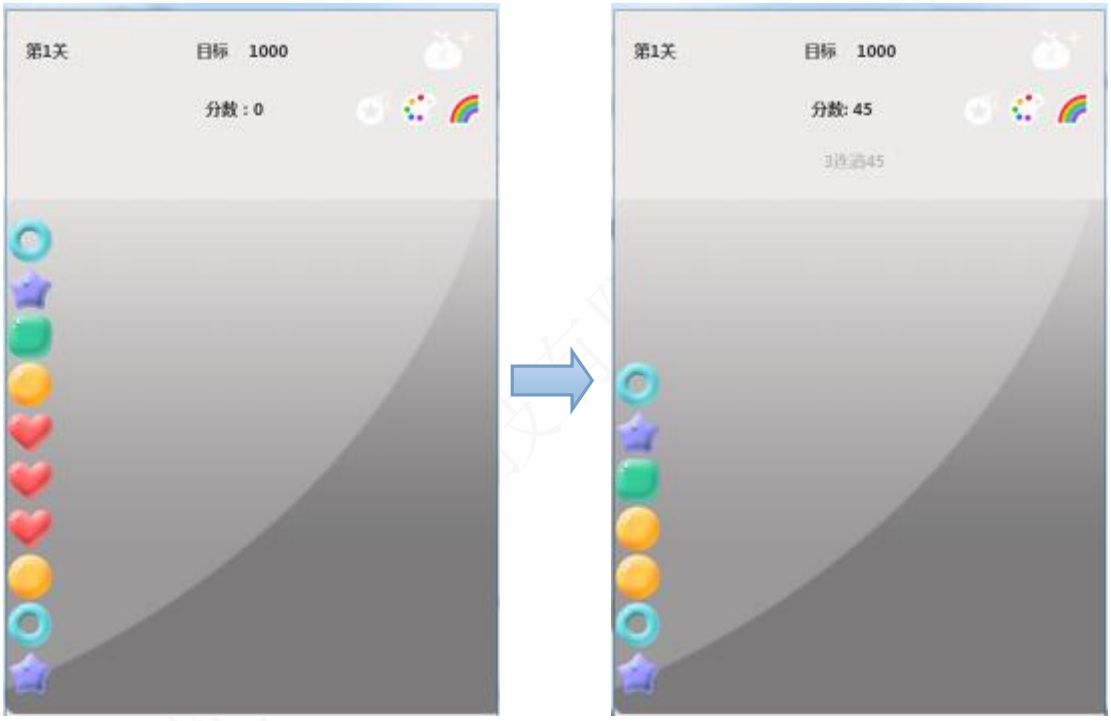


图 **4-1**

从图**4-1**中可见，当用户点击红色泡泡糖，游戏界面将完成以下操作：

2-1. 通过**StarServiceImpl**对象获取“**待消除的泡泡糖”**（场景**PRJ-BU2-JAVA-004**已完成）。

2-2. 通过**StarServiceImpl**对象获取**“垂直方向待移动泡泡糖”**（**本场景实现**）。

2-3. 根据获取的“待消除的泡泡糖”对象，消除相关联的**3**个红色泡泡糖。

2-4. 根据获取的“垂直方向待移动泡泡糖”对象，移动顶部的**4**个泡泡糖“填补空隙”。

获取**“垂直方向待移动泡泡糖”**由**StarServiceImpl**的**getYMovedStar**函数负责，并将结果

保存于**StarList**中传递给游戏界面，游戏主界面**MainForm**负责读取**StarList**中的泡泡糖对象，并依次完成**2-3**和**2-4**两个核心步骤。

3、核心组件介绍：

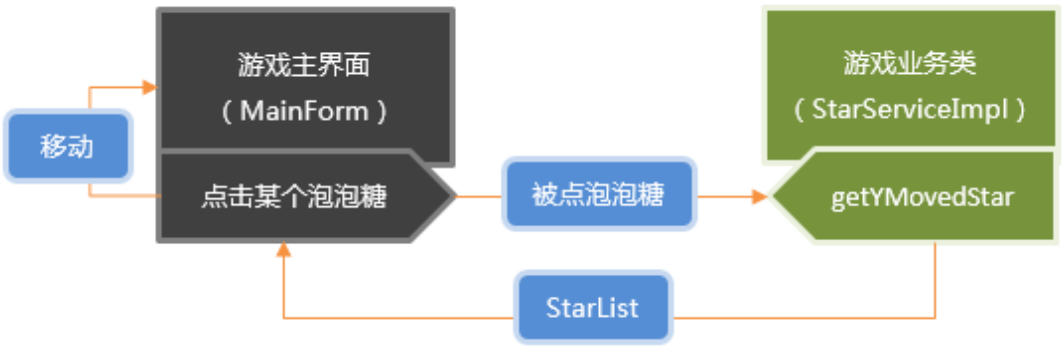


图 **4-2**

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. 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。

3-3. StarList - 用于保存待移动泡泡糖（**MovedStar**）集合。

3-4. StarServiceImpl中的getYMovedStars：

该方法主要负责计算**“垂直方向待移动泡泡糖”**的数量和移动步长，并保存在**StarList**中。回顾场景**PRJ-BU2-JAVA-005**发现，我们已经实现了简单的泡泡糖垂直方向移动，但是移动步长和移动数量都是固定的，在本场景中我们将会解决**PRJ-BU2-JAVA-005**场景中泡泡糖只能移动固定步长和数量的问题。

步骤一、按行号排序**“待消除的泡泡糖”**，以此获取**“待消除的泡泡糖”**中行号最大的泡泡糖，它所在的位置将是所有“垂直方向待移动泡泡糖”的移动“终点”。

步骤二、以行号最大的泡泡糖为起始点向顶部依次判断所有泡泡糖，以此计算移动步

长和待移动泡泡糖数量。

3-5. StarsUtil –工具类

提供一些简单的操作函数，如排序，克隆，类型转换等。

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-005** – 封装待移动的泡泡糖

5-2. 必备知识与技能：

5-2.1 Java开发工具 （**Eclipse**）。

5-2.2 Java面向对象编程（变量、变量类型、运算符、条件分支语句、循环、类的成员方法）。

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_010**

2、 进入开发环境：

详见**SPOC**平台上《**PRJ-BU2-JAVA-010** 前置任务：进入开发环境》

图 **5-1**

6、任务 1 – 交换两个泡泡糖

1、任务描述：

1-1. 实现交换两个泡泡糖数据的函数 – **swap**。

1-2. 为确保能正确计算“垂直方向待移动泡泡糖”的个数与移动步长，我们必须对“待消除泡泡糖“进行排序（以此获取行号最大的待消除泡泡糖）。

1-3. 当前场景我们选择冒泡排序算法，在实现排序算法前，我们需要先完成一个交换函数，它是冒泡排序的基础。

2、推荐步骤：

2-1. 场景定位

2-1.1. 定位到：**cn.campsg.practical.bubble.util.StarsUtil**类

2-2. 创建无返回值的静态方法**swap**

2-2.1. 方法参数：两个需要交换的**Star**对象

+ 算法说明：

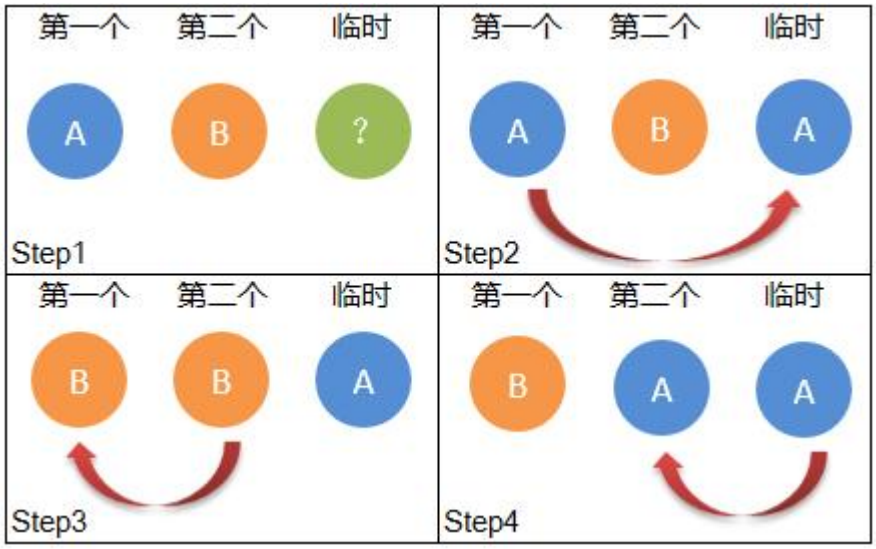
交换两个对象数据，一共分为四步：

Step1：创建临时对象，用于存储数据；

Step2：将第一个泡泡糖对象的属性数据，拷贝到临时对象；

Step3：将第二个泡泡糖对象的属性数据，拷贝到第一个对象；

Step4：将临时泡泡糖对象的属性数据，拷贝到第二个对象；



2-3. 在**swap**方法中，创建临时**Star**对象

2-4. 将**swap**第一个参数的对象属性数据（行、列、类型）赋值给临时对象对应的属性。

2-5. 将**swap**第二个参数的对象属性数据（行、列、类型）赋值给第一个对象对应的属性。

2-6. 将临时对象中的属性数据（行、列、类型）赋值给第二个对象对应的属性。

小心：对象交换时，务必交换对象中的属性，而不是简单的将A对象通过等于号赋给B对象。

+ 提示：

1） 由于交换函数定义在工具类中，因此函数可以设置为静态（**static**）。

2） 由于交换函数仅供后续的排序函数调用，因此可以设置为私有函数（**private**）。

3、验证与测试：

3-1. 在**StarsUtil**类中创建程序入口**main**函数

3-2. 编写以下测试语句：

3-2.1. 创建第一个**Star**对象：坐标为（**0,0**），类型为**BLUE**

3-2.2. 创建第二个**Star**对象：坐标为（**1,1**），类型为**GREEN**

3-2.2. 调用**swap**函数，交换两个**Star**对象

3-2.4. 分别打印两个对象

3-3. 打印测试结果：

3-3.1. 在**swap**函数调用前后，分别打印出两个**Star**对象

3-3.2. 输出结果与下图一致：

```
交换前:preStar:(0,0-BLUE) nextStar:(1,1-GREEN)
交换后:preStar:(1,1-GREEN) nextStar:(0,0-BLUE)
```

图 **6-1**

7、任务 2 – 泡泡糖集合的排序

1、任务描述：

1-1. 对“待消除泡泡糖”进行排序操作，排序过程中利用（任务**1**）实现的交换函数。

1-2. 排序算法：冒泡排序（按行号从小到大排序）。

2、推荐步骤：

2-1. 场景定位

2-1.1. 定位到：**cn.campsg.practical.bubble.util.StarsUtil**类

2-2. 创建静态方法**sort**

2-2.1. 方法参数：“待消除的泡泡糖”集合

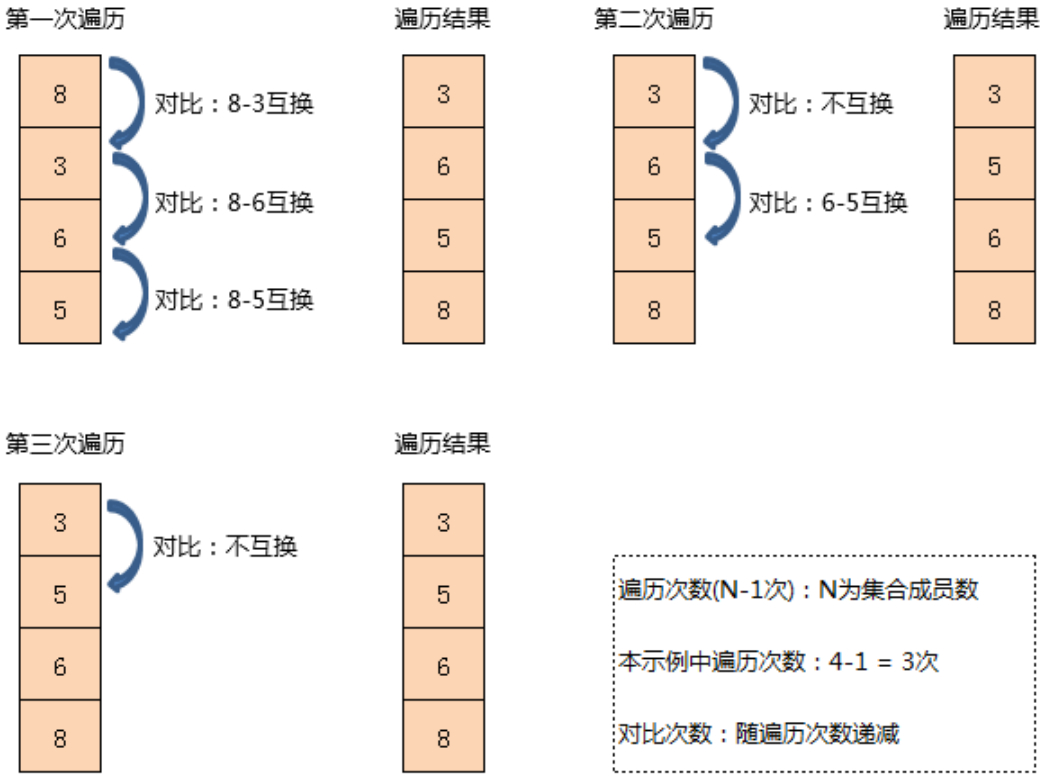
2-2.2. 返回值：无

+ 算法说明：

1） 冒泡排序的原理是对集合做**N-1**次遍历，**N**为集合的成员数量。

2） 每次遍历都需要依次比较集合中**2**个相邻对象的大小，将较大的对象放到后面，遍历完成后，最后一个对象即为当前最大元素。

3） 由于最后一个成员始终为最大元素，因此每遍历一次集合，相邻对象间的对比可以减少一次（如下图所示）。



4） 当前业务中以泡泡糖的行数值作为对比数据。

2-3. 在**sort**方法中，创建集合遍历**for**循环，循环范围：从**0 ~**【泡泡糖集合长度 **- 1**】。

2-4. 创建成员对比**for**循环，嵌套在集合遍历**for**循环内，循环范围：从**0 ~** 【泡泡糖集合长度 **-** 集合遍历**for**循环索引 **- 1**】

2-5. 在成员对比**for**循环中实现两个相邻泡泡糖的对比与交换：

2-5.1. 获取集合中两个相邻泡泡糖对象。

2-5.2. 如果当前对象的行值大于相邻对象的行值，则：

1） 将两个泡泡糖对象的属性数据交换。

2） 继续下一次循环。

+ 提示：

1） 泡泡糖交换的方法**-swap**在任务**1**已经完成。

2） 获取当前泡泡糖使用：集合**.get(**循环变量**)**。

3） 获取相邻泡泡糖使用：集合**.get(**循环变量**+1)**。

3、验证与测试：

3-1. 找到任务**1**中测试用的**main**函数，删除函数中原有测试代码

3-2. 编写以下测试语句：

3-2.1 创建**StarList**集合实例

3-2.2 依次添加五个不同位置的泡泡糖对象：

1） **【2,0】- BLUE**

2） **【5,0】- GREEN**

3） **【9,0】- PURPLE**

4） **【3,0】- RED**

5） **【8,0】- YELLOW**

3-2.3. 为泡泡糖集合进行排序

3-3. 打印测试结果：

3-3.1 在**sort**函数调用前后，分别打印出泡泡糖集合

3-3.2 输出结果与下图一致：

```
排序前, starList:
·(2,0-BLUE)      ,(5,0-GREEN)     ,(9,0-PURPLE)    ,(3,0-RED)       ,(8,0-YELLOW)
排序后, starList:
·(2,0-BLUE)      ,(3,0-RED)       ,(5,0-GREEN)     ,(8,0-YELLOW)    ,(9,0-PURPLE)
```

图 **7-1**

8、任务 3 – 移动垂直方向的泡泡糖

1、 任务描述：

1-1. 通过**getYMovedStars**函数实现获取“垂直方向待移动泡泡糖“的集合。

界面有能力获取**getYMovedStars**函数的返回，并在界面上实现泡泡糖的消除与移动。

1-2. 场景**PRJ-BU2-JAVA-005**实现的函数**-getYMovedStars**，只能实现固定坐标、固定数量的消除业务，这里我们将为函数添加随机位置、随机数量泡泡糖的消除功能：

1-2.1. 首先，通过排序函数获取“待消除泡泡糖“中行号最大的泡泡糖。

1-2.2. 然后，以行号最大的泡泡糖为起点逐行向顶部（行号**=0**）做判断：

1） 遇到“待消除的泡泡糖”那么移动步长**+1**。

2） 遇到普通泡泡糖，则根据移动步长创建待移动的泡泡糖，并加入**StarList**做好移动准备。

3） 遇到**null**，则代表上方没有泡泡糖，提前终止遍历。

4） 流程详见下图，图中绿色方块为待消除泡泡糖。

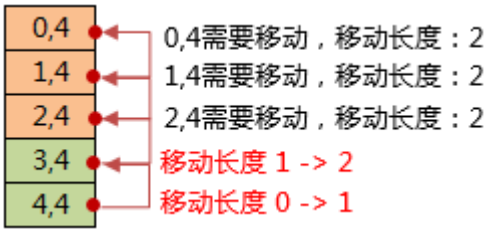


图 **8-1**

2、推荐步骤：

2-1. 场景定位：

2-1.1. 定位到类：**cn.campsg.practical.bubble.service.StarServiceImpl**

2-1.2. 找到类中**getYMovedStars**函数。

+ 业务说明：

getYMovedStars方法共包含**2**个参数：

1） **StarList clearStars**：待消除泡泡糖集合（场景**PRJ-BU2-JAVA-004**实现）。

2） **StarList currentStarList**：完整的泡泡糖集合（内含泡泡糖数与界面一致）。

2-2. 变量初始化

2-2.1. 创建待移动泡泡糖集合实例，用于存放待移动泡泡糖。

2-2.2. 定义变量：

1） 定义描述列号的变量，变量初始值为：**0**

+ 业务说明：

由于当前场景只需考虑单列泡泡糖，因此列号始终为**0**。

2） 定义变量用于描述垂直方向移动距离：初始值为**0**

2-3. 获取最下方待消除泡泡糖行值，作为遍历的起始值

2-3.1. 首先，我们需要对“待消除泡泡糖”集合进行排序。

2-3.2. 排序后，我们需要获取集合的最后一位成员，也就是行号最大的泡泡糖。

+ 提示：

1） 集合排序可以利用任务**2**实现的排序函数 **- sort**。

2） 获取集合中最后一个元素：集合**.get(**集合长度 **– 1)**。

2-4. 创建循环，循环范围从【行号最大的泡泡糖行值】到【**0**】，循环中实现以下代码：

2-5. 通过循环变量【行号】和【列值变量】，从完整的泡泡糖集合（**currentStarList**）中获取对应位置的泡泡糖对象。

+ 提示：

1） 从集合中获取对应行列值的元素，可以利用集合的**lookup**函数，该函数会在下一个场

景中实现。

2-6. 如果泡泡糖对象为**null**，则结束循环。

+ 业务说明：

1） 泡泡糖对象为**null**，代表上方已没有泡泡糖，可提前结束循环。

注意：行号 **= 0**表示到达泡泡糖矩阵的顶部，泡泡糖对象 **= null**，表示上方已没有泡泡糖，见下图：

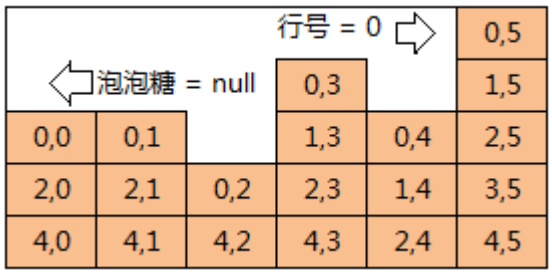


图 **8-2**

2） 循环结束可以使用**break**关键字。

2-7. 如果泡泡糖属于待消除泡泡糖，则让【移动距离的变量】**+1**，并直接进入下一次循环

+ 业务说明：

判断**Star**对象是否存在于集合中，可以直接使用集合的**existed**函数，该函数会在下一个场景中实现。

2-8. 如果以上**2**个判断都不成立，则封装待移动泡泡糖

2-8.1. 创建待移动泡泡糖对象

2-8.2. 将步骤**2-5**获取的泡泡糖类型和位置，赋值给待移动泡泡糖

2-8.3. 将步骤**2-5**获取的泡泡糖【行值**+**移动步长】、【列值】作为待移动泡泡糖的目标位置的行列值

2-9. 将封装的待移动泡泡糖添加到待移动泡泡糖集合中

2-10. 在方法末尾将方法返回值从**null**，修改为待移动泡泡糖集合

3、验证与测试：

3-1. 运行项目工程，选择**cn.campsg.practical.bubble.MainClass**类作为程序入口

3-2. 程序会显示一列泡泡糖，其中可消除泡泡糖的个数和位置均为随机生成

3-3. 点击同色泡泡糖可实现消除与移动功能。

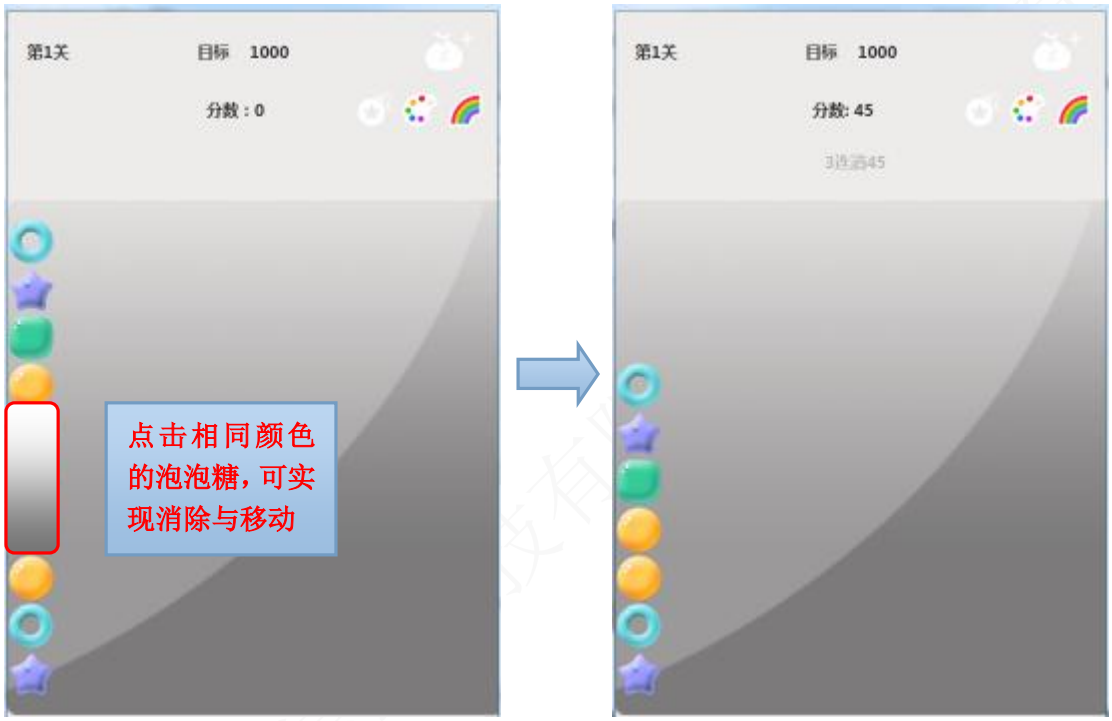


图 **8-3**

9、场景总结

Q1. 在您的项目中如何使用 ArrayList 对象？结合项目说说 ArrayList 对象的优势。

1. Java 集合框架隶属于 **java.util** 包，其中 **ArrayList** 非常常用。

2. ArrayList 又称为可变数组，它的优势在于可以随意存放任意数据。

3. ArrayList 通过 **add** 方法顺序存放数据，**get** 方法按索引号获取数据。

4. 项目中如果遇到类型相同的对象（当前游戏中 **Star** 对象）需要存放与交互，就可以考虑使用集合。

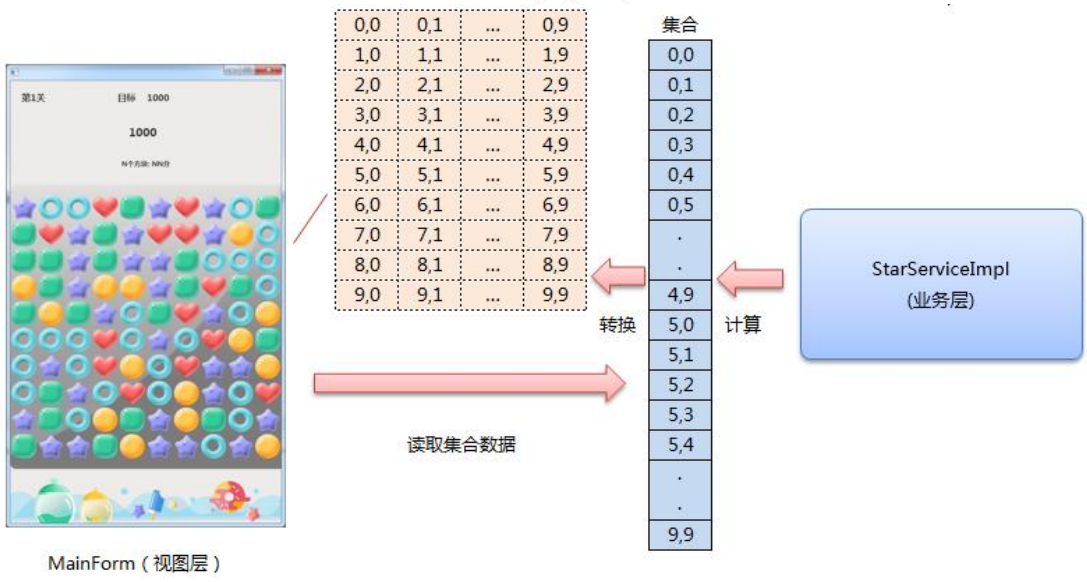
当前游戏中集合设计的目的如下（重要）：

1. 当前游戏系统分为界面层（**MainForm**）和业务层（**StarService**）：

1-1. MainForm 负责游戏界面的人机交互。

1-2. starService 负责"泡泡糖"的消除、计算、移动（之前场景已涉及）。

2. 由于界面只负责“泡泡糖”的显示工作，所有消除、移动、计算工作都由业务层负责，因此我们就必须在两层间设计一个组件，界面层从该组件中获取计算后的“泡泡糖”整列，业务层则将计算结果保存在这个组件中，该组件就是一个集合 **ArrayList**（我们将其扩展成 **StarList**），结构如下图：



Q2. 项目中有使用过 Utils 类，Utils 类中函数有哪些特点？

1. Utils类又被称为工具类，一般该类会提供各种业务所需的功能函数（例如：排序、分组等）。

2. Utils类往往存放具有相同业务特性的功能函数：

2-1. DateUtils中可以提供，例如：获取系统时间与日期、日期相减等功能函数。

2-2. StringUtils中可以提供，例如：字符串截取与拼接、字符串替换等功能函数。

3. Utils类一般都是不能继承（**final**）和实例化的（构造函数私有）。

4. 为了方便调用，**Utils**类中的函数一般都是静态的。

Q3. 静态关键字是否可以随意使用？为什么？

1. static关键字具有程序级生命周期，一旦创建直到程序停止才会销毁。

2. static关键字定义的元素（函数、变量）可在程序任意模块中访问，相对方便。

3. 由于**static**关键字占用的内存空间分配有限，绝不可以随意定义**static**元素。