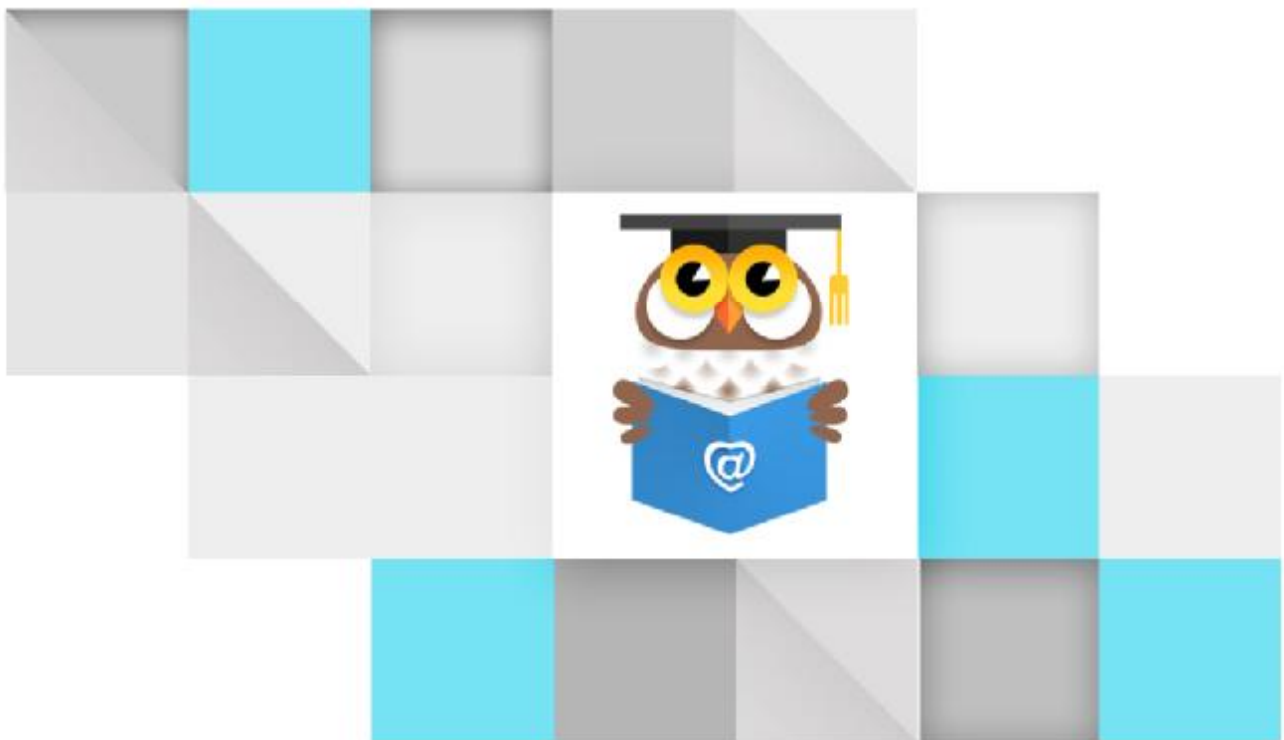


消灭泡泡糖 (Java)

实训指导手册

实训场景 002 - 显示泡泡糖



Campus Solution Group

目 录

一、任务编号：PRJ-BU2-JAVA-002	1
1、实训技能	1
2、涉及知识点	1
3、实现效果	2
4、场景说明	3
5、快速开始	5
6、任务 1 - 创建《消灭泡泡糖》实体类	6
7、任务 2 - 优化《消灭泡泡糖》实体类	9
8、任务 3 - 游戏界面呈现泡泡糖	11
9、场景总结	12

一、任务编号：PRJ-BU2-JAVA-002

1、实训技能

- I Java 面向对象编程技能

2、涉及知识点

- I 类的语法
- I 类的成员变量
- I 类的成员方法
- I public , protected , private
- I 带参构造器、缺省构造器、this
- I 定义枚举类型

3、实现效果

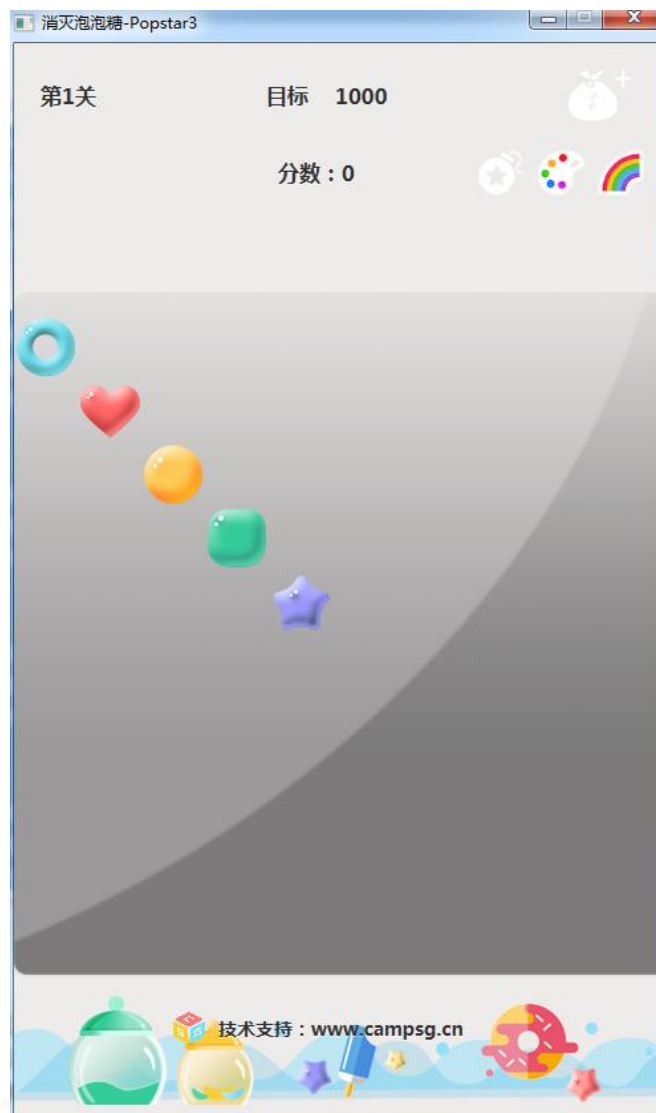


图 3-1

4、场景说明

1、业务说明：

本场景需要实现在游戏界面上呈现5个不同颜色的泡泡糖，泡泡糖的位置由坐标值决定。

2、实现思路：

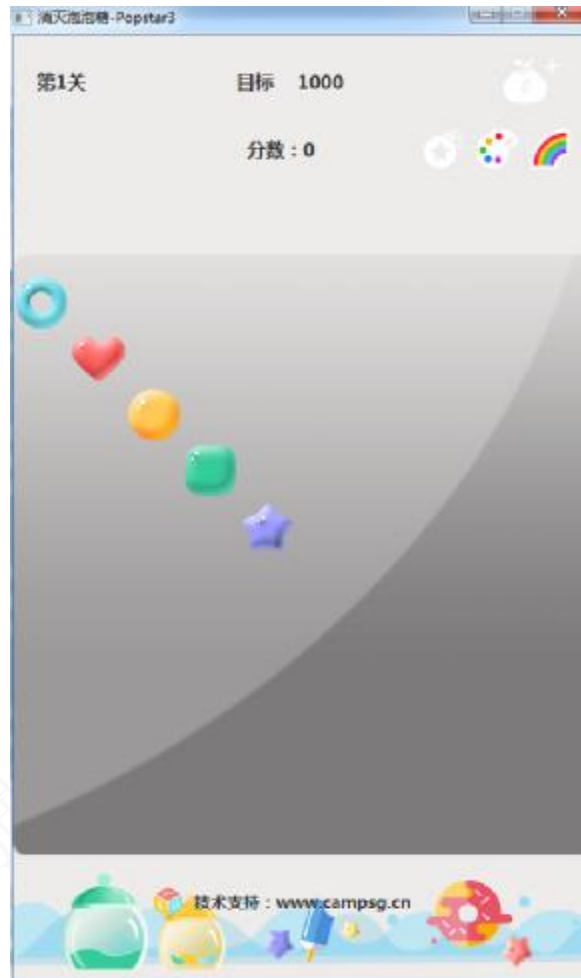


图 4-1

从图4-1可见，我们需要创建5个泡泡糖，并以斜角方式显示在游戏界面上，因此我们需要通过一个函数动态创建5个泡泡糖对象，并通过泡泡糖坐标来控制泡泡的显示位置。完成后，所有被创建的泡泡糖需要传递给游戏界面，游戏界面负责呈现。

在本场景中，我们使用Star类描述泡泡糖，每个在界面上显示的泡泡糖都是Star类的对象。

3、核心组件介绍：

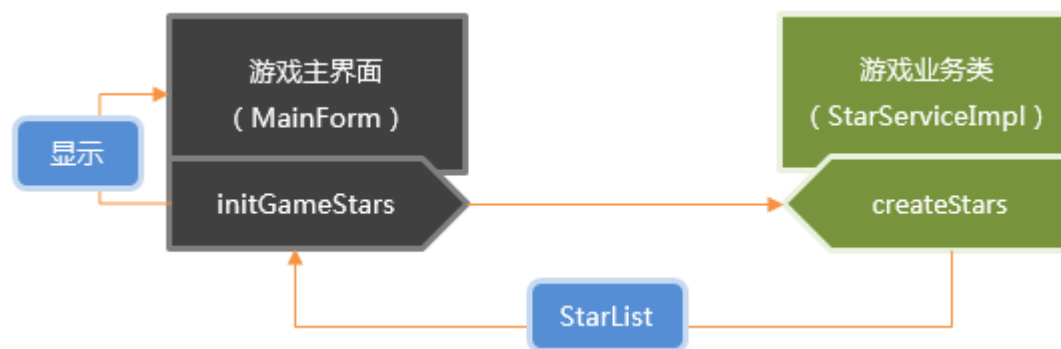


图 4-2

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. StarList - 泡泡糖集合：

用于保存所有待显示的泡泡糖。

保存在StarList中的数据由StarServiceImpl负责创建、删除、编辑、变更，而游戏界面根据StarList中的数据显示不同的效果。

3-4. StarServiceImpl 类中的createStars：

本场景，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，所有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中的泡泡糖对象，并呈现给用户。

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-001 - 泡泡糖体验

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、条件分支语句、运算符、数组）。

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_002

2、进入开发环境：

详见SPOC平台上《PRJ-BU2-JAVA-002 前置任务：进入开发环境》



图 5-1

6、任务 1 - 创建《消灭泡泡糖》实体类

1、任务描述：

1-1. 为游戏《消灭泡泡糖》创建实体类 – Star，以此描述游戏中的一个泡泡糖。

1-2. 每个泡泡糖有两个重要属性，分别为：

1-2.1. 坐标：坐标以行（row）和列（column）描述泡泡糖的位置，坐标规则如下：

1）左上角泡泡糖坐标（0,0），右上角泡泡糖坐标（0,9）

2）左下角泡泡糖坐标（9,0），右下角泡泡糖坐标（9,9），

3）其他泡泡糖以此类推



图 6-1

1-2.2. 类型：类型用来定义泡泡糖的外观，分为以下5种：

黄色圆圈、红色心形、绿色圆角、紫色五角星、蓝色空心圆



图 6-2

2、推荐步骤：

2-1. 定位项目包：cn.campsg.practical.bubble.entity

2-2. 创建实体类Position，Position用于描述泡泡糖在界面上的坐标

2-3. 为Position增加如下属性：

2-3.1. row成员变量 – 整数类型

2-3.2. column成员变量 – 整数类型

2-3.3. 为成员变量添加get与set访问器

+ 提示：

- 1) 成员变量的作用域应该设置为：private
- 2) 访问器的创建可使用Eclipse中的自动化工具实现：

单击右键，选择Source à Generate Getters and Setters

2-4. 创建实体类 Star，用于描述界面上的一个泡泡糖

2-5. 为泡泡糖Star类创建“类型”属性，属性类型为枚举型：

2-5.1. Star类中，创建枚举类型：StarType

2-5.2. 为枚举创建属性：value成员变量 – 整数类型，用于保存枚举值

2-5.3. 为枚举增加构造函数，构造函数需对value赋值

+ 提示：

- 1) 枚举的构造函数必须是私有的（private）
- 2) 枚举构造函数需要对内部成员变量赋值，保证例如：BLUE(0)的形式合法

2-5.4. StarType枚举具有以下枚举值：

- 1) 0 - BLUE
- 2) 1 - GREEN
- 3) 2 - YELLOW
- 4) 3 - RED
- 5) 4 - PURPLE

2-5.5. 实现【枚举值】向【数值】转换（BLUE -> 0）。

- 1) 为StarType枚举添加公共函数value，返回类型为整型。
- 2) 在value函数中，返回枚举值对应的整型数值。

+ 提示

- 1) value方法是枚举类型的功能函数，因此无需设置为静态。
- 2) value用于返回特定枚举值对应的数值，因此无需入参。
- 3) value成员变量用于保存枚举数值，在value函数中要善加利用。

2-6. 为Star实体类增加如下属性：

2-1.1. position成员变量 – Position类型。

2-1.2. type成员变量 – StarType类型。

2-1.3. 为成员变量添加get与set访问器。

+ 提示：

- 1) Position类型就是步骤2-3创建的类型，Position可以直观地描述泡泡糖的界面坐标。
- 2) StarType类型就是步骤2-5创建枚举，界面根据StarType值显示泡泡糖的外观。

2-7. 为Position创建两个构造函数：

2-7.1. 创建两参构造函数，第一参row，第二参数column。

2-7.2. 通过两参构造函数对Position的成员变量赋值。

2-7.3. 创建0参构造函数，该构造函数无任何操作。

2-8. 为Star创建两个构造函数：

2-8.1. 创建两参构造函数，第一参Position，第二参数StarType。

2-8.2. 通过两参构造函数对Star的成员变量赋值。

2-8.3. 创建0参构造函数，要求实例化Position变量，type默认值为BLUE。

3、验证与测试：

3-1. 定位：cn.campsg.practical.bubble.service.StarServiceImpl

3-2. 找到：public StarList createStars()函数

3-3. 编写以下测试语句：

3-3.1. 创建Star类的对象star1。

- 1) 利用setPosition函数为Star赋值 (0,0) 位置坐标，使用Position构造函数赋值。
- 2) 利用setType函数为Star复制泡泡糖类型：BLUE。

3-3.2. 创建Star类的对象star2。

- 1) star2的创建必须利用Star构造函数完成，不得使用任何set方法。
- 2) star2的数据为：(1,1) 位置坐标，类型：GREEN。

3-4. 打印测试结果：

3-4.1. 利用Java打印语句分别显示star1和star2的属性值

3-4.2. 运行项目，选择cn.campsg.practical.bubble.MainClass类

3-4.3. 控制台输出结果与下图一致：

```
(0),(0) - BLUE
(1),(1) - GREEN
```

图 6-1

7、任务 2 – 优化《消灭泡泡糖》实体类

1、任务描述：

任务1完成后，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，当前任务主要是为枚举添加转换函数，实现数值向枚举的转化 (0->BLUE)。

2、实现思路：

2-1. 为StarType枚举添加valueOf函数，实现数值向枚举的转换 (0->BLUE)。

2-1.1. 创建**公共静态函数**valueOf，参数为待转换的数值，返回类型为StarType

2-1.2. 建议利用switch case依次判断valueOf的参数是否按以下规则返回：

- 1) 0 - 返回BLUE
- 2) 1 - 返回GREEN
- 3) 2 - 返回YELLOW
- 4) 3 - 返回RED
- 5) 4 - 返回PURPLE

+ 提示

- 1) 静态函数可通过枚举类型直接调用，关键字static。
- 2) 由于是数值向枚举转换，因此入参应设置为整型。
- 3) 如使用switch case，那么编写时，每个case下都不要遗漏break代码。

3、验证与测试：

3-1. 定位：cn.campsg.practical.bubble.service.StarServiceImpl

3-2. 找到：public StarList createStars()函数

3-3. 编写以下测试语句：

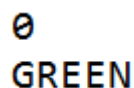
3-3.1. 打印StarType.BLUE对应的数值。

3-3.2. 打印“1”对应的StarType枚举值。

3-4. 打印测试结果：

3-4.1. 运行项目，选择cn.campsg.practical.bubble.MainClass类

3-4.2. 控制台输出结果与下图一致：



```
0
GREEN
```

图 7-1

3-4.3. 您可以考虑试试超过范围的数据显示，例如：StarType.valueOf(10)。

8、任务 3 – 游戏界面呈现泡泡糖

1、任务描述：

任务1和任务2均利用控制台显示了泡泡糖数据，当前任务将会把Star类数据显示在游戏界面上。

2、实现思路：

2-1. 创建泡泡糖存储“容器” - StarList。

2-2. 将所有实例化的泡泡糖保存于StarList中。

2-3. 将StarList返回游戏界面（游戏界面会将StarList中保存的泡泡糖数据呈现在界面上）。

+ 提示

1) 将Star保存于StarList中，可使用StarList中的add方法。

例如：

```
StarList sList = new StarList();  
  
sList.add(new Star());
```

3、验证与测试：

3-1. 定位：cn.campsg.practical.bubble.service. StarServiceImpl

3-2. 找到：public StarList createStars()函数

3-3. 编写以下测试语句：

3-4.4. 创建属性为：position: (0,0) ， type：BLUE的泡泡糖。

3-4.5. 创建属性为：position: (1,1) ， type：RED的泡泡糖。

3-4.6. 创建属性为：position: (2,2) ， type：YELLOW的泡泡糖。

3-4.7. 创建属性为：position: (3,3) ， type：GREEN的泡泡糖。

3-4.8. 创建属性为：position: (4,4) ， type：PUPPLE的泡泡糖。

3-4.9. 将以上泡泡糖对象保存于StarList中。

3-5. 打印测试结果：

3-5.1. 运行项目，选择cn.campsg.practical.bubble.MainClass类

3-5.2. 界面显示结果与下图一致：

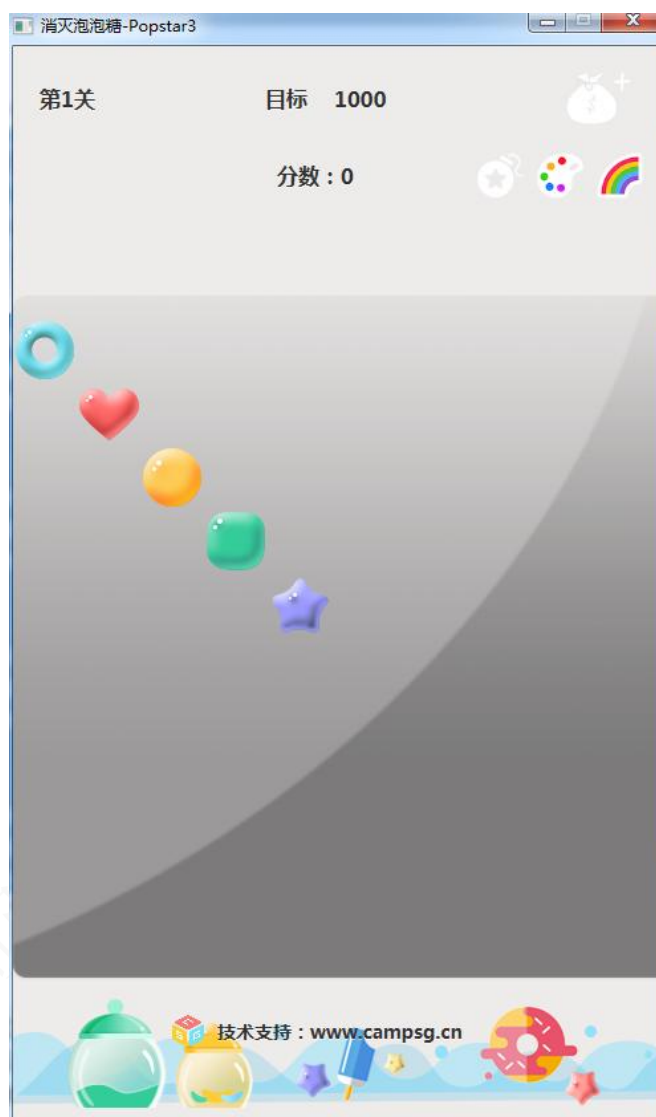


图 8-1

9、场景总结

Q1. Java中实体类的作用是什么？属性和字段有何区别？

1. 程序因为现实生活中的业务而诞生，业务由无数个物体所组成，业务信息化时，Java 中的实体用来描述现实生活中的物体，在为对象增加业务后程序逻辑就产生了，所以实体是程序的核心单元。

2. 字段一般是私有的变量，例如：`private String name`。

3. 属性一般是公有的函数，例如：`public void setName(String name)`

4. 字段是现实生活中物体的属性（例如：名字、年龄等），属性则可为字段增加相关业务。

5. 单独调用字段往往没有意义（`private`），所以Java实体一般只允许调用属性（`public`）。

Q2. 构造函数的作用是什么？项目中你会如何定义构造函数？

1. 构造函数用于初始化类中的重要成员变量。

2. 构造函数可以多次重载，默认类具有零参构造函数。

3. 构造函数如为私有作用域，表示该类不能实例化（例如：`Utils`类）。

例如：`private MovedStar() { }`

4. 构造函数初始化的变量往往非常重要，并不是所有成员变量都需要在构造函数中初始化。

例如：人类创建时，需要首先初始化脑袋成员对象，理由是没有脑袋，人存在没有意义。

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。

Q4. Java 中的枚举值有何作用？

1. 枚举值可以提升代码的可读性，枚举值可以使无意义的数值或字符变得更易理解。

【说明】 由于需求变更是软件开发的常态，因此可读性强的代码将降低软件的后期维护成本。

Q5. Java 中的枚举值如何实现数值与枚举常量间的转换？

1. 利用枚举结构中的 valueOf 函数。

作者：Frank.Chen