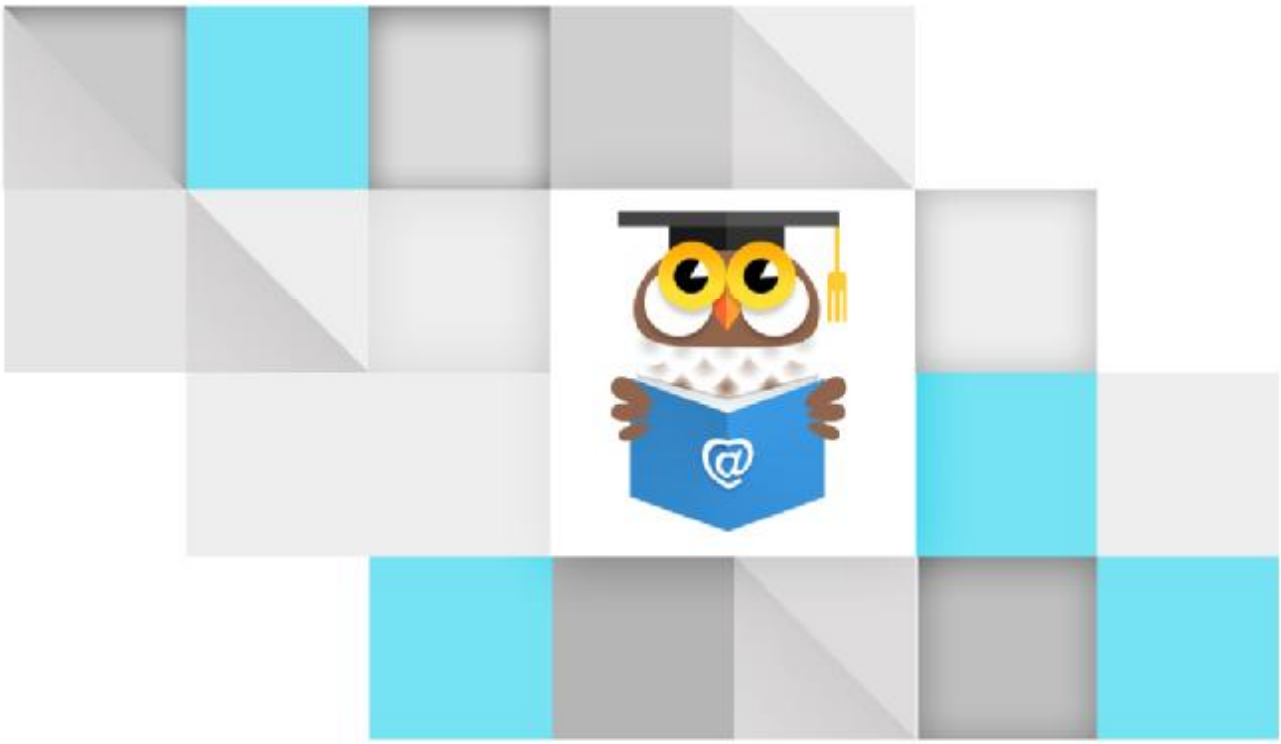
消灭泡泡糖（Java）

实训参考手册

实训场景 003 – 随机显示泡泡糖



Campus Solution Group

目 录

一、任务编号：PRJ-BU2-JAVA-003

1、实训技能

- Java 面向对象编程技能

2、涉及知识点

- 使用类定义对象
- 使用对象
- 类的成员方法
- 类的成员变量
- 构造器调用
- 循环嵌套使用
- 数学函数与数学运算
- 随机函数(*)

* 为扩展或体验用知识点

3、实现效果

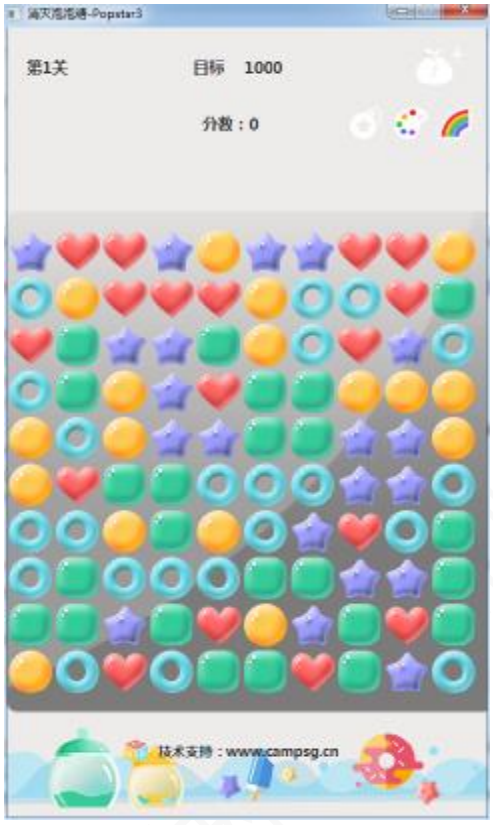


图 **3-1**

4、场景说明

1、业务说明：

在**PRJ-BU2-JAVA-002**场景中已经能做到在界面上的“指定位置显示指定颜色的泡泡糖”，本场景作为之前场景的补充，需要在界面显示一个**10*10**的随机颜色泡泡糖矩阵。

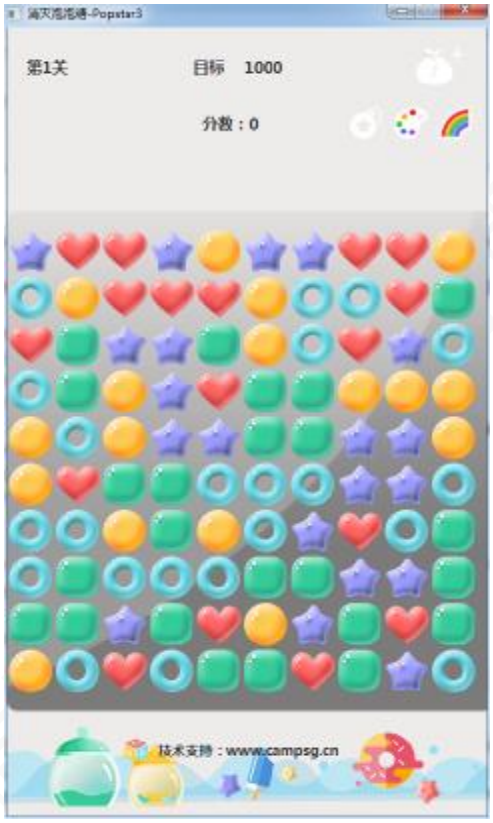


图 **4-1**

2、实现思路：

2-1. 在该业务中，我们使用**Star**类描述泡泡糖。

2-2. 使用外层**for**循环确定泡泡糖的行值。

2-3. 使用内层**for**循环确定泡泡糖的列值。

2-4. 使函数随机产生泡泡糖的类型，以达到随机的效果。

2-5. 把泡泡糖对象放到**StarList**中，再返回给界面层进行显示。

3、核心组件介绍：

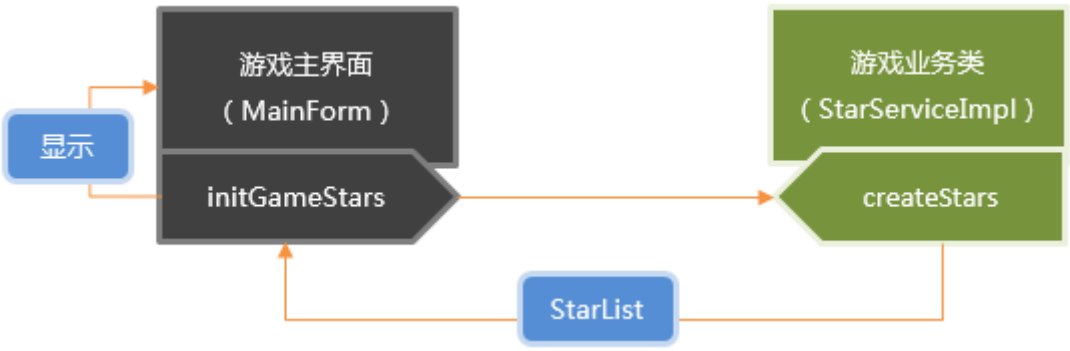


图 **4-2**

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 - 泡泡糖集合：

用于保存生成的**10 * 10**的泡泡糖矩阵，即**100**个随机颜色的泡泡糖。

3-4. StarServiceImpl类中的createStars：

3-4.1. 该方法主要负责产生**100**个随机颜色的泡泡糖，并保存在**StarList**中。

3-4.2. 产生的泡泡糖集合将会被返还到游戏界面上，游戏界面负责将**100**个随机泡泡糖以**10 * 10**矩阵的方式呈现在用户面前，详见下图：

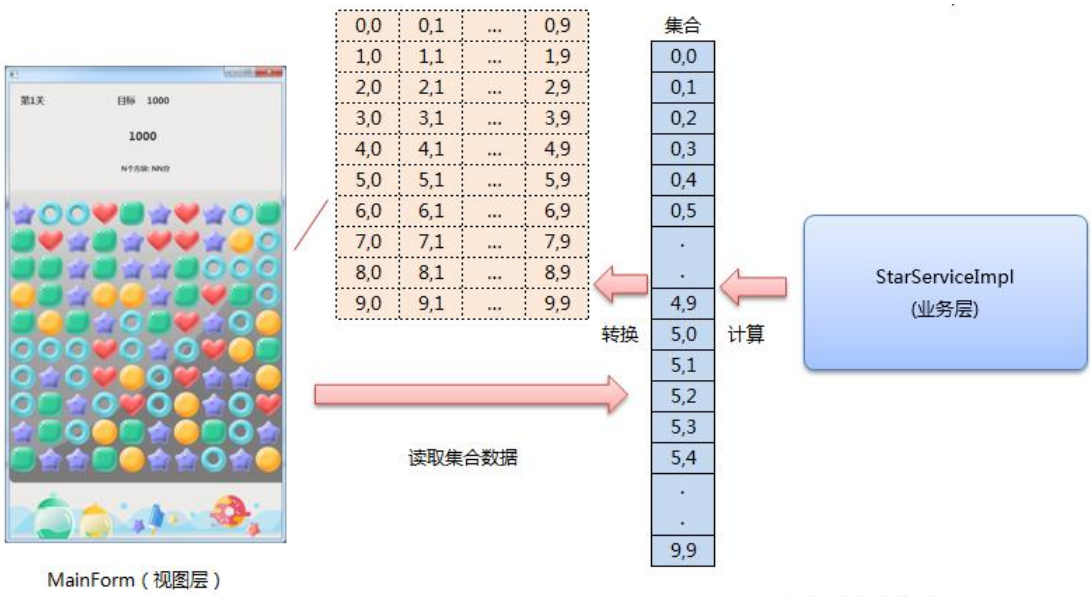


图 4-3

3-4.3. 从图**4-3**可知，**createStars**负责创建**100**个泡泡糖，垂直保存在**StarList**中，每个泡泡糖都有一个独立的坐标，游戏界面会将**StarList**中的**100**个泡泡糖转换成**10 * 10**的显示矩阵。

3-4.4. 该方法是当前场景需要重点实现的。

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-002** – 显示泡泡糖。

5-2. 必备知识与技能：

5-2.1. Java开发工具 （**Eclipse**）。

5-2.2. Java基本语法（变量、变量类型、运算符、**for**循环）。

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_003**

2、 进入开发环境：

详见**SPOC**平台上《**PRJ-BU2-JAVA-003** 前置任务：进入开发环境》

图 **5-1**

6、任务 1 – 显示一行泡泡糖

1、任务描述：

1-1. 使用**for**循环创建一行（共**10**个）泡泡糖。

1-2. 10个泡泡糖需要显示在界面的第一行。

1-3. 由于是第一行，因此**10**个泡泡糖的行号始终为**0**，我们只需对泡泡糖的列属性赋值即可。

2、推荐步骤：

2-1. 找到业务类：**cn.campsg.practical.bubble.service.StarServiceImpl**

2-1.1. 定位到方法**createStars**处。

2-1.2. 使用**for**循环创建一行共**10**个泡泡糖。

+ 提示

1）最大列值：定义在接口**StarService**中，可使用静态常量**MAX_COLUMN_SIZE**。

2）长按**Ctrl**键，点击**StarService**查看常量值的定义与注释说明。

3）**for**循环边界，**0 ~** 最大列值。

2-2. 在**for**循环中创建**Star**对象，并为**star**对象设置属性

2-2.1. 每次循环创建一个**Star**对象。

2-2.2. 设置**star**的坐标位置：坐标的行值始终为**0**，列值为循环变量。

2-2.3. 通过**star**的**type**属性为泡泡糖赋值颜色，颜色设置为红色。

2-2.4. 利用**add**方法，把泡泡糖对象添加到**StarList**列表中。

+ 提示：为泡泡糖的坐标位置赋值，可考虑使用以下赋值语句：**star.setPosition(new Position(0,**循环变量**))**。

3、验证与测试：

3-1. 定位函数入口所在类：**cn.campsg.practical.bubble.MainClass**。

3-2. 右键选择**Run As -> Java Application**运行该项目工程。

3-3. 观察是否能显示一行红心泡泡糖。

图 6-1

+ **提示：**可以使用快捷键 **Ctrl + F11** 快速运行工程项目

4、参考答案：

代码清单：**cn.campsg.practical.bubble.service.StarServiceImpl@createStars**

```
public StarList createStars() {
    StarList stars = new StarList();
    for (int col = 0; col < StarService.MAX_COLUMN_SIZE; col++) {
        Star star = new Star();

        // 设置泡泡糖在画面上的位置

        star.setPosition(new Position(0, col));
        star.setType(StarType.RED);

        // 加入列表

        stars.add(star);
    }
    return stars;
}
```

7、任务 2 – 显示 10 * 10 泡泡糖矩阵

1、任务描述：

任务1完成后只能显示一行10个红色泡泡糖，为了保证在界面能够显示10 * 10的泡泡糖矩阵，我们需要继续添加第2~10行的泡泡糖，开发流程如下：

1-1. 任务1完成的for循环用于创建第一行10个泡泡糖，它们的特性是行值=0，列值变化。

1-2. 本任务在原for循环的外层再套一个for循环，用于变化泡泡糖的行值。

1-3. 列循环每执行一次列号加1，行循环执行一次行号加1。

1-4. 当行循环、列循环都执行完毕时，创建的泡泡糖对应的坐标刚好从(0,0)到(9,9)。

1-5. 界面获取100个泡泡糖后便可显示10 * 10的泡泡糖矩阵。

2、推荐步骤：

2-1. 在任务1的循环结构外层再套一层for循环结构，用于变化泡泡糖的行值。

> **+ 提示**
>
> 1）最大行值：定义在接口StarService中，可使用静态常量MAX_ROW_SIZE。
>
> 2）长按Ctrl键，点击StarService查看常量值的定义与注释说明。
>
> 3）for循环边界，0 ~ 最大行值。

2-2. 更新泡泡糖的行坐标

2-2.1. 设置star的坐标位置：坐标的行值始终为【行循环变量】，列值为【列循环变量】。

2-2.2. star的type属性仍然为红色。

> **+ 提示：**为泡泡糖赋值，可考虑使用以下赋值语句：
>
> **star.setPosition(new Position(**行循环变量**,** 列循环变量**))**。

3、验证与测试：

3-1. 定位函数入口所在类：**cn.campsg.practical.bubble.MainClass**。

3-2. 右键选择**Run As -> Java Application**运行该项目工程。

3-3. 观察是否能显示**10 * 10**矩阵的红心泡泡糖：

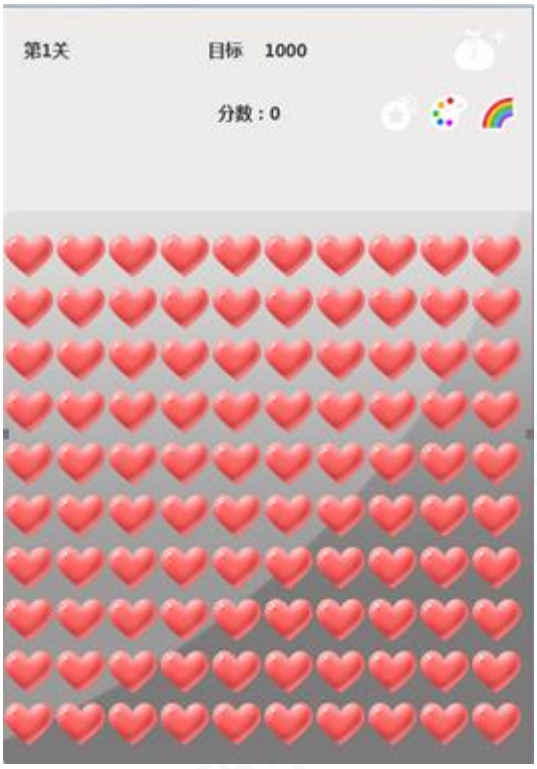


图 **7-1**

4、参考答案：

代码清单：**cn.campsg.practical.bubble.service.StarServiceImpl@createStars**

```
public StarList createStars() {
    StarList stars = new StarList();
    for (int row = 0; row < StarService.MAX_ROW_SIZE; row++) {
        for (int col = 0; col < StarService.MAX_COLUMN_SIZE; col++) {
            Star star = new Star();

            // 设置泡泡糖在画面上的位置

            star.setPosition(new Position(row, col));
            star.setType(StarType.RED);

            // 加入列表

            stars.add(star);
        }
    }
    return stars;
```

```
    }
```

8、任务 3 – 随机显示 10 * 10 泡泡糖矩阵

1、任务描述：

任务1和任务2实现了在界面上呈现10 * 10泡泡糖矩阵的效果，但当前游戏界面只能固定显示100个红色泡泡糖。

本任务主要实现对泡泡糖的颜色随机设置，以满足游戏的基本要素。

2、推荐步骤：

2-1. 删除createStars方法中对泡泡糖进行红色类型赋值的setType语句。

2-2. 在该位置使用【随机函数】获取泡泡糖的颜色，获取的数值保存到【整型】变量中。

2-2.1. 使用Math类的random函数得到随机数，并乘以泡泡糖的【颜色总数】（总数值为：5）。

2-2.2. 把随机的【整型】数值赋值给一个整型变量。

+ 业务说明

1）Math类的random函数会得到一个【大于等于0】且【小于1】的【小数】。

2）本任务通过【随机数】 * 【颜色总数】，使随机数介于0~5的【小数】之间。

3）最后务必对计算结果进行int型的强制类型转化，产生一个0~4的【整型】随机数。

4）int型对【小数】的强制类型转换会将小数部分剔除，产生整型值。

+ 提示

1）随机数产生的是小数类型，需要通过int型强转。

2）【颜色总数】：定义在接口StarService中，可使用静态常量STAR_TYPES。

3）长按Ctrl键，点击StarService查看常量值的定义与注释说明。

2-3. 将产生的【整型】随机数转换为泡泡糖的颜色类型

2-3.1. 利用**Star**类型中**StarType**枚举的**valueOf**方法，将随机【整型】数转换为枚举值。

2-3.2. 将转换后的枚举值赋值给**star**泡泡糖对象的相关属性。

3、验证与测试：

3-1. 定位函数入口所在类：**cn.campsg.practical.bubble.MainClass**。

3-2. 右键选择**Run As -> Java Application**运行该项目工程。

3-3. 观察是否能显示**10 * 10**的随机颜色的泡泡糖矩阵：

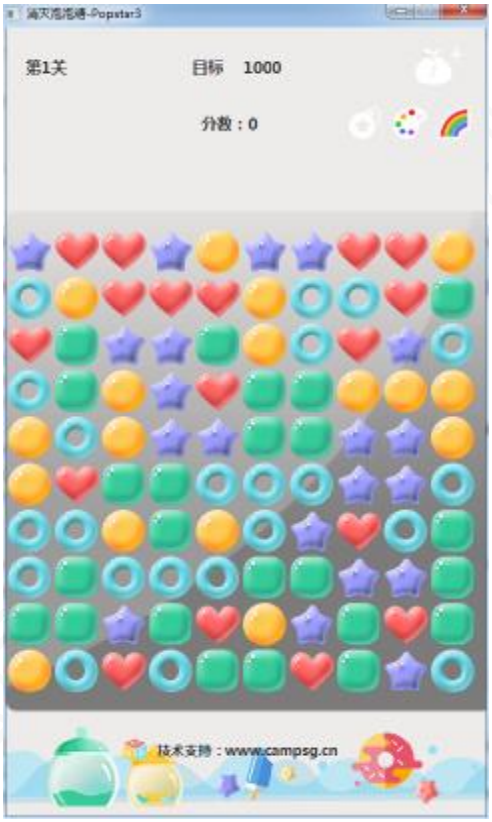


图 **8-1**

4、参考答案：

代码清单：**cn.campsg.practical.bubble.service.StarServiceImpl@createStars**

```
public StarList createStars() {
    StarList stars = new StarList();
    /******************** PRJ-BU2-JAVA-003 Task3 ********************/
    for (int row = 0; row < StarService.MAX_ROW_SIZE; row++) {
        for (int col = 0; col < StarService.MAX_COLUMN_SIZE; col++) {
            Star star = new Star();
```

```
                // 设置泡泡糖在画面上的位置
                star.setPosition(new Position(row, col));
                // 产生随机的泡泡糖
                int typeIndex = (int) (Math.random() * StarService.STAR_TYPES);
                star.setType(StarType.valueOf(typeIndex));
                // 加入列表
                stars.add(star);
            }
        }
        /*************************************************************/
        return stars;
    }
```

9、场景总结

Q1. 项目何时需要定义常量，你会怎么做？

1. 从语法上讲，常量是一个不可修改的变量。

2. 项目中代码逻辑比较复杂，一个固定数值需要被不同业务模块调用，常量具有很强的复用性，可被不同业务函数调用，并保证常量代码修改后，调用函数无需修改。

Q2. for 循环结构中哪个部分用于控制循环次数的？

1. for 循环中第二段控制循环次数，一旦满足条件，则循环结束。

2. 如果第二段的判断始终为 **true**（例如：**1==1**），则将进入死循环。

【说明】:循环初学者可考虑先测试循环的编写是否达到预期效果，随后再为循环加入真实业务。

Q3. 谈谈随机函数 Math.random 的作用与价值。

1. Math类的**random**函数会得到一个【大于等于**0**】且【小于**1**】的【小数】。

2. 我们一般会将 **random** 产生的随机结果 ***** 随机极限值，确保随机上限。

3. 我们一般会将 **random** 产生的随机结果 **+** 随机初始值，确保随机下限。

例如：产生 **3~10** 之间的整数随机值

```
int result = (int) (Math.random * 8) +3
```

4. 随机结果默认为小数，我们可以通过强制类型转换调整随机数的返回结果。

作者：**Roger.Huang**