**=**

**课 程 设 计 报 告**

**课程名称** 程序设计实践

**题 目** 消灭泡泡糖游戏设计与开发

**指导教师**

**起止日期**

**学 院**  计算机学院

**专 业**

**姓名（班级/学号）**

**成 绩**

|  |  |  |
| --- | --- | --- |
| **需求分析和概要设计（20分）** | **详细设计**  **（20分）** | **文献查阅和关键问题解决（10分）** |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 评分标准 |  | 分值 | 说明 |
| 课程设计报告（50分） | 需求分析和概要设计（20分） | 0-9 | 需求分析不详细，功能不够完善，业务流程设计和应用架构设计不够合理。 |
| 12-13 | 需求分析基本大致合理，功能基本完善，业务流程设计和应用架构设计基本合理，但存在一些小问题。 |
| 14-15 | 需求分析基本详细合理，功能较为完善，业务流程设计和应用架构设计基本合理。 |
| 16-17 | 需求分析较为详细合理，功能模块划分合理完善，业务流程和应用架构设计较为合理。 |
| 18-20 | 需求分析详细合理，功能模块划分合理完善，业务流程和应用架构设计合理。 |
| 详细设计（20分） | 0-9 | 类的设计以及类与类之间的关系设计不够合理，UML类图绘制不够规范，算法描述和关键代码不够详细、缺失或者存在错误。 |
| 12-13 | 类的设计以及类与类之间的关系设计大致合理，UML类图绘制基本规范，针对算法和关键代码基本的描述说明，在不太重要的地方存在小问题。 |
| 14-15 | 类的设计以及类与类之间的关系设计基本合理，UML类图绘制基本规范，针对算法和关键代码基本的描述说明。 |
| 16-17 | 类的设计以及类与类之间的关系设计较为合理，UML类图绘制较为规范正确，算法描述和关键代码实现部分有详细的描述说明。 |
| 18-20 | 类的设计以及类与类之间的关系设计合理，UML类图绘制规范正确，算法描述和关键代码实现部分有详细的描述说明。 |
| 文献查阅和关键问题解决（10分） | 0-6 | 查阅文献自学能力不够，针对关键性问题的解决给出的方案有瑕疵，也没有详细说明。 |
| 7 | 查阅文献自学能力一般，针对关键性问题的解决能给出较为正确的解决方案，但没有详细说明。 |
| 8 | 查阅文献自学能力较强，能够针对关键性问题给出较为正确解决方案，并详细说明。 |
| 9-10 | 查阅文献自学能力强，能够针对关键性问题给出正确的解决方案，并详细说明。 |

**目录**

**注：文章中的代码不是截图，为插入的OpenDocument Text文档。双击可打开**

**第一章**  **整体开发流程分析**

1.1 开发分工明细

1.2 整体业务流程

1.3 游戏结构设计

**第二章**  **显示泡泡糖**

2.1 创建《消灭泡泡糖》实体类

2.2 优化《消灭泡泡糖》实体类

2.3 游戏界面呈现泡泡糖

**第三章 随机显示泡泡糖**

3.1 显示一行泡泡糖

3.2 显示10\*10泡泡糖矩阵

3.3 随机显示10\*10泡泡糖矩阵

**第四章 获得待消除的泡泡糖**

4.1 泡泡糖克隆函数

4.2 查询某个泡泡糖左侧同色泡泡糖

4.3 查询某个泡泡糖右侧同色泡泡糖

4.4 查询某个泡泡糖顶部同色泡泡糖

4.5 查询某个泡泡糖底部同色泡泡糖

**第五章 封装待移动的泡泡糖**

5.1 创建带移动泡泡糖实体类

5.2 实体类的文本化输出函数

5.3 实现移动泡泡糖封装

**第六章 体验接口解耦性**

6.1 创建服务测试类

6.2 实现界面泡泡糖显示

6.3 通过接口动态切换实现类

**第七章 体验接口隔离性**

7.1 处理泡泡糖点击事件

7.2 实现点击-消除效果（不考虑移动）

**第十章 移动垂直方向的泡泡糖（一）**

10.1 交换两个泡泡糖

10.2 泡泡糖集合的排序

10.3 移动垂直方向的泡泡糖

**第十一章 移动垂直方向的泡泡糖（二）**

11.1 根据坐标查找泡泡糖

11.2 根据位置查找泡泡糖

11.3 判断泡泡糖是否存在

**第十二章 移动垂直方向的泡泡糖（三）**

12.1 更新集合的排序算法

12.2 带消除泡泡糖的排序与分组

12.3 获取垂直方向待移动泡泡糖

**第十三章 移动水平方向的泡泡糖**

13.1 获取被整列清空的泡泡糖集合

13.2 获取水平待移动泡泡糖

**第十四章 更新关卡通关分数**

14.1 获取初始通关分数

14.2 更新关卡通关分数

**第十五章 实现泡泡糖的积分规则**

15.1 获取消除奖励分数

15.2 获取结算时奖励分数

15.3 显示通关提示

**第十六章 遇到问题解决**

16.1 枚举类型

16.2 HashMap

16.3 BufferedReader

**第十七章 参考文献**

1. **整体开发流程分析**

**1.1开发分工明细：**

美工：负责涉及泡泡糖和泡泡糖界面样式，负责CSS相关

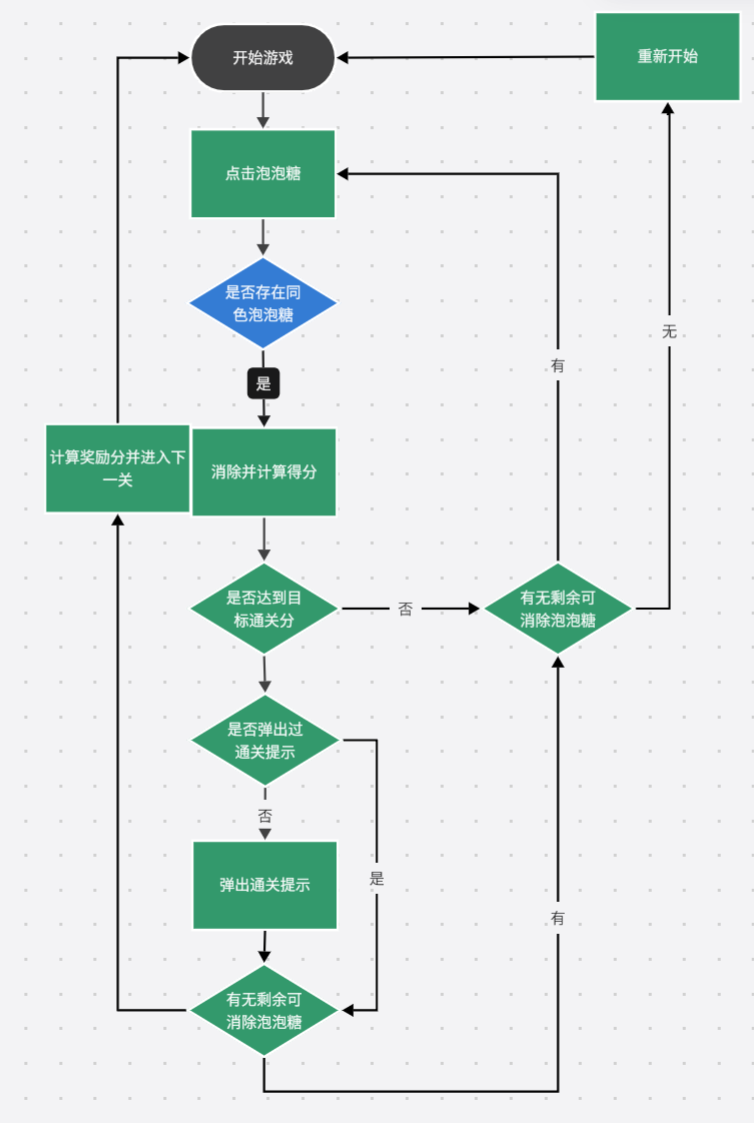
分析员：进行用户需求的分析和游戏功能，数值的研究

界面设计人员：负责进行游戏界面的前端开发，负责JavaFX相关

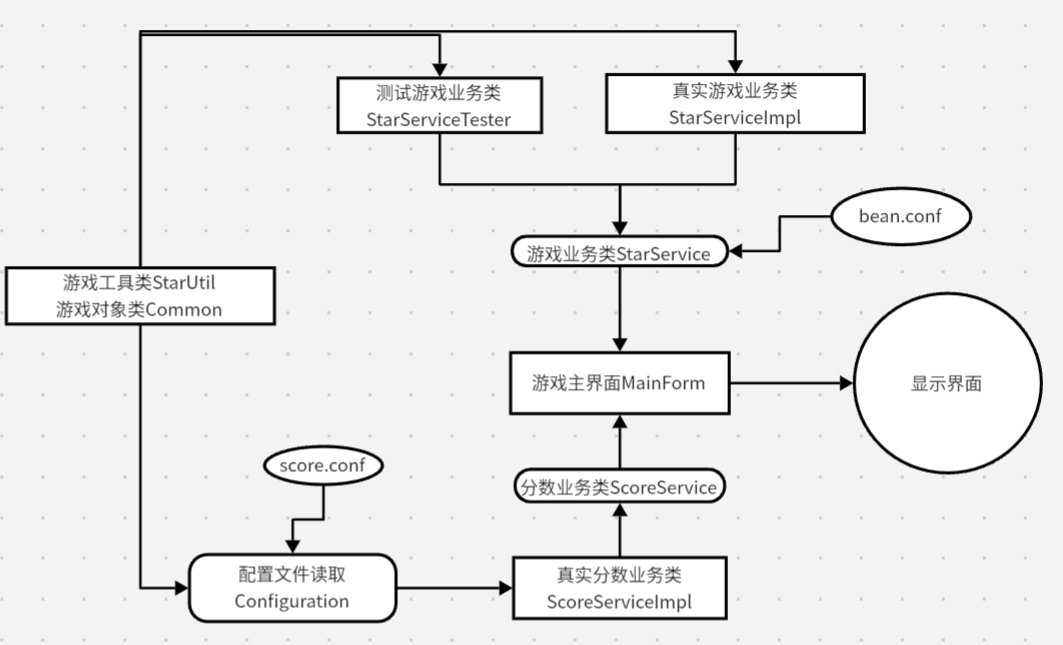
业务设计人员：负责游戏功能的后端开发,负责Java相关

测试人员：负责在界面调用测试类业务进行功能验证

**1.2整体业务流程：**

****

**1.3游戏结构设计：**

****

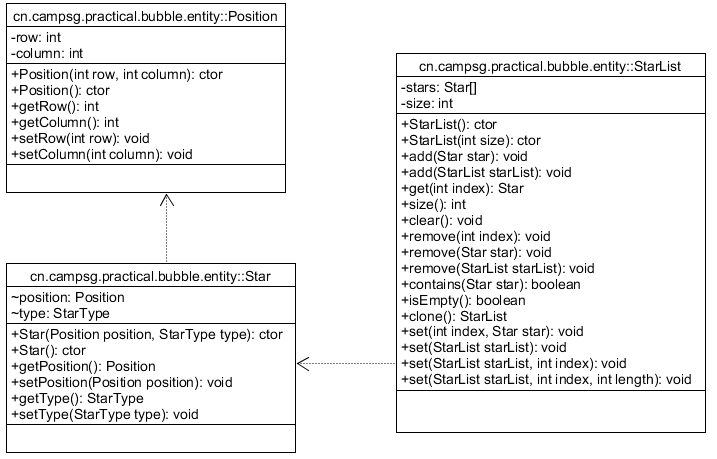
**第二章 显示泡泡糖**

需求分析：该部分要求在游戏界面上呈现五个不同颜色的泡泡糖，且泡泡糖的位置由其坐标值来决定。

功能分析：由需求可知，需要呈现五个不同的泡泡糖，那么可以通过创建一个创建泡泡糖对象的函数，通过赋予不同坐标值和颜色来创建不同的泡泡糖类型。完成以后，将所以被创建的泡泡糖传递给游戏界面，游戏界面负责呈现。

业务流程和应用架构设计：在这部分中，我们利用StarServiceImpl的createStars函数创建5个颜色不同的泡泡糖，每个泡泡糖都是Star类的一个实例，所以被创建的泡泡糖都保存在StarList之中，游戏主界面MainForm负责读取StarList中改动泡泡糖对象，并呈现给用户。

涉及UML图：



**2.1创建《消灭泡泡糖》实体类**

**2.1.1任务描述：**

创建实体类Star以描述游戏中的一个泡泡糖。它有2个重要属性，分别为Position和Type分别表示位置和颜色。其中Position属性含有一个坐标分别以row和column来表示，坐标规则为从左到右，从上到下，从0到9。Type属性含有5中，分别为黄色圆圈，红色心形，绿色圆角，紫色五角星，蓝色空心圆。

**2.1.2代码实现：**

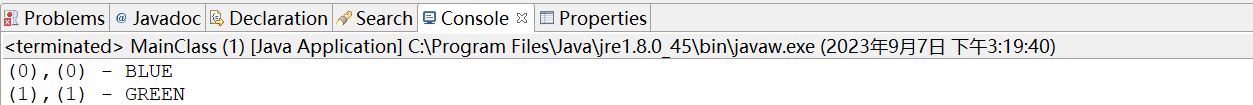
Position类：****

Star类：



**2.1.3验证与测试**

****

****

**2.2优化《消灭泡泡糖》实体类**

**2.2.1任务描述：**

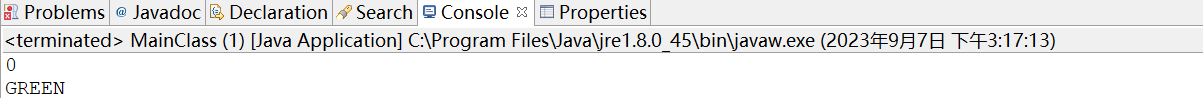
任务2.1完成以后，Star类中的StarType枚举还不够完善，枚举类型只能使用，不能实现类型转换，该任务为给枚举添加转换函数，实现数值向枚举的转化（0->BLUE）

**2.2.2代码实现：**

****

**2.2.3验证与测试**

****

****

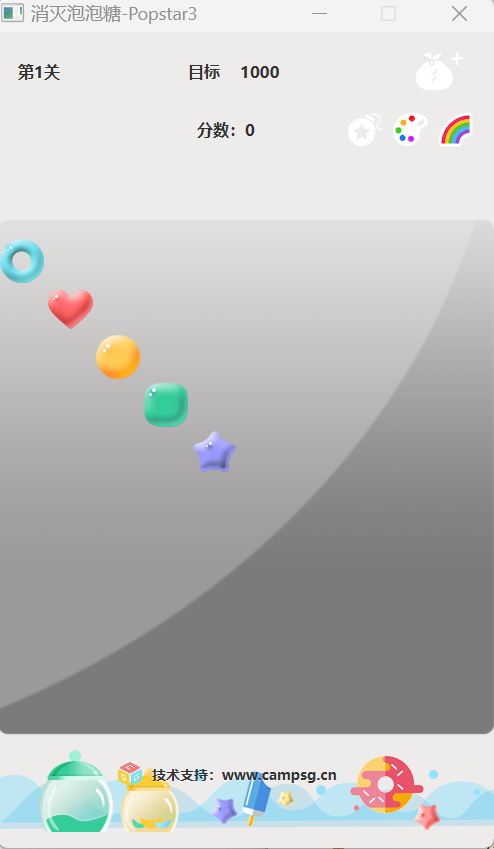
**2.3游戏界面呈现泡泡糖**

**2.3.1任务描述：**

**任务1,2创建并利用控制台显示了泡泡糖数据，当前任务会把Star类数据显示在游戏界面上。**

**2.3.2代码实现：******

**2.3.3验证与测试：**

****

第三章 随机显示泡泡糖

**需求分析：**在002场景中已经能实现在指定位置显示指定颜色的泡泡糖，现在次场景作之前场景的补充，需要在界面显示一个10\*10的随机颜色的泡泡糖矩阵。

**功能分析：为设计10\*10的矩阵，**则需要两层for循环创建Star实例对象，外层确定行值，内层确定列值。并且每一次循环都在指定的枚举类型中随机选择一种。然后将矩阵呈现在游戏界面即可。

**业务流程和架构分析设计：**两层for循环，每次循环均通过StarServiceImpl的creatStars方法创建一个指定Position属性随机Type属性的Star实例对象，并且添加到StarList这个实例数组中。循环结束，将StarList显示到游戏主界面MainForm。

**3.1显示一行泡泡糖**

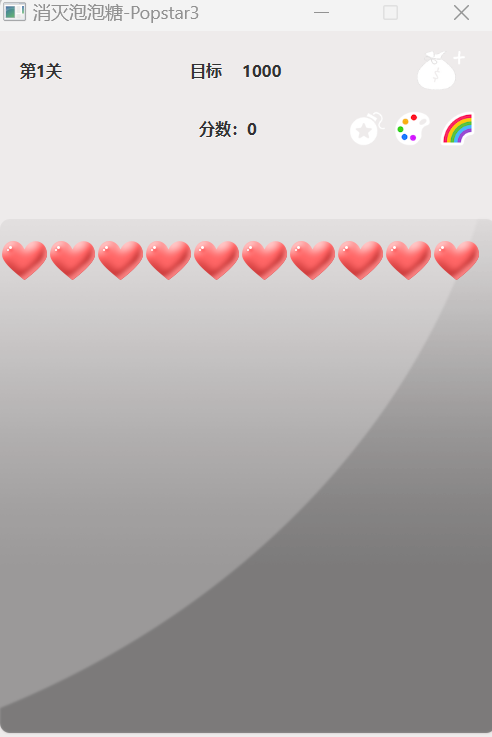
**3.1.1任务描述:**

使用for循环创建一行（10个）泡泡糖，要求十个泡泡糖显示在界面的第一行（行号为0）。则，只需要对泡泡糖的列属性赋值即可。

**3.1.2代码实现:**

****

**3.1.3验证与测试**

****

**3.2显示10\*10泡泡糖矩阵**

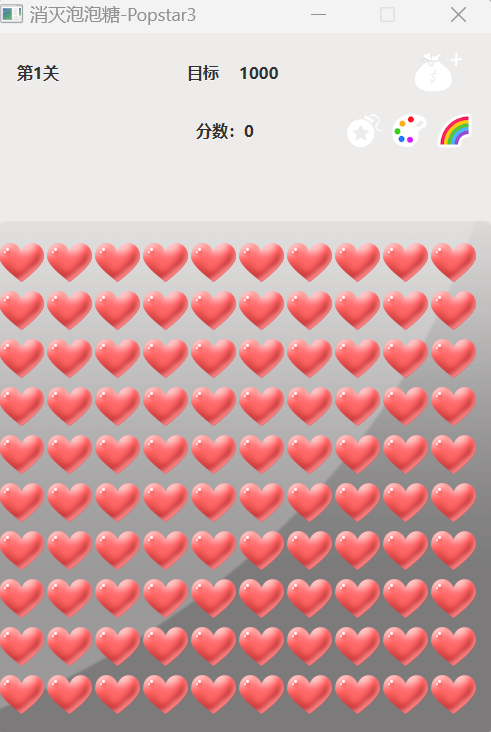
**3.2.1任务描述：**

任务1只能显示一行10个红色泡泡糖，为了保障在界面显示10\*10的泡泡糖矩阵，需要再套一层for循环，使其在0~9的行内变化。

**3.2.2代码实现：**

****

**3.2.3验证与测试**

****

**3.3随机显示10\*10泡泡糖矩阵**

**3.3.1任务描述：**

任务1,2实现了呈现10\*10的指定颜色泡泡糖矩阵效果，现在要求泡泡糖的颜色要再5个指定颜色内随机设置。

**3.3.2代码实现：**

****

**3.3.3验证与测试：**

****

第四章 获得待消除的泡泡糖

**需求分析：**本场景用于查找某个【被点击】的泡泡糖四周是否存在【同色】的泡泡糖，将获得的所以【同色】泡泡糖存于集合中，将次集合称为“待消除泡泡糖”集合。

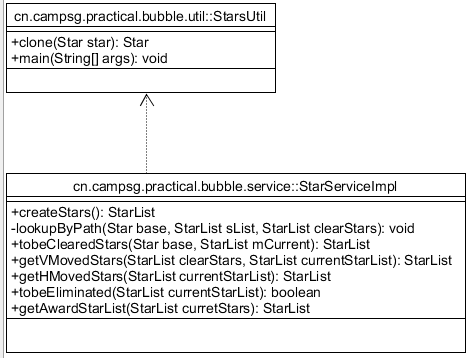
**功能分析：**该场景需要查找被点击泡泡糖周围（上，下，左，右四个方向）的同色泡泡糖。并且需要判断检索到的泡泡糖是否已在”待消除泡泡糖”的集合中。在遍历的时候，若发现同色的泡泡糖，需要以该泡泡糖为中心，在周围再检索一次。

**业务流程和架构分析设计：**通过StartEventHandler监视鼠标点击事件，触发事件后，现在的泡泡糖矩阵mCurrent：StarList以及被点击位置的Star:base传入到StarServiceImpl的tobeCleardStars方法中来获得待消除泡泡糖集合StarList并将他传入到游戏界面MainForm。

**涉及算法分析：**

在tobeCleardStars方法中，需要调用lookupByPath方法查找【被点击】泡泡糖四周【同色】泡泡糖，该方法主要负责调查【被点击】泡泡糖的左，右，上，下内存放的泡泡糖是否为【同色】若是同色，则以当前【被检索到】的泡泡糖为基准，再次调用lookupByPath方法，若不是，则从其他方向执行搜索操作。这是一个函数递归调用的一个算法，该算法可以使得调用的函数本身作为函数的参数再次执行函数，直到达到停止条件。其中当递归最大或者最小时，需要退出递归。

**涉及UML图：**



**4.1泡泡糖克隆函数：**

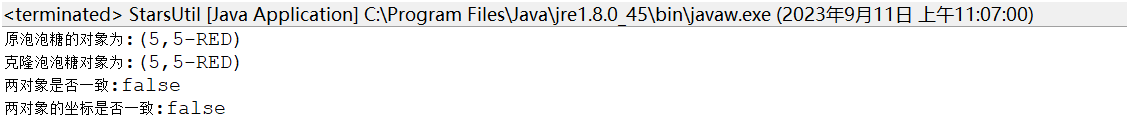
**4.1.1任务描述;**

为了使“待消除泡泡糖”集合中的每一个泡泡糖均独立存在，不会影响到原本正在检索的泡泡糖矩阵。使得泡泡糖进入该集合时是它的克隆对象进入，而不是本体。该任务要求写出一个泡泡糖克隆函数，要求创建一个全新的对象与原版的对象属性和属性值完全相同，但占用的存储空间和引用对象不同。

**4.1.2代码实现：**

****

**4.1.3验证与测试：**

****

**4.2查询某个泡泡糖左侧同色泡泡糖**

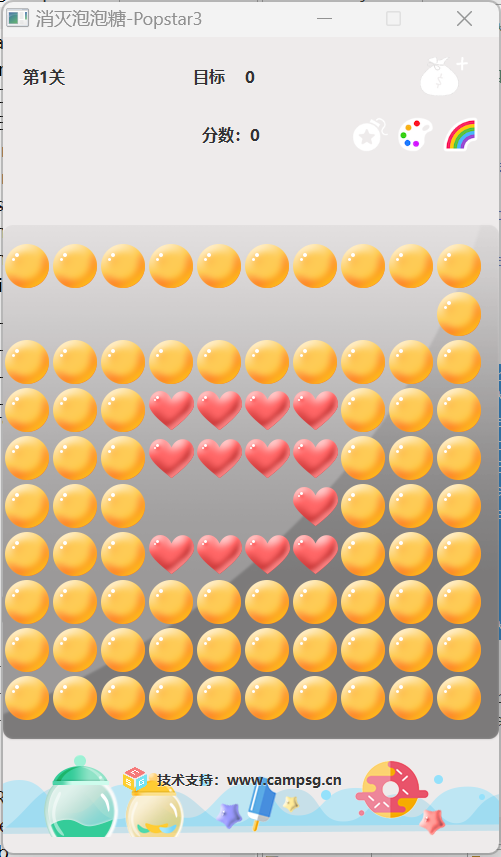
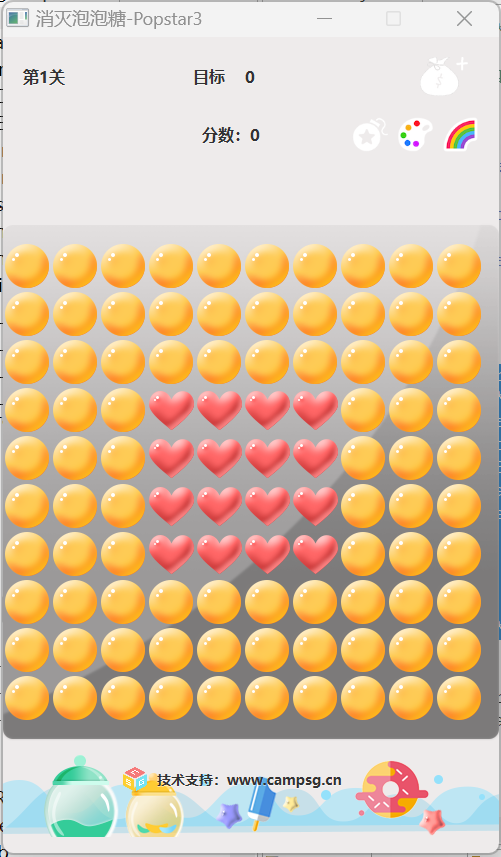
**4.2.1任务描述：**

检索整个游戏界面【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。

**4.2.2代码实现：**

****

**4.2.3验证与测试**

****

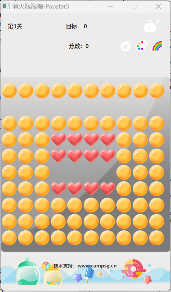
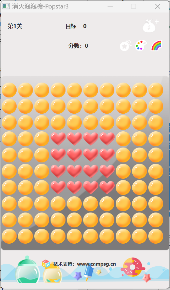
**4.3查询某个泡泡糖右侧同色泡泡糖**

**4.3.1任务描述：**

任务二实现了递归判断左侧【同色】泡泡糖的功能，此任务在任务二的基础上递归判断右侧【同色】泡泡糖。

**4.3.2代码实现：**

****

**4.3.3验证与测试**

**4.4查询某个泡泡糖顶部同色泡泡糖**

**4.4.1任务描述：**

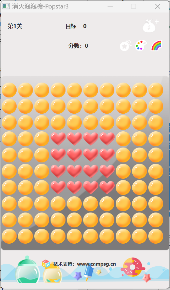
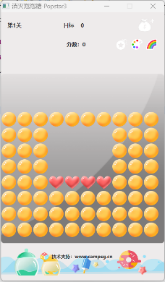
任务2,3实现了判断【被点击】泡泡糖为基准的一行左右侧同行【同色】泡泡糖的功能。

此任务在2,3，基础上，实现向【顶部】递归判断【同色】泡泡糖的功能。

**4.4.2代码实现：**

****

**4.4.3验证与测试：**

****

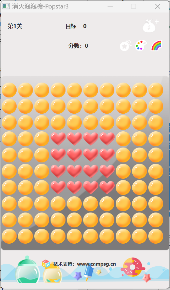
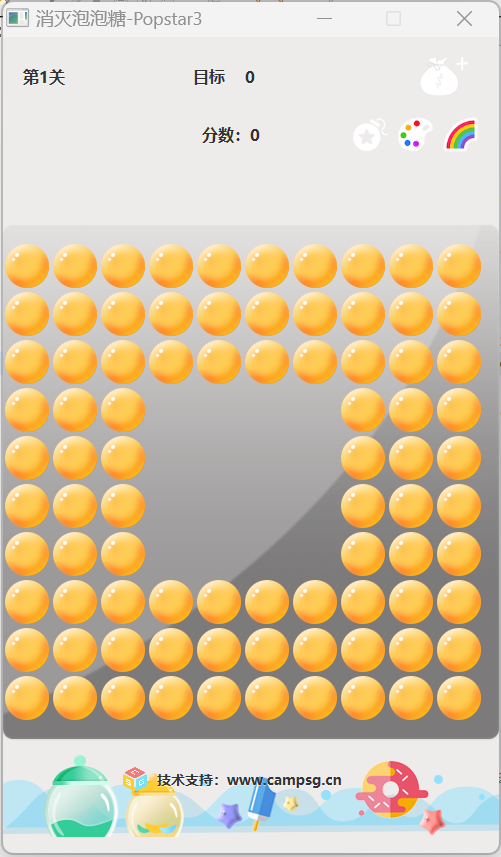
**4.5查询某个泡泡糖底部同色泡泡糖**

**4.5.1任务描述：**

任务2,3,4，分别实现了【被点击】泡泡糖向【左侧】【右侧】【顶部】递归判断【同色】泡泡糖的功能，该任务在之前的基础上，实现向【底部】递归判断【同色】泡泡糖的功能

**4.5.2代码实现：**

****

**4.5.3验证与测试：**

第五章 封装待移动的泡泡糖

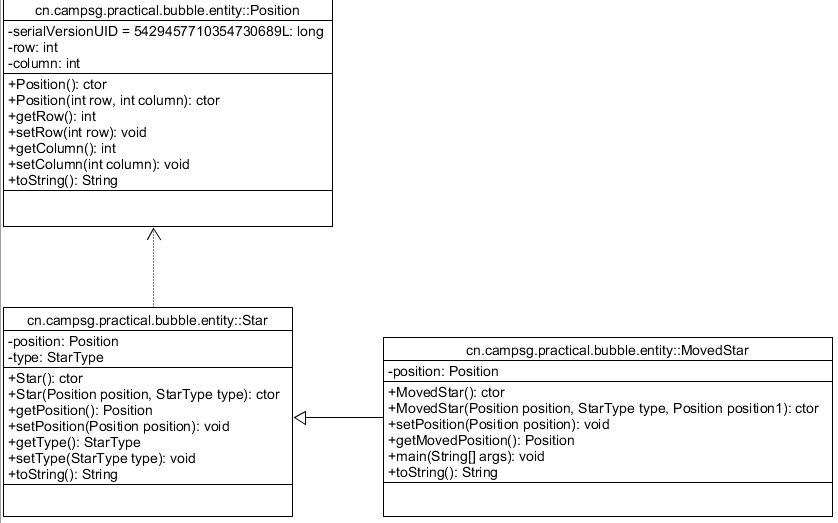
**需求分析：** 由目标功能可知，将所以相邻的【同色】泡泡糖消除后，需要执行【移动】操作来填补【同色】泡泡糖消除后留下的“空隙。”需要确定【移动】的泡泡糖的集合以及对应集合移动的位置与距离。（该场景目前仅考虑单列固定颜色的泡泡糖）

**功能分析：**由场景4已经能找到【被点击】泡泡糖四周所有的【同色】泡泡糖，接下来，将所有【同色】的泡泡糖定义为【待消除泡泡糖】并保存在集合中，该场景仅考虑固定单列泡泡糖。

**业务流程和架构分析设计：** 当用户点击指定的泡泡糖时，通过调用tobeClearedStars函数来获取【同色】【待消除泡泡糖】集合clearStars（已知为2个红色泡泡糖）。然后调用getYMovedStars方法计算并获取所有的【待移动泡泡糖】，该对象要求拥有【待移动坐标】(movedPosition)属性，所以要创建一个MovedPosition来封装【待移动泡泡糖】的移动属性。将所有【待移动泡泡糖】合并到集合currentStarList。然后把currentStarList返回给游戏主界面MainForm。界面依次【消除】泡泡糖，并【移动】未消除泡泡糖填补空隙。

**涉及数据结构分析:**该场景应用到了链表的数据结构，在该场景的单列泡泡糖的移动，实际上就是一个单向链表，当【待消除泡泡糖】集合的数据剔除以后，剩下的【待移动泡泡糖】集合他们之间的前后关系并没有改变，所以只要把最后一个的链接点与消除后的第一个链接点相连，就形成了整列向下平移的效果。

**涉及UML图：**



**5.1创建待移动泡泡糖实体类**

**5.1.1任务描述：**

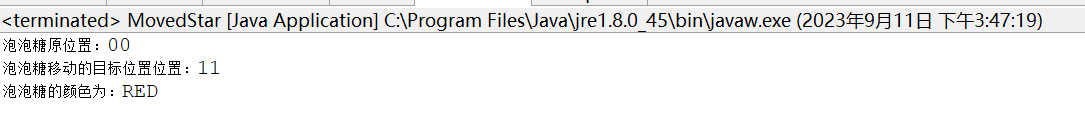
该任务用于创建一个用于描述【待移动泡泡糖】的实体类，该类不单具有泡泡糖（Star）的所以属性，还有自己的独立属性：待移动位置（坐标）。可知，该类与Star类之间具有继承关系

**5.1.2代码实现：**

****

**5.1.3验证与测试：**

****

****

**5.2实体类的文本化输出函数**

**5.2.1任务描述：**

该任务为显示完整的泡泡糖属性数据，以简化泡泡糖的测试语句（利用继承的优势）。具体为给Position,Star，MovedStar类型分别添加toString函数的重写代码。

**5.2.2代码实现：**

**Position类:******

**Star类：**

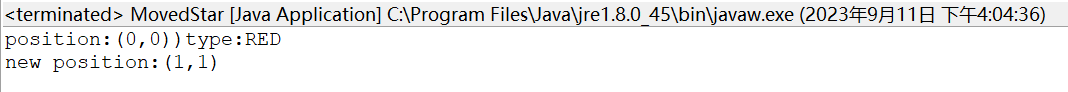
****

**MovedStar类：**

****

**5.2.3验证与测试：**

****

****

**5.3实现移动泡泡糖封装**

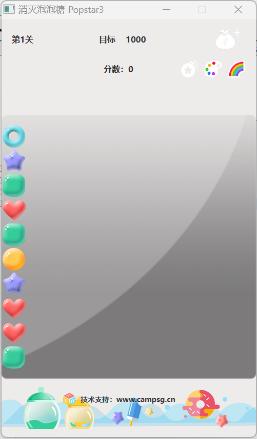
**5.3.1任务描述：**

本任务用于计算并获取垂直方向的【待移动泡泡糖】。因为该场景只涉及单列泡泡糖，所以只需判断固定数量与颜色的泡泡糖列表即可。

点击指定泡泡糖后执行tobeClearedStars函数获取【同色】【待消除泡泡糖】集合，然后从界面传递的【完整泡泡糖列表】中调用getYMovedStars获取无需消除的泡泡糖并将这些对象封装为【待移动泡泡糖】。getYMovedStars负责获取【待移动泡泡糖】然后返回到游戏界面

**5.3.2代码实现：**

****

**5.3.3验证与测试：**

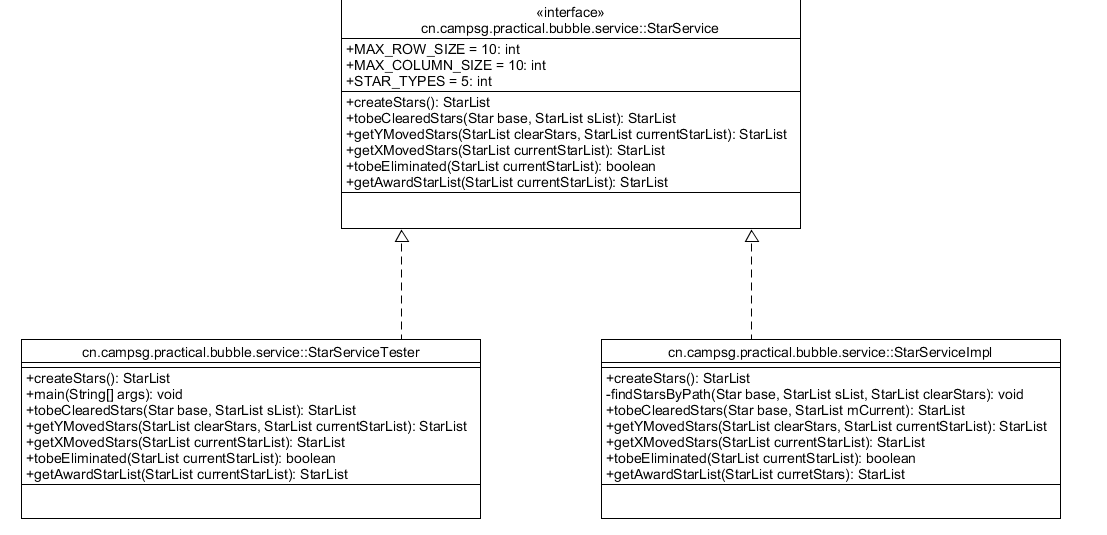
第六章 体验接口解耦特性

**需求分析：**在企业上，开发功能的实现往往需要多人共同参与，每人负责制定的模块，该任务将体验企业级开发流程。界面开发人员和业务开发人员是两个独立但又同步进行的过程，因此界面开发人员缺乏真是实现验证界面代码正确性，所以需要系统设计人员编写一个【业务测试类】以对界面的代码进行验证，完成验证后，才能切换到真实环境下测试。

**功能分析：** 需要设计一个满足业务需求的接口（该任务已给出,StarService类），编写测试业务类和真实业务类，通过接口调用他们的测试矩阵。

**业务流程和架构分析设计：**设计测试业务StarServiceTester类和真实业务StarServiceImpl类，分别重写StarService接口的createStars方法，将生成的StarList泡泡糖的矩阵通过initGameStars传到游戏主界面MainForm上，显示调用接口的不同类的实现。

涉及UML图：



**6.1创建服务测试类**

**6.1.1任务描述：**

扮演界面开发人员，创建业务测试类StarServiceTester,实现StarService接口，要求实现creatStars方法，共创建5个不同颜色，坐标的泡泡糖，组成测试用泡泡糖矩阵返回游戏界面。

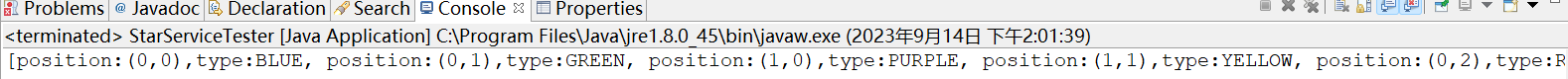
**6.1.2代码实现：**

**StarServiceTester类**

****

**6.1.3验证与测试：**

****



**6.2实现界面泡泡糖显示**

**6.2.1任务描述：**

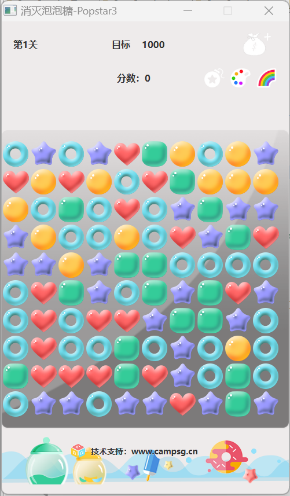
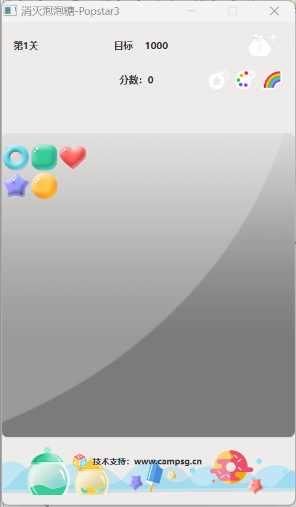
扮演界面开发人员，访问任务1创建的【测试业务类】对象，把测试类的泡泡糖矩阵显示在游戏界面上（涉及相关的javaFX知识），把【测试业务类】获取的多个泡泡糖对象Star,转换为界面显示控件Label（包含泡泡糖类型，位置坐标（x,y）属性）,并存储在界面容器中。测试完成后把【业务测试类】对象修改为【真实业务类】对象，确保显示10\*10随机泡泡糖矩阵。

**6.2.2代码实现：**

****

**6.2.3验证与测试:**

****

**【业务测试类】验证： 【真实业务类】验证：**

**6.3通过接口动态切换实现类**

**6.3.1任务描述：**

本任务在任务2的基础上，实现更高级的【动态业务类】的切换功能，能保证在不修改代码的情况下，能够实现【测试】与【真实】业务类之间的相互转换

**6.3.2代码实现：**

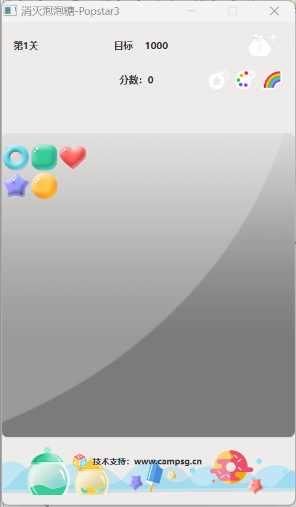
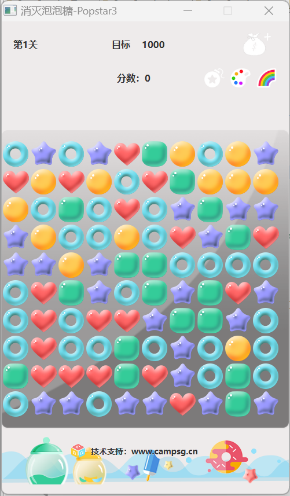
****

**6.3.3验证与测试：**

**测试业务类：******

**真实业务类;**

****

第七章 体验接口隔离性

**需求分析：**该场景是为了体验接口的隔离性，需要通过鼠标点击事件完成泡泡糖的消除功能，基于接口的隔离性，因此只需要关注鼠标的点击事件即可。

**功能分析：** 需要通过鼠标点击事件来将被点击的泡泡糖对象放入到【待消除泡泡糖】集合中然后消除对应显示控件。

**业务流程和架构分析设计：** 需要创建一个处理点击事件的类StartEventHandle并实现相关的接口EventHandler, Label初始化时，需要把【点击事件处理】对象传递给每个Label对象。该对象被多个Label公用，因此需调用StartEventHandler的handle方法来判断那个Label被点击了。然后调用StarService接口的【业务实现类】，访问tobeClearedStars函数得到【待消除泡泡糖】集合，然后清除Star对象对应的显示控件Label。

**7.1处理泡泡糖点击事件**

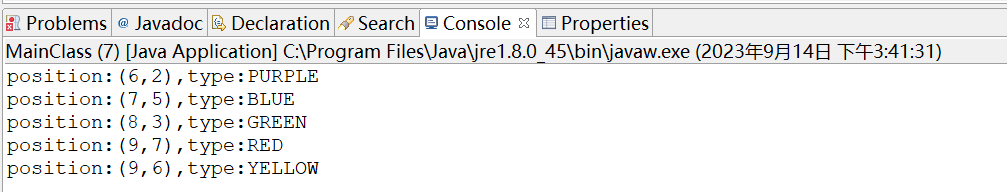
**7.1.1任务描述：**

该任务为实现处理【点击事件】的接口StartEventHandler,使其能将【被点击】泡泡置入【待消除泡泡糖】集合。然后把每个【点击事件处理】对象绑定到界面中每个泡泡糖的显示控件Label上。

**7.1.2代码实现：**

****

**7.1.3验证与测试：******

****

**7.2实现点击-消除效果(不考虑移动)**

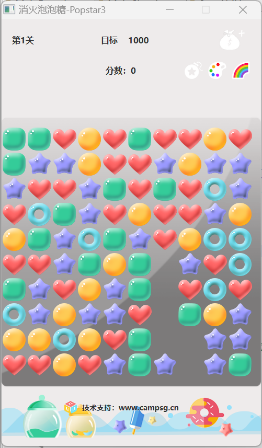
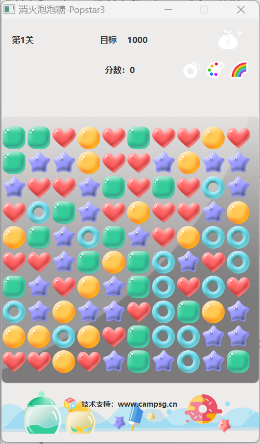
**7.2.1任务描述：**

由任务1已完成获取被点击泡泡糖的Label并转换成泡泡糖对象Star，接下来该任务通过调用服务类StarService的tobeClearedStars函数，获取所以【待消除泡泡糖】集合，然后执行消除操作

**7.2.2代码实现：**

****

**7.2.3验证与测试：**

****

第十章 移动垂直方向的泡泡糖（一）

**需求分析：** 该场景为实现界面上单列泡泡糖的消除与移动，本场景基于005场景实现了单个泡泡糖移动步长和移动个数的计算。要求，当用户点击（红色泡泡糖）时，消除【同色】泡泡糖，同时将上方剩余泡泡糖补充到底部。

**功能分析：** 当泡泡糖【被点击】时，需要获取“待消除泡泡糖”（004场景已实现）和“垂直方向待移动泡泡糖”。然后执行消除“待消除泡泡糖”操作和将“垂直移动泡泡糖”按照填补到底部的操作。

**业务流程和架构分析设计：** 被点击泡泡糖通过点击事件执行业务类的StarServiceImpl的getYMovedStar方法，然后getYMovedStar返回“垂直移动泡泡糖”的集合（包含移动距离）(场景005只获取了泡泡糖对象集合，并没有获取他们移动距离的属性)。

**涉及数据结构与算法分析：**

1. Swap函数：参数(两个需要交换的对象)。作用（用于交换两个对象的数据）。

步骤：需要创建一个临时的对象，用于暂存，将对象1的数据拷贝到临时对象，然后把对象2的数据拷贝给对象1，最后再把临时对象（对象1）的数据拷贝给对象2。这样就完成了两个对象的数据进行交换的操作。

1. Sort函数（冒泡排序）：参数（一个集合）。作用（对集合内的数据进行排序）。

时间复杂度o（n^2）,空间复杂度o（1）

步骤：

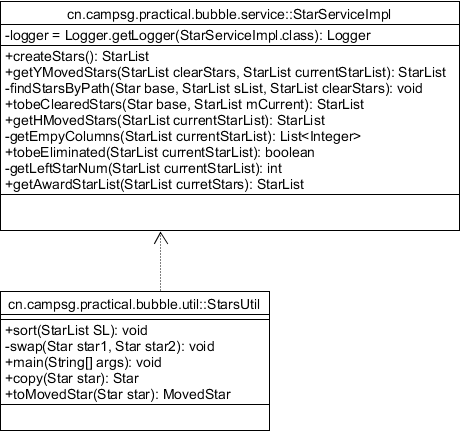
一：比较相邻的元素，前一个比后一个大（或者前一个比后一个小）调换位置。

二：每一对相邻的元素进行重复的工作，从开始到结尾，这步完成后，结尾为最大或最小的数.

三：针对除了最后一个元素重复进行上面的步骤

四：重复1-3步骤直到完成排序

**涉及UML图：**

****

**10.1交换两个泡泡糖**

**10.1.1任务描述:**

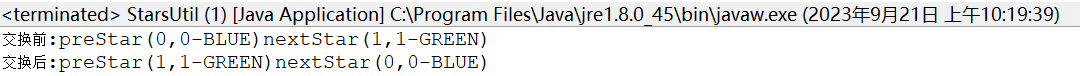
该任务为实现交换两个泡泡糖的数据（swap）函数。

**10.1.2代码实现：**

**Swap方法:******

**10.1.3验证与测试：**

****

****

**10.2泡泡糖集合的排序**

**10.2.1任务描述：**

该任务实现对“待消除泡泡糖”进行排序的sort函数（sort过程利用任务1的swap函数）

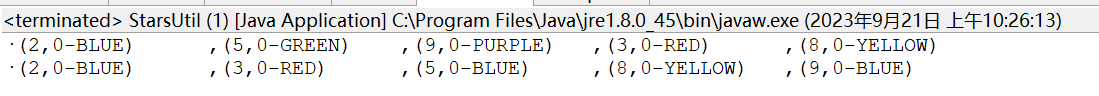
**10.2.2代码实现:**

**Sort方法：**

****

**10.2.3验证与测试:**

****

****

**10.3移动垂直方向的泡泡糖**

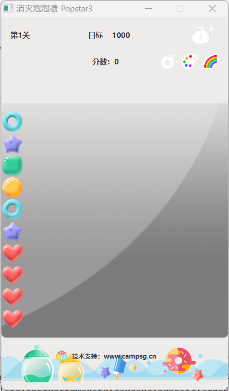
**10.3.1任务描述：**

该任务为实现通过getYMovedStars函数，获取“垂直方向待移动泡泡糖”集合在场景005中已实现固定坐标和数量的消除，该任务将为该函数添加随机位置和数量泡泡糖的消除功能。

**10.3.2：代码实现：**

**getYMovedStars函数：**

****

**10.3.3验证与测试：**

第十一章 移动垂直方向的泡泡糖（二）

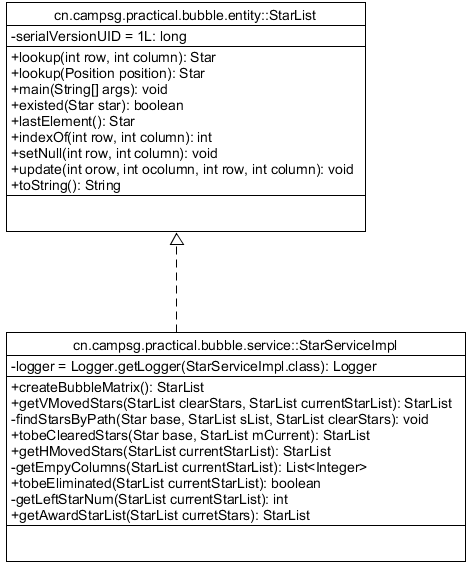
**需求分析：** 场景010实现了单列泡泡糖的消除与垂直移动“填补空隙”功能，在实现的时候，使用了StarList类的lookup和existed函数，该场景将对以上两个函数进行实现。而lookup和existed函数需要操作List类型的集合，但List集合并为提供相关函数，所以选择用StarList继承ArrayList类

**功能分析：** lookup方法需要通过行列值来获取集合中的泡泡糖元素，在实现时可以通过遍历StarList集合，直到获得的Star对象的行列值（或者Position属性）与目标相等，返回该Star对象。Existed方法则调用lookup方法，若lookup方法为获得目标对象（null）则说明不存在，返回null

**业务流程和架构分析设计：** 当泡泡糖被点击时，调用getYMovedStars函数获得“待消除泡泡糖”集合，getYMovedStars函数需要使用StarList的existed和lookup方法。

**涉及数据结构与算法分析：**顺序表：ArrayList集合中保存的是一系列连续的对象，所以可以通过从前往后遍历来获取自己所需要的目标对象。

**涉及UML图：**



**11.1根据坐标查找泡泡糖**

**11.1.1任务描述:**

该任务对ArrayList进行扩展，实现按行列值查找泡泡糖的方法。

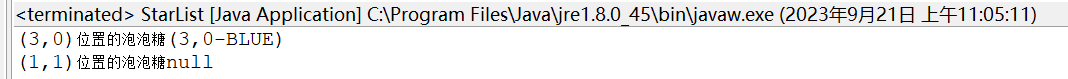
**11.1.2代码实现：**

**Lookup方法：**

****

**11.1.3验证与测试：**

****

****

**11.2根据位置查找泡泡糖**

**11.2.1任务描述：**

任务1实现了指定行列值查找的方法，该任务将实指定Position属性的查找方法（重载）

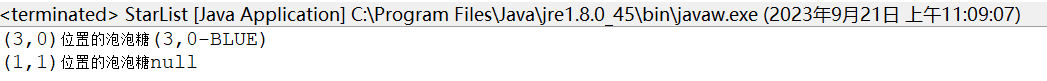
**11.2.2代码实现：**

**Lookup方法**

****

**11.2.3验证与测试：**

****

****

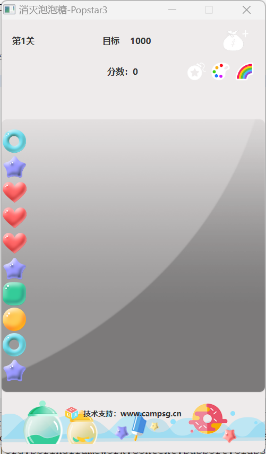
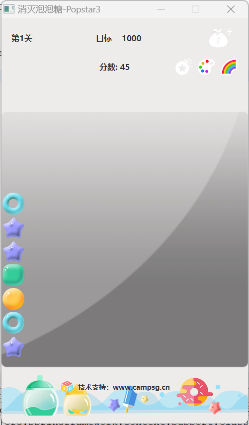
**11.3判断泡泡糖是否存在**

**11.3.1任务描述：**

利用任务2的lookup函数，实现判断泡泡糖是否存在于集合的函数existed

**11.3.2代码实现：**

****

**11.3.3验证与测试：**

第十二章 移动垂直方向的泡泡糖（三）

**需求分析:**场景010实现了单列“待移动泡泡糖”的移动，该场景将实现对多列“待移动泡泡糖”的移动并确保移动顺序。要求通过多次调用场景010的移动单列来移动多列

**功能分析：** 循环遍历“待移动泡泡糖”的列集合，通过对每列“待移动泡泡糖”不仅进行垂直移动计算，还要求计算水平移动距离。

**业务流程和架构分析设计：** 当泡泡糖【被点击】时，调用getYMovedStaras方法，利用StarsUtil的group方法进行单列分组，将每一个“待消除泡泡糖”集合按照列进行排序，以却界面按从左向右，从上向下的顺序移动泡泡糖。然后遍历场景010的单列移动，并赋予新的坐标（原坐标column+moved）

**涉及数据结构与算法分析：（**1）进行了二维的排序，一维的排序和场景010的sort函数原理一样，当一维的数值相等时，进入二维排序，排序原理仍和场景010的sort一样，直到一维的数值不相等。

（2）HashMap:hashmap是用来进行分组的功能，它可实现按指定标志，进行链性散列分组。他包含两个数值Map<Key泛型，Value数值>在该场景中，Key为整形泛型，Value为StarList集合，这样可以实现按“待消除泡泡糖”集合的列进行分组处理，方便在排序时进行调用。在调用时，由于HashMap中链表的元素不能直接调用修改，所以需要用到一个迭代器Iterator来进行元素的读取，方法为获取Map的Key值后调用Iterator方法。

**12.1更新集合的排序算法**

**12.1.1任务描述：**

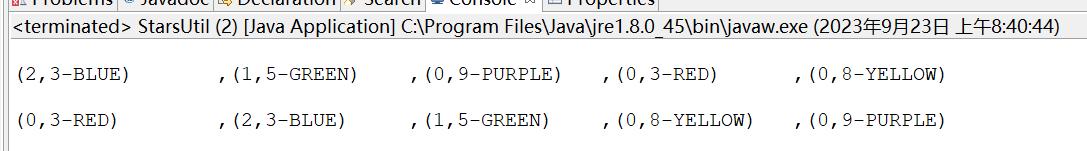
该任务为更新在场景010实现的sort函数，使其能够实现由单列排序到多列排序。

**12.1.2代码实现：**

****

**12.1.3验证与测试：**

****

****

**12.2待消除泡泡糖的排序与分组**

**12.2.1任务描述：**

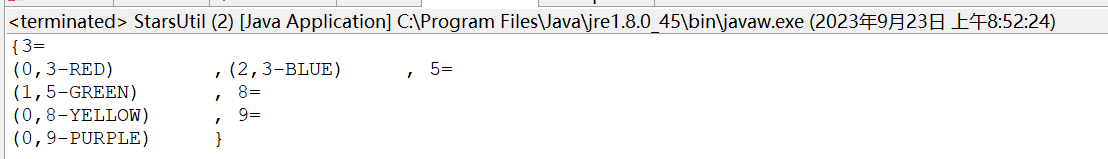
为了确保界面能按照从左向右，从上到下的顺序移动所有“待移动泡泡糖”，需要对已经排序完的“待消除泡泡糖”集合进行分组，使其按列分成多个泡泡糖集合，每个集合中“待消除泡泡糖”行号不同，列号相同。

**12.2.2代码实现**

****

**12.2.3验证与测试：**

****

****

**12.3获取垂直方向待移动的泡泡糖**

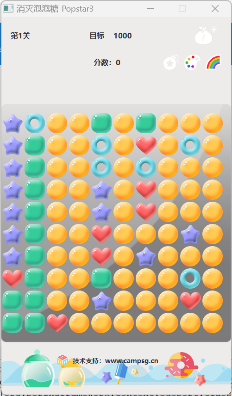
**12.3.1任务描述：**

该任务在场景010实现获取单列“垂直移动泡泡糖”集合的基础上，实现多列“垂直移动泡泡糖”集合的获取，要求对其进行按列分组。先获取“待消除泡泡糖”集合，将其封装为“垂直方向待移动泡泡糖”（封装单列“待移动泡泡糖”在场景010已实现）。

**12.3.2代码实现：**

**getYMovedStars方法：**

****

**12.3.3验证与测试：**

第十三章 移动水平方向的泡泡糖

**需求分析：**该场景在012能够移动垂直方向的泡泡糖的基础上，进行水平移动。

**功能分析：**要求获取“待消除泡泡糖”的按列分组的集合，然后遍历列值，判断是否有空列，若有空列，则后面的不为空列的待移动列值+1。

**业务流程和架构设计分析：**当界面通过tobeClearedStars函数获取“待消除泡泡糖”集合后，通过getYMovedStars函数，将垂直方向待移动泡泡糖返回给界面，然后调用getXMovedStars函数获取水平移动的泡泡糖。

**13.1获取被整列清空的泡泡糖集合**

**13.1.1任务描述：**

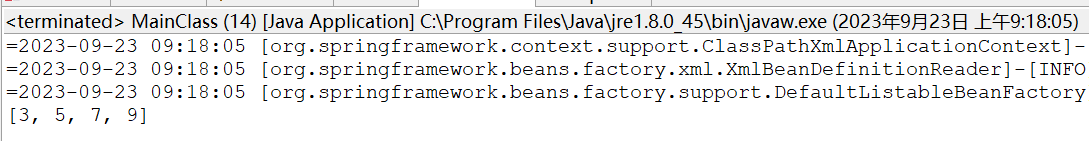
在进行清除操作后，会出现空列的情况，需要右边不为空列的泡泡糖集合进行列平移。该任务为获取“被清空所有泡泡糖的【列】”集合。

**13.1.2代码实现：**

****

**13.1.3验证与测试：**

****

****

**13.2获取水平待移动泡泡糖**

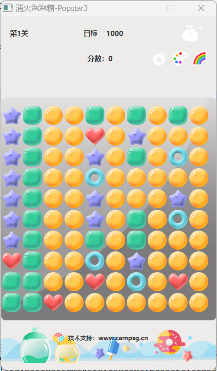
**13.2.1任务描述：**

利用任务1获取的“被清空列”集合，获取水平待移动的泡泡糖数量和移动距离

**13.2.2代码实现：**

****

**13.2.3验证与测试：**

****

第十四章 更新关卡通关分数

**需求分析：**该场景为了实现通关后，对下一通过关卡分数的获取与计算，对与游戏相关分数的信息由设计人员保存在了配置文件的score,conf中，要求获取配置文件的分数信息，并且能给根据当前的关卡数来计算下一关卡的分数并呈现在游戏界面上。

**功能分析：**要求能够通过IO输出流来进行文件的读取，保存在一个用于计算分数的类中，然后通过获取当前的关卡数来进行相关的数学计算。得出下一个关卡的通关分数，并呈现在游戏的主界面上。

**业务流程和结果设计分析：**由分数读取类Configuration同过IO流来获取score.conf里的关于score的相关信息，并保存在Score类，并设计Score类中的相关方法来进行数学计算。通过业务类的ScoreServiceImpl的nextScoreByLevel方法来获取当前的关卡数和Score类的getScore方法，计算出下一个关卡的分数返回到游戏主界面MainForm的changeLevel方法来进行分数的呈现。

**相关数学计算情况：**

1.每关的通关目标分 - levelScore : 默认第一关，之后随关卡变化而更新

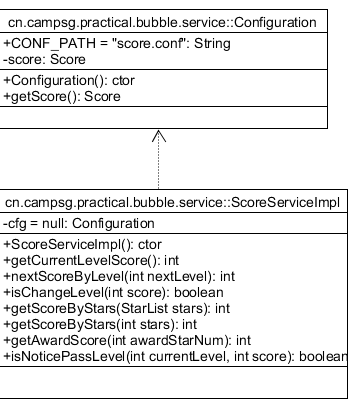
2.每关的通关递增分-step : 该分数表示关卡切换时，每关通关目标分递增值

3.难度追加递增分-increment：该分值每隔固定关卡会增加一倍

4.难度追加关卡间隔数-length:表示每隔多少关卡，难度增加

****

**涉及UML图：**

****

**14.1获取初始通关分数**

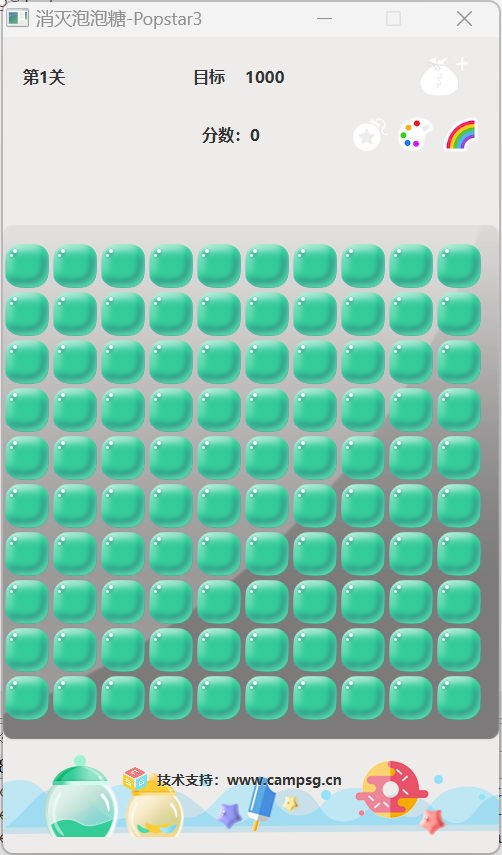
**14.1.1任务描述：**

当前任务为获取第一关的通关目标分，该分数保存在配置文件中，要求动态读取配置文件中的分数

**14.1.2代码实现：**

**Configuration类**

****

**14.1.3验证与测试：**

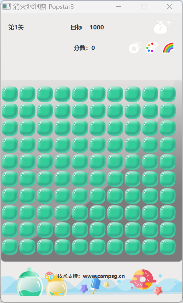
**14.2更新关卡通关分数**

**14.2.1任务描述：**

该任务在任务1获取配置文件中Score相关数据后，实现通关后，计算并显示下一关的分数，并更新Score的levelScore属性。

**14.2.2代码实现：**

****

**     14.2.3验证与测试：**

第十五章 实现泡泡糖的积分规则

**需求分析：**该场景将实现ScoreService接口的三个功能函数，getScoreByStars，getAwardScore和isNoticePassLevel,isChangeLevel。其功能分别为，根据被消除的泡泡糖计算消除得分，根据剩余泡泡糖数量结算奖励，界面显示通过提示语句。

**功能分析：**（1）getScoreByStars,d该方法首先需要获取【已消除泡泡糖】数量，场景004已实现，然后根据场景014得到的Score类中的相关数据，计算得分返回到游戏界面

（2）getAwardScore,从游戏界面获取剩余泡泡糖数量（总泡泡糖数量-被消除泡泡糖数量），然后根据场景014得到的Score类中的相关数据计算剩余泡泡糖的奖励分。

（3）isChangePassLeevel,从以上两个方法活动通过分数总和后，通过该方法判断是否已达到通关目标分，如果通过，则执行isNoticePassLevel判断是否已给出通过提示，若没给过，则弹出通关提示。

直到无可消除泡泡糖为止，每一步重复以上步骤。

**业务流程和结果设计分析：**当泡泡糖【被点击】时，获取以消除泡泡糖传递给业务类getScoreByStars方法获取消除分数，获取剩余泡泡糖数传递给业务类getAwardScore方法，获取Score类的当前得分和关卡传递给业务类isChangeLevel方法判断是否已达到通过目标分若达到则调用isNoticePassLevel方法判断是否已给出过通过提示。然后将结果返回到游戏主界面。

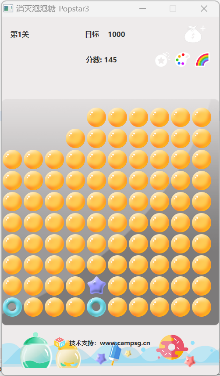
**15.1获取消除奖励分数**

**15.1.1任务描述：**

要求根据消除的泡泡糖个数，计算单词消除时所获取的奖励分。计算规则为：得分=单个泡泡糖得分\*消除个数^2

**15.1.2代码实现：**

****

**15.1.3验证与测试：**

**15.2获取结算时奖励分数**

**15.2.1任务描述：**

该任务当界面无可消除泡泡糖时，若剩余数量小于【限定值】，则根据剩余泡泡糖数进行结算奖励。

**15.2.2代码实现：**

****

**15.2.3验证与测试：**

**15.3显示通关提示**

**15.3.1任务描述：**

该任务为当游戏积分达到通关目标分时，判断是否在游戏界面显示通关提示语句。

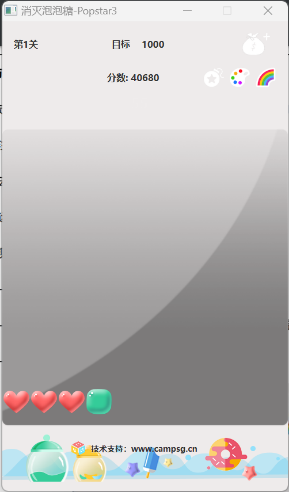
15.3.2代码实现：

**IsChangeLevel方法：**

****

**IsNoticePassLevel方法：**

****

**15.3.3验证与测试：**

**第十六章 遇到问题解决**

**16.1枚举类型：**

一开始不知道什么是枚举类型，应该如何去定义，后来经过上网查找和查看java的解释文档，了解了如何去创建枚举类型。它是通过enum来去定义的，包含一个或者多个枚举常量（枚举值），他们通常带有数值，在枚举类型里可通过value和valueOf等方法来进行枚举值和数值之间的转换。

**16.2HashMap:**

这个了解听说过，但不知道在java中的具体用法，上网查询了使用说明后，其他基本用法就已经了解了。它是一种映射的集合，通过键值对的形式来保存数据的一个方式。可以轻松的实现数据的分类和查找

**16.3BufferedReader:**

BufferedReader 是一个缓冲字符输入流，该流**可以对 FileRead 进行包装**，提供一个缓存数组，一次将数据按照一定规则读取到缓存区中，通过这个方法可以读取到指定目录的文件，且能更有效的读取字符，数组和行。

**第十七章 参考文献**

**参考文献：**

[1]:[GitHub - gagle/node-bufferedreader: Binary and event-based data buffered readers.](https://github.com/gagle/node-bufferedreader) <https://github.com/gagle/node-bufferedreader>

[2]:Java的相关API文档（HashMap和BufferedReader）

[3]: Java中BufferedInputStream和BufferedOutputStream基本使用详解,

<https://blog.csdn.net/lyb1832567496/article/details/52727862>

[4]: 深入理解HashMap：实现原理，使用方法及优化技巧<https://zhuanlan.zhihu.com/p/622919857>

[5]:

[GitHub - cornelk/hashmap: A Golang lock-free thread-safe HashMap optimized for fastest read access.](https://github.com/cornelk/hashmap) https://github.com/cornelk/hashmap