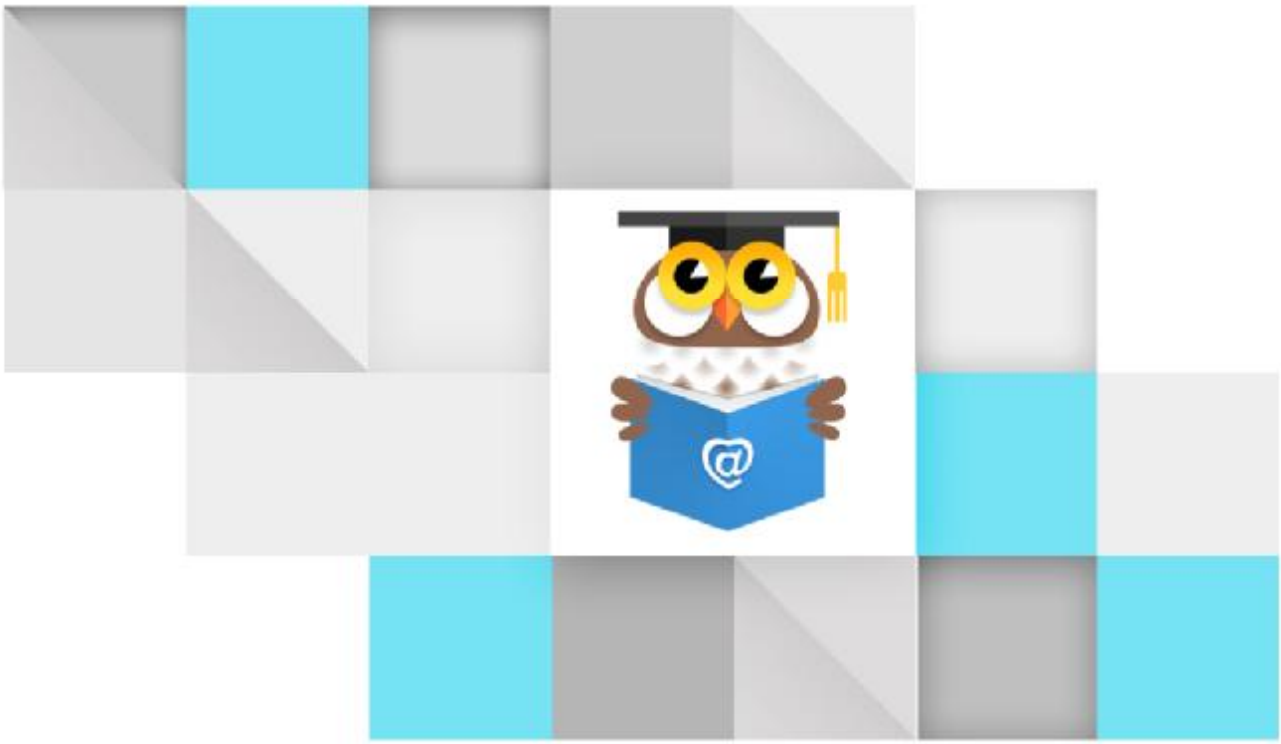
消灭泡泡糖（Java）

实训指导手册

实训场景 005 – 封装待移动的泡泡糖



Campus Solution Group

目 录

一、任务编号：PRJ-BU2-JAVA-005

1、实训技能

- **Java** 面向对象编程技能

2、涉及知识点

- 类的扩展
- 类的继承语法
- 子类对象实例化与构造器
- **super**
- 类型转换
- 覆盖与多态
- 认识 **Object** 类
- **toString** 方法使用与继承

3、实现效果

图 3-1

4、场景说明

1、业务说明：

场景**PRJ-BU2-JAVA-004**已经实现了：根据【被点击】泡泡糖获取四周所有【同色】泡泡糖的功能，所有【同色】泡泡糖均被定义为【待消除泡泡糖】并保存在**StarList**。

本场景将继续深入开发《消灭泡泡糖》游戏功能，用于初步实现：获取【待移动泡泡糖】的功能，这些对象将会在【同色】泡泡糖消除后，执行【移动】操作并填补【同色】泡泡糖消除后留下的“空隙”（本场景仅考虑单列固定颜色的泡泡糖，如下图）。

图 **4-1**

2、实现思路：

2-1. 从图**4-1**可知，本场景主要用于获取所有【待移动的泡泡糖】，为了完成该业务功能，我们需要创建一个全新的实体类型：**MovedStar** – 待移动泡泡糖。该实体类不但具有泡泡糖

（**Star**类型）的所有特性（位置、类型），同时还具有自己独立的属性：**待移动位置（坐标）对象**。

2-2. 在图**4-1**中，当用户点击了【倒数第二个】或【倒数第三个】红色泡泡糖，我们可以通过调用场景**PRJ-BU2-JAVA-004**完成的**tobeClearedStars**函数获取【同色】【待消除泡泡糖】集合（从图**4-1**中可见共获得**2**个红色泡泡糖）。

2-3. 随后，我们将通过调用本场景新建的 “**getYMovedStars**”方法计算并获取所有【待移动泡泡糖】（从图**4-1**中可见共获得**7**个待移动泡泡糖）。

2-4. 把获取的【待移动泡泡糖】返回给界面，界面将依次【消除】泡泡糖，并【移动】未消除泡泡糖填补“空隙”。

3、核心组件介绍：

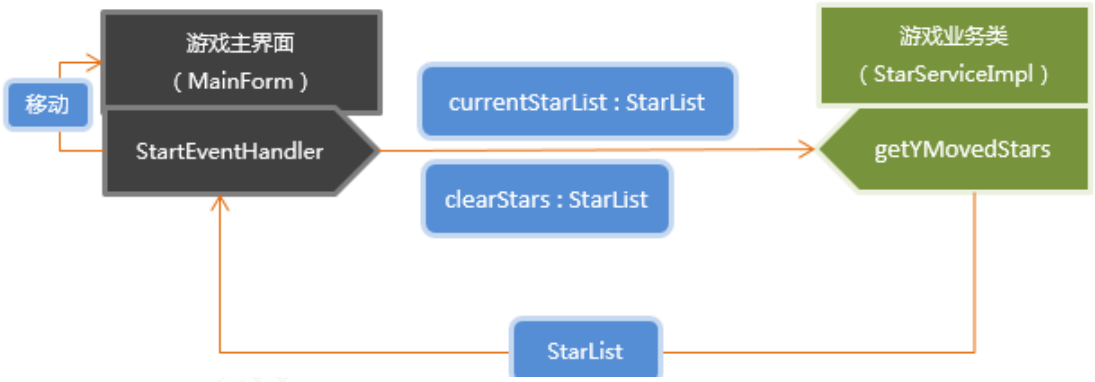


图 **4-2**

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、得分计算等业务操作。

3-3. MovedStar – 待移动泡泡糖。

该类型继承于泡泡糖类型（**Star**），它不但具有泡泡糖（**Star**类型）的所有特性（位置、

类型），同时还具有自己独立的属性：**待移动位置（坐标），如图4-2所示。**

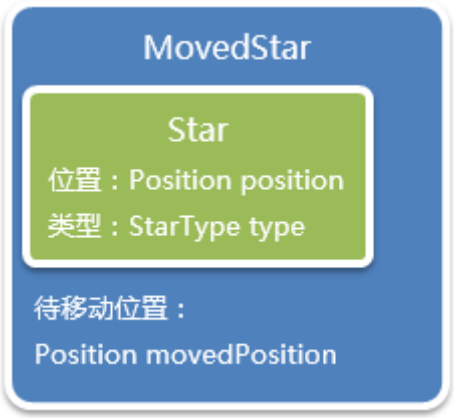


图 **4-3**

3-4. StarServiceImpl类中的getYMovedStars：

3-4.1. 在本场景中，该方法将会根据场景**PRJ-BU2-JAVA-004**获取的【待消除泡泡糖】列表，循环计算【垂直方向】存在多少用于填补“空隙”的【待移动泡泡糖】。

3-4.2. 为降低开发难度，逐步开发《消灭泡泡糖》的游戏业务，本场景并不要求按真实业务判断随机产生的**10 * 10**泡泡糖矩阵，只需判断单列固定数量与颜色的泡泡糖列表即可，后续场景会依次补全该函数，最终实现真实业务。

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-004** – 获得待消除泡泡糖

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（类、方法调用、实例化对象）。

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_005**

2、 进入开发环境：

详见**SPOC**平台上《**PRJ-BU2-JAVA-005** 前置任务：进入开发环境》

图 **5-1**

6、任务 1 – 创建待移动泡泡糖实体类

1、任务描述：

1-1. 创建一个用于描述【待移动泡泡糖】的实体类。

1-2. 待移动泡泡糖不但具有泡泡糖（**Star**类型）的所有特性（位置、类型），同时还具有自己独立的属性：**待移动位置（坐标）**。

1-3. 从**1-2**可知，【待移动泡泡糖】的属性扩展了【泡泡糖】的属性，因此两个实体类之间存在继承关系。

1-4. 本任务将基于以上思路实现实体类之间的业务继承（扩展）。

2、推荐步骤：

2-1. 定位到包：**cn.campsg.practical.bubble.entity**。

2-2. 新建一个类**MovedStar**并继承类**Star**。

2-3. 定义一个私有**Position**属性描述**待移动位置（坐标）**，并为其创建**get**和**set**访问器。

2-4. 新建带参构造器，该构造器拥有三个参数：

2-4.1. 参数**1**：泡泡糖原始坐标对象（**Position**）。

2-4.2. 参数**2**：泡泡糖的类型（**StarType**枚举）。

2-4.3. 参数**3**：将要移动的新位置（坐标）对象（**Position**）。

2-5. 实现三参构造函数。

2-5.1. 对泡泡糖**原始坐标和类型**赋值：直接调用父类**Star**中的带参构造器。

小心：调用时切勿将**MovedStar**构造体的**Position**参数直接传入父类构造体。

建议：新建一个**Position**对象，并设置该对象行值与列值，再传递给父类构造体。

2-5.2. 把“将要移动的新位置（坐标）”对象赋值给本类的私有属性。

+ **提示：**子类调用父类的构造函数可以使用**super**关键字。

3、验证与测试：

3-1. 在当前类中创建一个**main**函数，作为方法测试的程序入口。

3-2. 测试前，请先准备三个测试数据：

3-2.1. 新建一个**Position**对象**(0,0)**，用于描述初始坐标。

3-2.2. 新建一个**Position**对象**(1,1)**，用于描述将要移动的新位置（坐标）。

3-2.3. 新建一个**StarType**类型并设置为红色，用于描述泡泡糖的颜色。

3-3. 通过以上参数实例化一个**MovedStar**类型的对象，并调用三参构造体。

3-4. 向控制台输出初始坐标、移动坐标和泡泡糖类型。

3-5. 控制台输出结果应保证与下图一致：

```
泡泡糖原位置：(0,0)
泡泡糖移动的目标位置位置：(1,1)
泡泡糖的颜色为：RED
```

图 **6-1**

7、任务 2 – 实体类的文本化输出函数

1、任务描述：

1-1. 本任务将利用面向对象三要素中的多态重写特性体验实体继承的优势。

1-2. 本任务需要实现：当通过控制台打印语句打印【待移动泡泡糖】时，能够显示完整的泡泡糖属性数据，以此简化泡泡糖的测试语句。

1-3. 任务实现思路如下：

1-3.1. Java语法规定任意类型的父类均**Object**，当前场景我们使用**MovedStar**继承**Star**类型，而**Star**类型的父类默认就是**Object**类型。

1-3.2. Object类型中拥有**toString**函数，当控制台打印语句打印某个对象时，会自动调

用**toString**方法。

1-3.3. 实现本任务的最佳方法是：为**Position**、**Star**、**MovedStar**类型分别添加**toString**函数的重写代码。

2、推荐步骤：

2-1. 定位包：**cn.campsg.practical.bubble.entity**。

2-2. 重写**Position**类型的**toString**方法

2-2.1. 利用**Eclipse IDE**快捷功能重写**Object**类的**toString**函数（见提示）。

2-2.2. 在重写的**toString**函数中定义一个字符串变量存放输出内容。

2-2.3. 字符串变量存放的内容格式为：“**position**：（行属性，列属性）”。例如：**position**：（**0,0**）

2-2.4. 返回该字符串变量（不使用**Eclipse**自动生成的返回代码）。

+ 提示：重写方法快捷操作：**Alt + Shift + S -> Override/Implement Methods…..**

在出现的对话框内勾选：**toString**函数。

2-3. 重写**Star**类型的**toString**方法。

2-3.1. 利用**Eclipse IDE**快捷功能重写**Object**类的**toString**函数。

2-3.2. 在重写的**toString**函数中定义一个字符串变量存放输出内容。

2-3.3. 该字符串的内容格式为：

“**position**：（行属性，列属性），**type**：泡泡糖**type**枚举值”

例如：**position**：（**0,0**），**type**：**BLUE**

2-3.4. 返回该字符串变量（不使用**Eclipse**自动生成的返回代码）。

+ 提示：Star类型的**toString**函数需要显示内容的前半部分：“**position**：（行属性，列属性）”在**Position**类型中已经实现，因此，只需调用**Position**的**toString**并将返回内容拼

接到输出内容的后半部分“，**type**：泡泡糖**type**枚举值”即可。

2-4. 重写**MovedStar**类型的**toString**方法

2-4.1. 利用**Eclipse IDE**快捷功能重写**Object**类的**toString**函数。

2-4.2. 在重写的**toString**函数中定义一个字符串变量存放输出内容。

2-4.3. 该字符串的内容格式为：

“ **position**：（行属性，列属性），**type**：泡泡糖**type**枚举值

new position：（新位置的行属性，新位置的列属性）”

注意：需显示两行字符，例如：

position：（**0,0**），**type**：**BLUE**

new position：（**1,1,**）

2-4.4. 返回该字符串变量（不使用**Eclipse**自动生成的返回代码）。

+ 提示：Star类型的**toString**函数需要显示内容的前半部分：“**position**：（行属性，列属性），**type**：泡泡糖**type**枚举值”在**Star**类型中已经实现，因此，只需调用**Star**的**toString**方法获取输出内容，并将 “**new position**：（新位置的行属性，新位置的列属性）” 拼接到输出内容的后半部分即可。

3、验证与测试：

3-1. 找到任务**1**在**MovedStar**类中测试**main**函数，删除原有的测试代码。

3-2. 添加当前场景的测试代码：

3-2.1. 创建一个从（**0,0**）到（**1,1**）的红色待移动泡泡糖对象。

3-2.2. 通过打印语句直接将上述对象打印到控制台。

3-3. 运行**main**函数，观察控制台输出是否与下图一致：

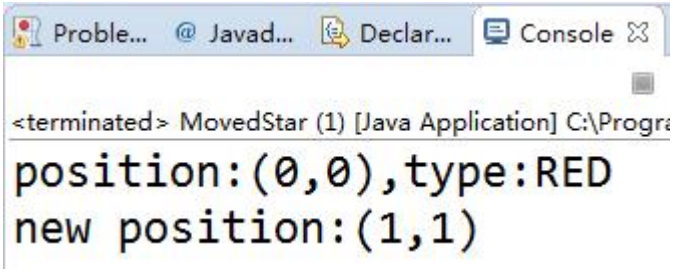


图 **7-1**

8、任务 3 – 实现移动泡泡糖封装

1、任务描述：

1-1. 本任务用于计算并获取垂直方向的【待移动泡泡糖】，根据场景说明可知，当前任务只需判断单列固定数量与颜色的泡泡糖列表即可。

1-2. 计算并获取垂直方向的【待移动泡泡糖】步骤如下：

1-2.1. 首先，当用户点击了【倒数第二个】或【倒数第三个】红色泡泡糖，游戏界面会调用**tobeClearedStars**函数获取【同色】【待消除泡泡糖】集合（本场景一共可获得**2**个红色泡泡糖），随后界面执行清除操作。

1-2.2. 其次，我们需要依次从界面传递的【完整泡泡糖列表】中获取**7**个无需消除的泡泡糖，并将这些对象封装为【待移动泡泡糖】。

1-3. getYMovedStars负责实现获取【待移动泡泡糖】的业务，该函数共包含三个参数，他们的含义分别为：

1-3.1. StarList clearStars参数 – 【待消除泡泡糖】集合（如图**7-1**）：

【被点击】泡泡糖四周的【同色】【待消除泡泡糖】集合。

该集合由场景**PRJ-BU2-JAVA-004**实现的**tobeClearedStars**函数创建。

1-3.2. StarList currentStarList 参数 – 界面【完整的泡泡糖列表】（如图**7-2**）：

界面单列共**10**个泡泡糖矩阵对应的【完整泡泡糖列表】。

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

图 **8-1**

图 **8-2**

1-3.3. 返回值**StarList -** 【待移动泡泡糖】集合（如图**8-3**）：

根据场景**PRJ-BU2-JAVA-004**获取的【待消除泡泡糖】列表，循环计算【垂直方向】存在多少用于填补“空隙”的【待移动泡泡糖】。该集合中保存的所有泡泡类型均为：**MovedStar**。

图 **8-3**

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**getYMovedStars**的注释处。

2-1.2. 新建一个【待移动泡泡糖】集合对象（**StarList**），用来存放所有待移动对象。

2-2. 定义一个存放【列值】的变量，初始值为**0**（当前场景仅考虑最左侧单列泡泡糖）。使用**for**循环，获取所有无需消除的泡泡糖（数据来源：界面传递的【完整泡泡糖列表】）。

2-2.1. 从界面传递的【完整泡泡糖列表】对象中依次获取无需消除的泡泡糖对象。

+ 提示

1）由于本场景仅考虑最左侧单列固定颜色的泡泡糖，因此循环范围可以简单的设置为：

从**6 ~ 0**依次递减（其中**6**为待移动泡泡糖的最大行号，**0**为待移动泡泡糖的最小行号）。

注意：以上循环封装待移动泡泡糖的开发思路仅适合本场景，真实业务中的循环范围与开发思路需要重新设计，后续场景将逐层依次调整开发思路以完成真实游戏业务。

2）从界面传递的【完整泡泡糖列表】对象：

getYMovedStars的第二个参数 **- StarList currentStarList**

3）从【完整泡泡糖列表】中获取对象，可以调用**StarList**的自定义函数：**lookup**

该函数的行参数为循环变量，列参数固定为初始值 **= 0**的列变量。

2-2.2. 将获取的**Star**对象封装成待移动的泡泡糖对象**MovedStar**：

1）封装可考虑利用**MovedStar**的带参构造器。

2）原始泡泡的位置、类型属性不变。

3）待移动位置的列属性 **=** 原始泡泡糖位置对象中的列值。

4）待移动位置的行属性 **=** 原始泡泡糖位置对象中的行值 **+ 2**。

+ **提示：**由于当前场景的泡泡糖颜色和数量是固定的，因此仅**2**个红色泡泡糖可以消除，所以【待移动泡泡糖】的行移动距离始终为**2**。

注意：以上移动距离计算的开发思路仅适合本场景，真实业务中的移动距离计算思路需要重新设计，后续场景将逐层依次调整开发思路以完成真实游戏业务。

2-2.3. 把封装好的对象添加到【待移动泡泡糖】列表中（**StarList**）。

2-3. 向游戏界面返回【待移动泡泡糖】列表。

3、验证与测试：

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

3-2. 点击【倒数第二个】或【倒数第三个】红色泡泡糖。

3-3. 游戏界面上两个红色的泡泡糖会直接消除，顶部**7**个泡泡糖会依次向下移动**2**格。

图 **8-4**

9、场景总结

Q1. 在您的项目中哪里用到了继承？请结合项目说明何时使用继承比较妥当？

1. 继承一般有以下三种情况：

1-1. 继承**JDK**或第三方组件类，实际项目中大多是为了修改父类函数（因父类函数无法满足要求）。

例如：**Java**的**JDK**为程序员提供了集合，集合可以存放任意数据，但当前游戏中，集合需要存放泡泡糖，**JDK**的集合并不是为**"**泡泡糖**"**而生的，所以我们编写了**StarList**继承了**JDK**的集合，并修改了**List**中无法满足要求的函数（例如：**indexOf**等），如感兴趣可以查看**StarList**类。

1-2. 继承**JDK**或第三方组件类，完全获取父类的功能。

例如：当前游戏中，还记得《消灭泡泡糖》的界面吗？我们无需开发如此复杂的**Windows**窗体界面，只需要在代码中继承**JDK**为我们提供的**Application**类就可以实现一个完整窗体。

1-3. 如本场景，实现类间属性扩展。

说明：**Star**具有两个描述属性：位置，颜色（类型），当**Star**因为消除操作需要移动时，**Star**将产生一个新位置，此处**MovedStar**的出现就是为了确定移动后**Star**的位置，同时它具有**Star**的所有属性。

2. 本游戏系统中具有以上三种继承适用场景的实现。