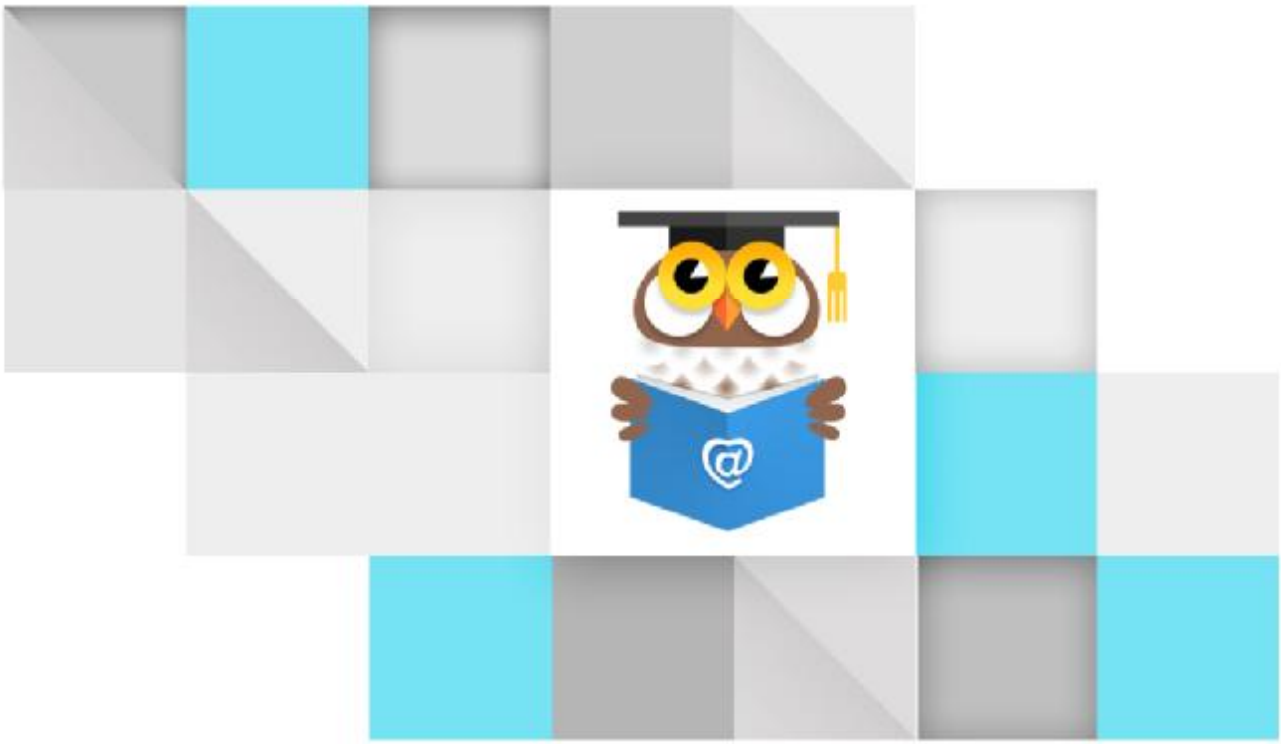
消灭泡泡糖（Java）

实训指导手册

实训场景 006 – 体验接口解耦特性



Campus Solution Group

目 录

一、任务编号：PRJ-BU2-JAVA-006

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
- 构造器调用
- 接口的特性
- 接口的语法
- 接口的实现
- 适配器类
- JavaFX-Label 视图(*)
- JavaFX-Pane 容器(*)
- JavaFX 视图属性(*)
- 反射-动态加载类(*)
- 反射-动态创建类实例(*)

* 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：

1-1. 本场景利用接口技术的特性，结合《消灭泡泡糖》游戏业务体验企业级软件开发模式与流程，感受真实软件系统开发功能模块的流程与方法。

1-2. 由**PRJ-BU2-JAVA-002**场景可知，**createStars**方法用于创建界面显示的**10 * 10**泡泡糖矩阵，当前场景基于该业务方法还原企业级开发流程如图**4-1**。

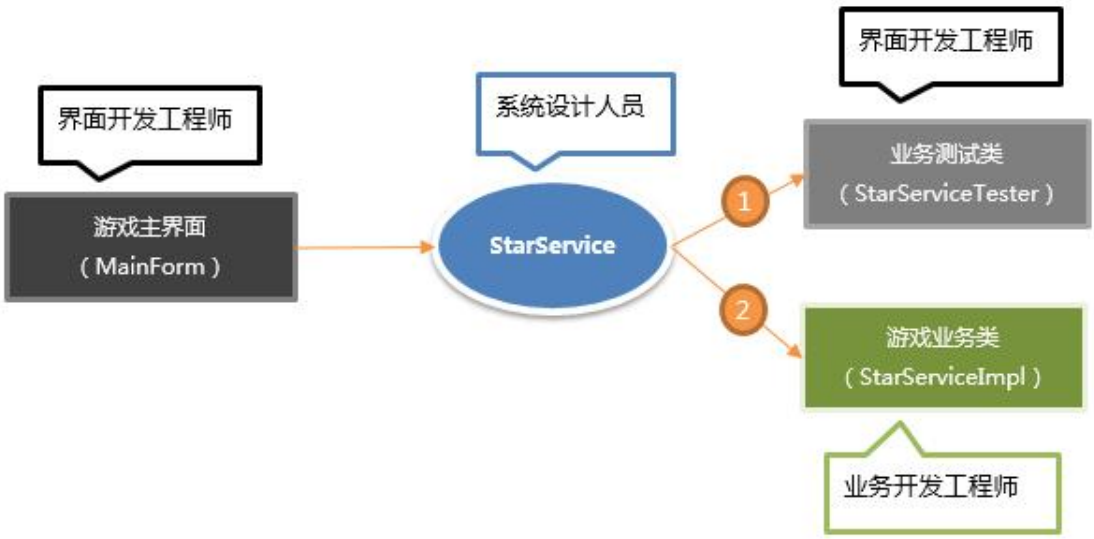


图 **4-1**

1-3. 如图**4-1**创建界面显示的**10 * 10**泡泡糖矩阵，企业级开发流程如下：

1-3.1. 界面开发人员负责游戏界面开发，不负责游戏业务计算。

1-3.2. 业务开发人员负责开发业务类中的**createStars**方法，不考虑界面显示。

1-3.3. 企业级项目中界面与业务层的开发往往是同步进行的，因此界面开发人员缺乏真实数据验证界面代码的正确性。

1-3.4. 此时，界面开发人员会编写一个存放假数据的业务测试类，对界面的逻辑代码进行验证。

1-3.5. 测试正确后，待业务开发人员完成业务类的实现后，再切换到真实环境下进行测试。

1-4. 为确保**1-3**流程正常执行，系统设计人员就需要设计出一个满足业务需求的接口，分别交由界面与业务开发人员实现。

1-5. 由于系统设计人员只考虑游戏业务与功能函数的对应关系，不负责最终的代码实现，因此接口中所有的函数都是抽象的（没有函数体）。

1-6. 本场景将基于**1-3**和**1-4**的企业级开发流程，体验接口技术的【解耦特性】。

2、 实现思路：

2-1. 首先，扮演界面开发人员，创建【业务测试类】并新建多个固定位置的泡泡糖对象。

2-2. 其次，通过界面类**MainForm**调用【业务测试类】对象，显示测试矩阵。

2-3. 测试正确后，把【业务测试类】切换为【真是业务类】，测试泡泡糖的显示效果。

3、 核心组件介绍：

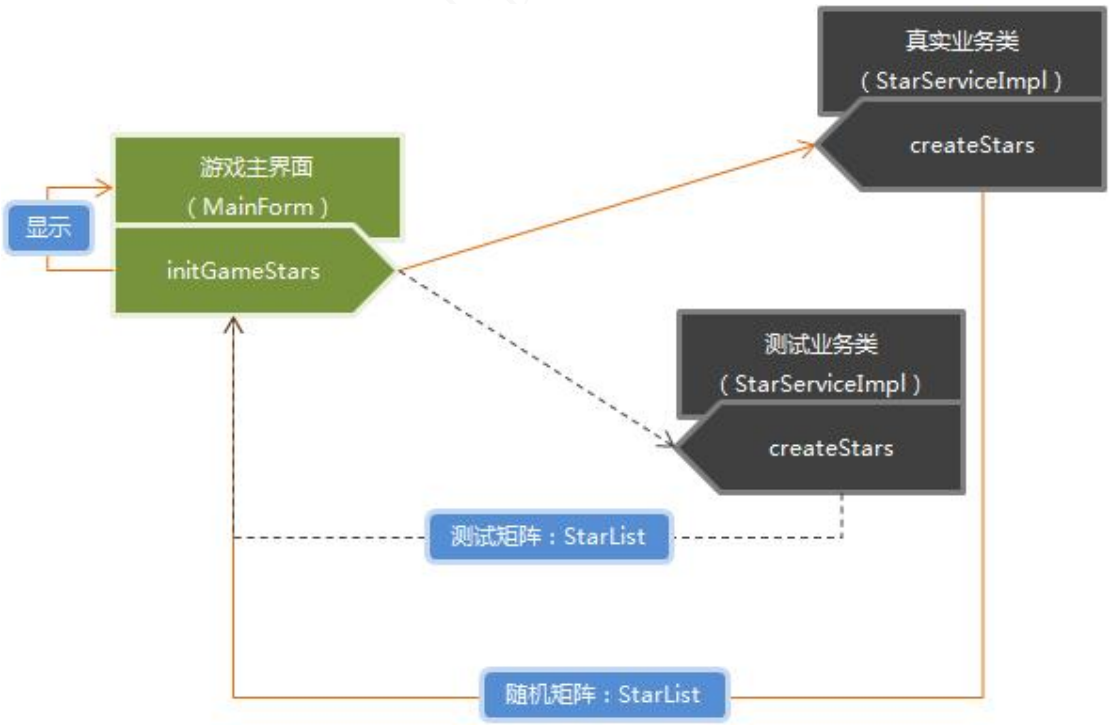


图 **4-1**

3-1. MainForm - 游戏界面类：

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【真实业务类】：

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. StarServiceTester – 游戏【业务测试类】：

该类由界面开发人员创建，通过创建多个固定位置的泡泡糖测试游戏界面。

3-4. MainForm - initGameStars :

该方法用于把泡泡糖对象以可视化的方式呈现到界面上。

3-4.1. 该方法负责调用业务类的**createStars**函数（不考虑是测试类还是真实类）。

3-4.2. 根据**createStars**的返回，将泡泡糖对象转换为显示控件，呈现在游戏界面上。

3-5. StarList –泡泡糖集合：

用于保存生成的的泡泡糖列表。

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** – 封装待移动泡泡糖

5-2. 必备知识与技能：

5-2.1. Java面向对象编程技能（类、方法调用、接口、重写方法）。

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_006**

2、 进入开发环境：

详见**SPOC**平台上《**PRJ-BU2-JAVA-006** 前置任务：进入开发环境》

图 **5-1**

6、任务 1 – 创建服务测试类

1、 任务描述：

1-1. 扮演界面开发人员，创建业务测试类**StarServiceTester**，实现**StarService**接口。

1-2. 实现**createStars**方法，共创建**5**个不同颜色，不同坐标位置的泡泡糖，组成测试用的泡泡糖矩阵返回游戏界面。

2、 推荐步骤：

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

2-2. 创建业务测试类**StarServiceTester**，实现**StarService**接口。

2-3. 实现接口的所有方法

+ 提示

1） 快捷方式：把光标放到红色波浪线的类名**StarServiceTester**上，在弹出的提示窗中选择“**Add unimplemented methods**”就会自动填充待实现的方法。

2） 自动生成函数顶部的**@Override**关键字含义：该关键字用来告知编译器，函数重写了父类或接口定义的函数，如果您试图修改函数名或函数结构，编译器会给出错误提示。

2-4. 实现**createStars**方法

2-4.1. 创建一个用于保存泡泡糖的**StarList**对象

2-4.2. 创建**5**个泡泡糖对象，并把它们添加到**StarList**中

1）坐标分别为：（**0,0**）、（**0,1**）、（**1,0**）、（**1,1**）、（**0,2**）

2）类型分别为：**BULE**、**GREEN**、**PURPLE**、**YELLOW**、**RED**

2-4.3. 返回**StarList**对象。

3、 验证与测试：

3-1. 在当前类中创建一个**main**函数作为测试程序的入口。

3-2. 创建一个**StarServiceTester**对象

3-3. 调用**createStars**方法获得测试用泡泡糖矩阵

3-4. 向控制台输出该泡泡糖矩阵的信息。

3-5. 观察控制台是否输出了相应的信息：

```
[(0,0-BLUE), (0,1-GREEN), (1,0-PURPLE), (1,1-YELLOW), (0,2-RED)]
```

图 **6-1**

7、任务 2 – 实现界面泡泡糖显示

1、 任务描述：

1-1. 扮演界面开发人员，完成游戏界面【呈现泡泡糖】的功能开发。

1-1.1. 本任务在开发游戏界面时，将涉及到部分**JavaFX**的知识，任务说明将会提供相关实现代码并配备有相关说明，使用者以体验的方式了解**JavaFX**技术即可。

1-2. 访问任务**1**创建的【测试业务类】对象，把测试类中创建的泡泡糖显示在界面上。

1-2.1. 当前步骤需要将【测试业务类】获取的多个泡泡糖对象**Star**，转换成界面显示控件**Label**，并存储在界面容器中，整个转换过程如图**7-1**所示：

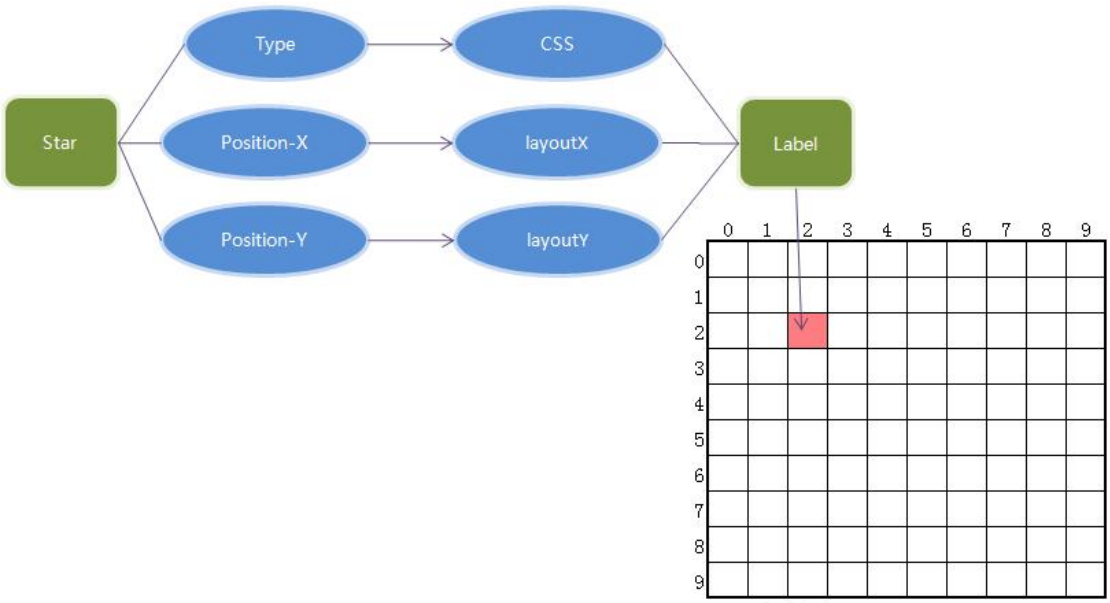


图 **7-1**

1-2.2. 从图**7-1**中可见，泡泡糖对象**Star**与界面显示控件**Label**属性对应关系如下：

属性名	泡泡糖对象属性	游戏界面控件属性
泡泡糖类型	**type = StarType.RED**	**styleClass = red_star**
位置坐标（**x**）	**position.x = 2**	**layoutX = 2 *** 泡泡糖宽度
位置坐标（**y**）	**position.y = 2**	**layoutY = 2 *** 泡泡糖高度

1-2.3. Star à Label后，我们还需要将**Label**保存入游戏界面的泡泡糖容器中，容器根据控件的**styleClass**、**layoutX**、**layoutY**属性显示不同颜色、不同位置的泡泡糖。

1-3. **1-2**测试正确后，界面开发人员可基本确定游戏界面代码已符合要求，随后把【业务测试类】对象修改为【真实业务类】对象，确保界面显示**10*10**的随机泡泡糖矩阵。

2、 推荐步骤：

2-1. 定位到类：**cn.campsg.practical.bubble.MainForm**

2-1.1. 定位到方法**initGameStars**的注释处

2-2. 以面向接口的方式创建一个**StarServiceTester**对象。

+ 提示

1）面向接口创建对象：接口 对象 **= new** 实现类（）；

例如：**StarService service = new StarServiceTester**（）；

2-3. 使用【测试业务类】对象创建泡泡糖矩阵，赋值给全局变量**mCurretStars**。

+ 业务说明

1）全局变量**mCurretStars**用于保存【服务类】创建的【泡泡糖集合】，游戏界面将根据该集合显示矩阵。

2-4. 使用**for**循环遍历整个**mCurretStars**集合，将所有【泡泡糖】对象转换为显示控件。

2-4.1. 依次获取集合中每个【泡泡糖】对象 **- Star**。

2-4.2. 创建一个泡泡糖显示控件 **- Label**对象。

2-4.3. 并将**Label**控件的宽度和高度均设置为**48**（每个泡泡糖的高度和宽度是固定的）。

+ 提示

1） 本处使用了**JavaFX**的知识，请按以下代码实现任务需求：

```
Label starFrame = new Label(); //创建界面泡泡糖显示控件Label
starFrame.setPrefWidth(48);    //设置显示控件的宽度
starFrame.setPrefHeight(48);   //设置显示控件的高度
```

2-4.4. 分别获取泡泡糖对象**Star**的行值与列值。

2-4.5. 设置**Label**对象的**id**值，**id**的规则为：**s**行号列号（例如：**s00**，**s10**，**s02**）

+ 提示

1） 本处使用了**JavaFX**的知识，请按以下代码实现任务需求：

```
starFrame.setId("s" + row + col);
```

2） **ID**是显示控件**Label**的唯一标识符，**JavaFX**中每个控件都应该具有一个唯一的标识符**ID**（就如同中国公民的身份证）。

2-4.6. 将泡泡糖对象的行值、列值数据保存在泡泡糖显示控件中。

+ 提示

1） 本处使用了**JavaFX**的知识，请按以下代码实现任务需求：

```
starFrame.setUserData(row + ";" + col);
```

2） 以上代码将泡泡糖的行列位置保存起来，用于识别泡泡糖在界面中的位置。

2-4.7. 根据泡泡糖对象的**position**属性，设置泡泡糖显示的像素坐标位置

+ 提示

1） 本处使用了**JavaFX**的知识，请按以下代码实现任务需求：

```
starFrame.setLayoutX(col * 48); //设置泡泡糖显示控件在界面的显示位置（x）
starFrame.setLayoutY(row * 48); //设置泡泡糖显示控件在界面的显示位置（y）
```

2） 以上代码用来设置泡泡糖显示的坐标位置，为保证显示泡泡糖不发生重叠，因此每个泡泡糖显示控件的**x**轴和**y**轴数据都应该相互间隔**48**（**48**正是泡泡糖的宽度与高度）。

2-4.8. 根据泡泡糖对象的**type**属性，设置界面泡泡糖的外观。

+ 提示

1） 本处使用了**JavaFX**的知识，请按以下代码实现任务需求：

```
switch (star.getType().value()) {
    case 0:
        starFrame.getStyleClass().add("blue_star");
        break;
    case 1:
        starFrame.getStyleClass().add("green_star");
        break;
    case 2:
        starFrame.getStyleClass().add("yellow_star");
        break;
    case 3:
        starFrame.getStyleClass().add("red_star");
        break;
    case 4:
        starFrame.getStyleClass().add("purple_star");
        break;
    }
```

2） 以上代码通过判断泡泡糖对象的**type**枚举值，依次计算界面显示控件对应的外观样式，界面通过外观样式（**styleClass**）设置**Label**的皮肤，以此达到显示不同颜色泡泡糖的效果。

3） 感兴趣的用户可以通过文件“**src/res/css/skin2.css**”查看泡泡糖的样式代码。

2-4.9. 把转换完毕的泡泡糖显示控件，添加到游戏界面上的泡泡糖容器中。

+ 提示

1） 本处使用了**JavaFX**的知识，请按以下代码实现任务需求：

```
mStarForm.getChildren().add(starFrame);
```

2） **mStarForm**：游戏界面。

3） **getChildren()**：游戏界面上的泡泡糖容器。

4） **add**方法：把转换完毕的泡泡糖显示控件加入容器中。

3、 验证与测试：

3-1. 定位到程序入口类：**cn.campsg.practical.bubble.MainClass**

3-2. 运行该项目，观察界面是否能显示测试用的泡泡糖：

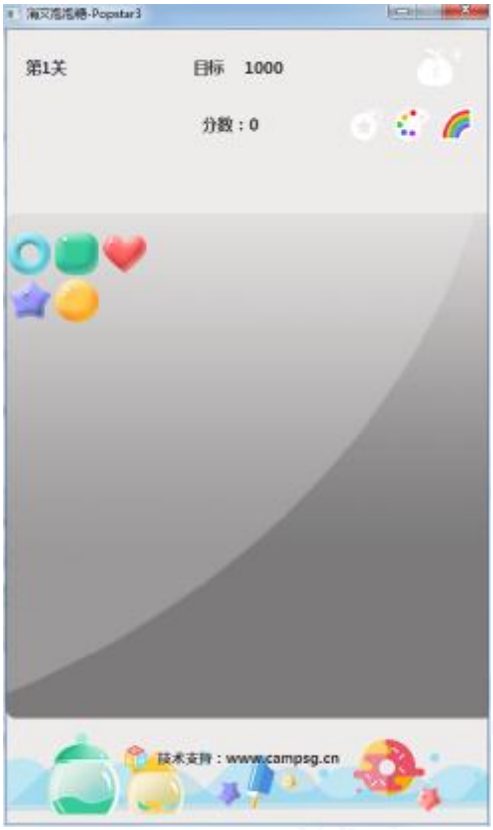


图 **7-2**

3-3. 定位到类：**cn.campsg.practical.bubble.MainForm**

3-4. 现假设业务开发人员已经完成了【真实业务类】的开发，为保证代码逻辑的正确性，现需要将实例化的【业务测试类】**StarServiceTester**对象修改为【真实业务类】**StarServiceImpl**的对象。

+ 提示

1）面向接口创建对象：接口 对象 **= new** 实现类（ ）；

例如：**StarService service = new StarServiceImpl**（ ）；

3-5. 再次运行该项目观察是否能看到随机泡泡糖矩阵：

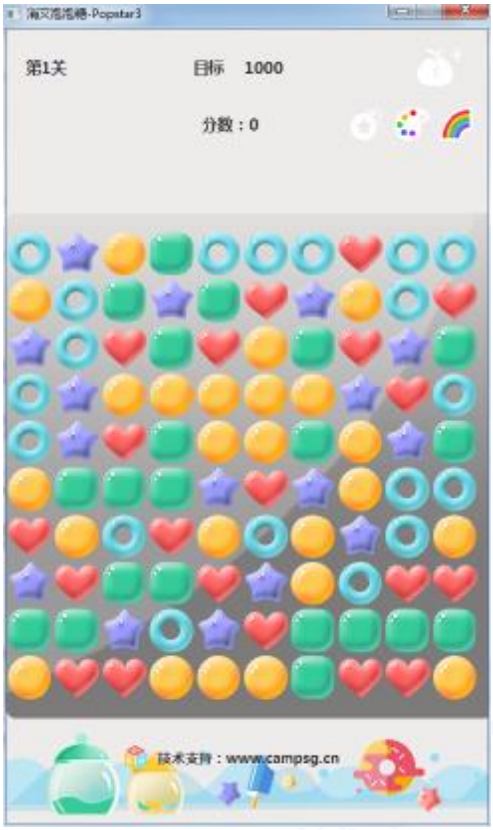


图 **7-3**

8、任务 3 – 通过接口动态切换实现类

1、 任务描述：

1-1. 任务**2**中，界面开发人员实现了：从【业务类】获取泡泡糖对象集合，并将其转换为显示控件**Label**呈现在游戏界面上的业务。

1-2. 同时，任务**2**还实现了【业务测试服务类】与【真实业务服务类】代码切换的功能，基本体验了企业级软件开发流程中某个功能函数的标准开发流程。

1-3. 本任务在任务**2**的基础上，实现更高级的“动态业务类”切换功能，保证在不修改代码的情况下，能够实现【测试】与【真实】业务类之间的相互切换。

2、 推荐步骤：

2-1. 定位到该类中的**initGameStars**方法处

2-2. 以面向接口的方式创建【业务】对象的代码，修改为调用私有方法**getStarService**来获得业务类的方式。

+ 业务说明

1） **getStarService**中逻辑代码在企业级开发过程中，一般由系统设计人员开发。

2） **getStarService**方法涉及了部分**Java**反射机制和文件流读取的知识，有兴趣的用户可以查看该方法的源代码了解。

3） 不修改代码的情况下，实现 “业务类动态切换”在复杂的企业级开发逻辑中，可全面提供应用系统的维护性和可扩展性。

3、 验证与测试：

3-1. 定位到程序入口类：**cn.campsg.practical.bubble.MainClass**

3-2. 运行该项目，观察界面是否能显示测试用的泡泡糖：

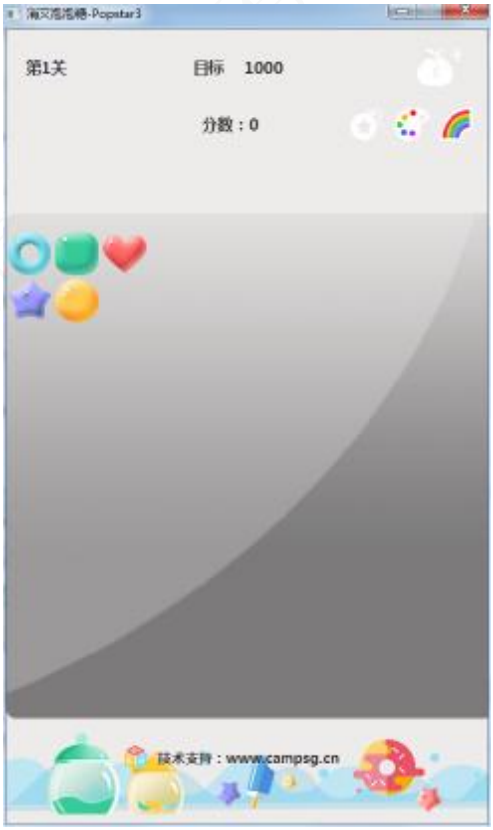


图 8-1

3-3. 定位文件：**src/ bean.conf**文件。

3-4. 打开文件后，按如下规则修改文件内容：

```
service=cn.campsg.practical.bubble.service.StarServiceTester
```

修改为

```
service=cn.campsg.practical.bubble.service.StarServiceImpl
```

3-5. 再次运行该项目，观察界面是否能显示真实业务的泡泡糖矩阵：

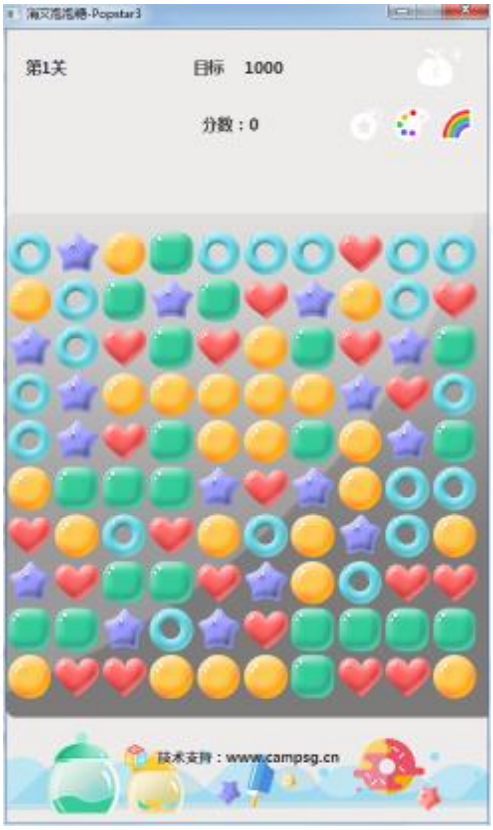


图 **8-2**

9、场景总结

Q1. 谈谈接口的作用（不要讲语法），并说明在您所开发的项目中如何定义与设计接口？

1. 接口有两大作用：定标准、解耦合（切记）。

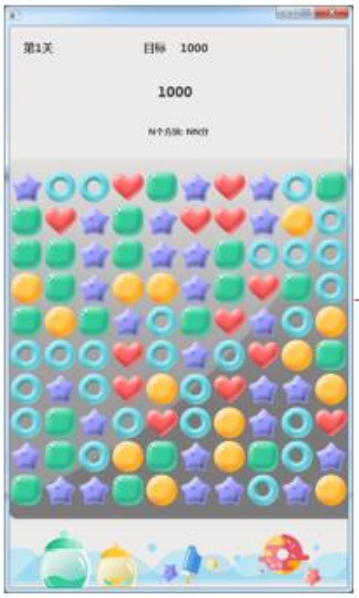
1-1. 定标准，业务调用方允许设定业务规范，只有满足业务标准的组件才可以与调用方交互

（多用于系统架构设计）。

1-2. 解耦合，原本只能一人开发的系统，降耦合后可以由多人同时开发，最后组装。

例如：当前场景通过接口**StarService**将界面和业务彻底隔离，界面层只负责显示**"**泡泡糖**"**，**"**泡泡糖**"**的消除、移动、计算全部由业务层负责。

由于界面层需要调用获取业务层计算结果后才可以显示“泡泡糖”，因此界面层扮演接口调用方，业务层扮演接口实现方，具体见下图：



MainForm（视图层）

调用

StarService（接口）

实现

public StarList createStars();

StarServiceTester (业务测试类)

实现

public StarList createStars();

StarServiceImpl (真实业务类)

2. StarService使原本一人自顶向下的开发模式，变成了多人并行开发，**1**人负责界面，**1**人负责业务。

扩展：层与层之间的数据交互，一般需要通过对象来实现，当前系统的数据交互对象是**StarList**，**1**个**StarList**中可以存放多个**Star**。

作者：**Roger.Huang**