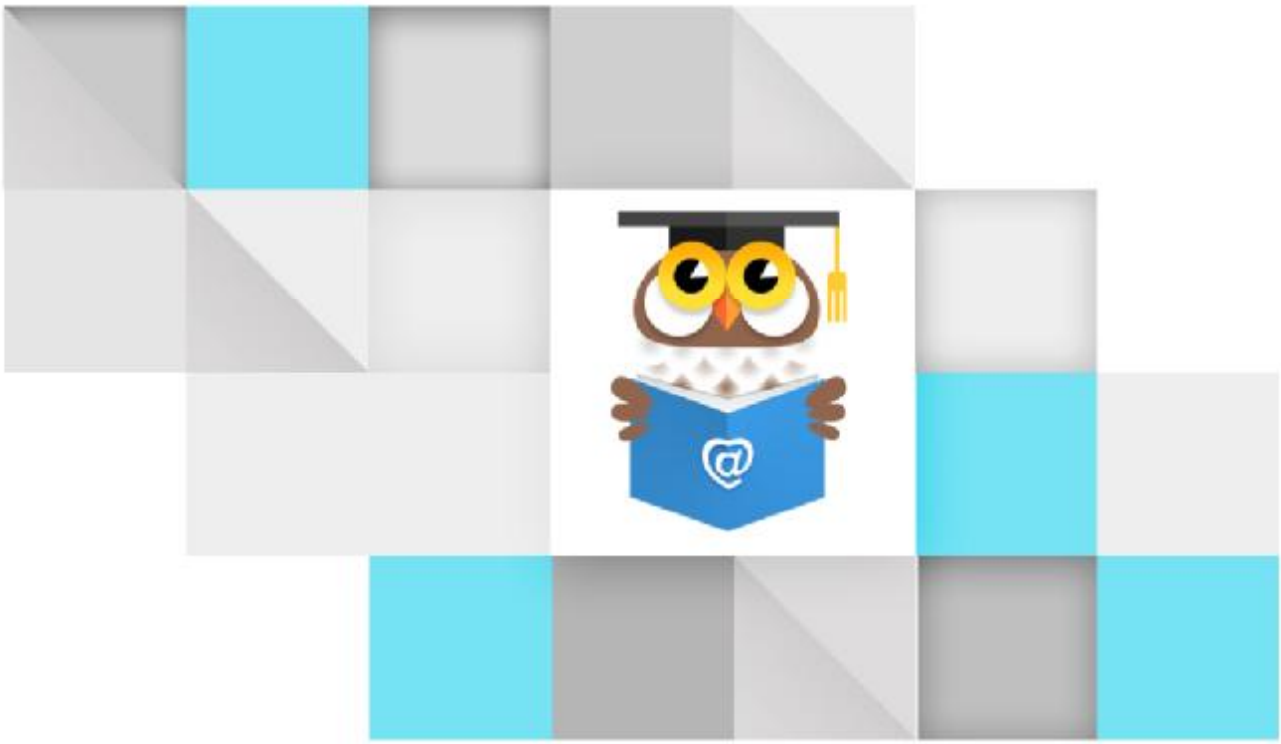
消灭泡泡糖（Java）

实训指导手册

实训场景 008 – 在游戏中合理利用异常



Campus Solution Group

目 录

一、任务编号：PRJ-BU2-JAVA-008

1、实训技能

- Java 面向对象编程技能

2、涉及知识点

- 异常的捕捉语法
- 异常对象
- 抛出异常
- 异常转移
- 创建异常类
- JavaFX-模式对话框(*)

* 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：

本场景将利用**Java**异常技术（**Exception**）实现《消灭泡泡糖》游戏中，类与类之间的错误信息传递，本场景共设计有两个自定义异常，分别为：

1-1. 游戏初始化异常 – **ServiceInitException**

1-2. 无可消除的泡泡糖异常 – **NoClearedStarsException**

我们将分别通过定义异常、申明异常、抛出异常、捕捉异常、处理异常等技术手段，为《消灭泡泡糖》游戏建立企业级排错机制，保证游戏程序运行的健壮性。

2、 实现思路：

2-1. 通过捕捉**ServiceInitException**异常，防止游戏程序加载【业务类】时发生错误。

2-2. 创建**NoClearedStarsException**异常，用于通知游戏界面无可消除的泡泡糖。

3、 核心组件介绍：

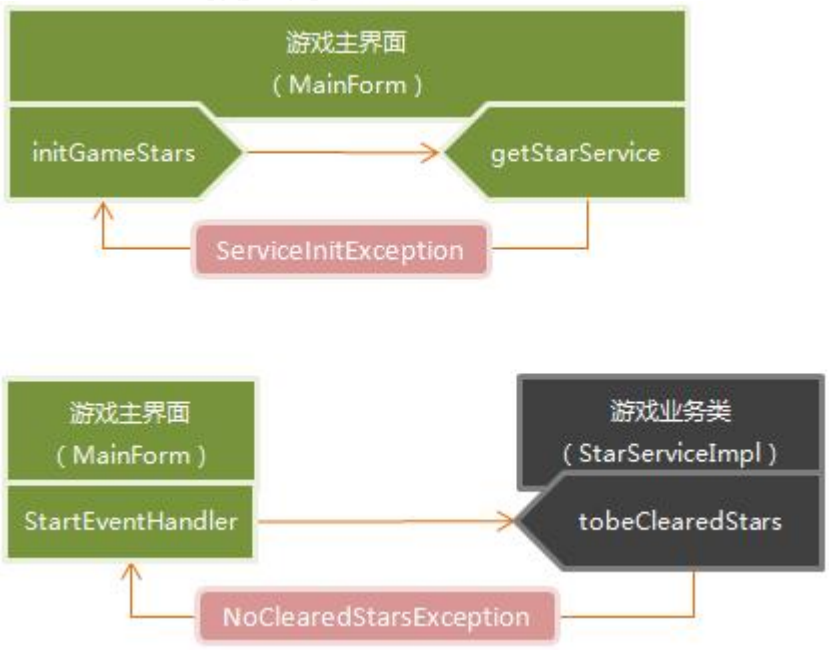


图 **4-1**

3-1. ServiceInitException – 游戏初始化异常：

3-1.1. 抛出者：**MainForm**类的**getStarService**方法

3-1.2. 抛出时机：**bean.conf**文件不存在或文件中配置的业务类类名不正确。

3-1.3. 捕捉者：**MainForm**类的**initGameStars**方法

3-1.4. 处理方式：弹提示对话框通知用户，并结束游戏程序。

3-2. NoClearedStarsException – 无可消除泡泡糖异常：

3-2.1. 抛出者：**StarServiceImpl**类的**tobeClearedStars**方法

3-2.2. 抛出时机：【被点击】泡泡糖四周无【同色】【待消除】泡泡糖。

3-2.3. 捕捉者：【事件处理类】**StartEventHandler**中的**handle**事件处理方法。

3-2.4. 处理方式：不执行消除操作及消除泡泡糖的后续业务流程。

3-3. 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。

3-4. 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。

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-007** – 体验接口隔离性

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、异常类创建、捕获异常）。

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_008**

2、 进入开发环境：

详见**SPOC**平台上《**PRJ-BU2-JAVA-008** 前置任务：进入开发环境》

图 **5-1**

6、任务 1 – 优化服务类加载

1、 任务描述：

1-1. 本任务主要通过**ServiceInitException**为**getService**函数设计错误处理机制，保证“动态加载业务类”功能函数在失去核心配置文件的情况下，不出现致命错误。

1-2. 开发流程如下：

1-2.1. 通过场景**PRJ-BU2-JAVA-006**可知，我们通过**getService**方法实现了“动态加载业务类”的功能，业务类的切换无需编码，只需修改**src/bean.conf**配置文件中的数据即可。

1-2.2. 当游戏发布后，如果游戏用户误删**bean.conf**配置文件，可能导致程序发生致命

性错误，为了规避该错误，我们必须为“动态加载业务类”的功能设计错误处理机制。

1-2.3. **getService**方法发现**bean.conf**文件不存在或文件中配置的业务类类名不正确，会抛出**ServiceInitException**，通知调用方处理该异常。

1-2.4. 游戏界面负责调用**getService**，因此有义务捕捉**ServiceInitException**异常，如果该异常被抛出，执行的异常处理手段是：弹提示对话框通知用户，并结束游戏程序。

2、 推荐步骤：

2-1. 打开**PRJ_BU2_JAVA_007**工程包。

2-2. 定位工程包中的**MainForm**类的**initGameStars**方法。

2-3. 请将该方法内的所有代码复制到：**PRJ_BU2_JAVA_008**工程包的同名函数中。

+ 业务说明

1）复制完毕后，**getStarService**函数下出现了错误提示，原因是该方法将会抛出一个**ServiceInitException**异常，由于未捕捉该异常，因此编译器给出了错误提示。

2）当前任务中，异常处理中的各个角色如下：

2-1. 抛出者：**MainForm**类的**getStarService**方法。

2-2. 抛出时机：**bean.conf**文件不存在或文件中配置的业务类类名不正确。

2-3. 捕捉者：**MainForm**类的**initGameStars**方法。

2-4. 利用**try...catch**语法块，捕获**getStarService**方法抛出的异常**ServiceInitException**

+ 提示

1） 快捷方式 :把光标放到红色波浪线的语句上 ,选择“**Surround with try/catch**” ,**Eclipse**会自动出现异常捕捉代码。

2） 请为**StarService**对象设置初始值，初始值为**null**。

2-5. 在**catch**块中编写异常处理逻辑：

2-1.1. 编写弹窗代码提示用户，因为缺少核心文件，因此程序无法继续运行。

2-1.2. 终止应用程序的运行。

+ 提示

1）本处使用了**JavaFX**的知识，请按以下代码实现任务需求：

```
StarFormUtils.errorMessage(e.getMessage());
```

2）以上代码中的**e.getMessage()**用于获取异常信息，该信息在**getStarService**中创建。代码如下：

```
throw new ServiceInitException("核心文件丢失，请重新安装本游戏");
```

3）终止应用程序运行的代码：

```
System.exit(0);
```

3、 验证与测试：

3-1. 在**src**目录下找到**bean.conf**文件。

3-2. 修改该文件的名称，文件名任意。

3-3. 运行该项目，观察是否会有相应的错误提示弹窗：

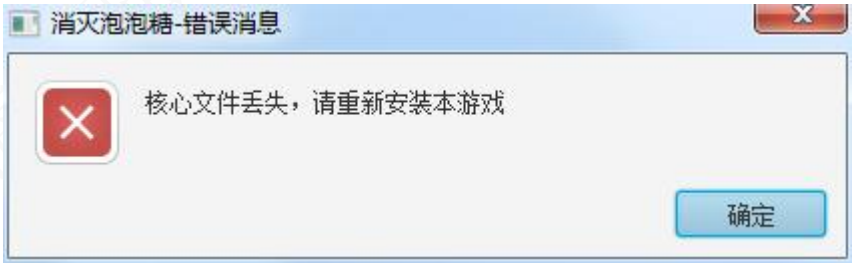


图 **6-1**

3-4. 修改该文件的名称为：**bean.conf**。

3-5. 运行该项目，观察是否会显示正常的泡泡糖游戏界面。

7、任务 2 – 创建自定义异常类

1、 任务描述：

1-1. 任务**1**通过捕捉已有异常，提升了**getStarService**的健壮性，保证“动态加载业务类”功能函数在失去核心配置文件的情况下，不出现致命错误。

1-2. 本任务需要体验如何创建一个自定义异常类：**NoClearedStarsException**。

1-3. 该异常类用来描述：当点击界面的泡泡糖时，可能会出现不能消除的情况。此时需要一个名称鲜明并且提示信息到位的异常对象予以标注，当发生该异常时能准确的定位到发生的位置，再根据提示信息进行相应的修改，方便了程序的测试与维护。

1-4. 所有业务异常类都应该继承**java.lang.Exception**。

2、 推荐步骤：

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

2-2. 创建名为**NoClearedStarsException**的类，并继承**Exception**。

2-3. 创建一个无参构造器，调用父类的构造器。

2-4. 创建一个带参构造器，参数为用户自定义的提示信息字符串，它同样需要调用父类的带参构造器。

+ 提示

1） 快速创建构造函数：

1-1. 在**NoClearedStarsException**中单击右键，选择**Source à Generate Constructors from superclass**。

1-2. 在出现的对话框内容中，勾选**0**参和**1**参字符串类型的构造器。

3、 验证与测试：

无。

8、任务 3 – 使用自定义异常类

1、 任务描述：

1-1. 任务2已经完成了自定义异常**NoClearedStarsException**的创建。

1-2. 本任务利用该类为**StarServiceImpl**类的**tobeClearedStars**函数设计业务异常抛出流程，确保在没有【待消除泡泡糖】的情况下，可以及时通知游戏界面。

1-3. 本任务将全方位实现一个自定义异常的申明、抛出、捕捉、处理过程。

1-4. 异常实现流程如下：

1-4.1. **StarServiceImpl**的**tobeClearedStars**方法，负责获取【被点击】泡泡糖四周的【同色】【待消除】泡泡糖。

1-4.2. 当**tobeClearedStars**方法未找到任何【待消除】泡泡糖时，需要通过抛出**NoClearedStarsException**异常，通知调用它的游戏界面。

1-4.3. 游戏界面负责调用**tobeClearedStars**，因此有义务捕捉**NoClearedStarsException**异常。

1-4.4. 如果该异常被抛出，执行的异常处理手段是：不执行消除操作及消除泡泡糖的后续业务流程。

2、 推荐步骤：

2-1. 定位到业务接口：**cn.campsg.practical.bubble.service.StarService**

2-2. 为**tobeClearedStars**添加：【异常抛出】申明语句。

2-2.1. 在方法尾部增加【异常抛出】申明语句（**throws**）。

2-3. 定位到业务类：**cn.campsg.practical.bubble.service.StarServiceImpl**

2-4. 为**tobeClearedStars**添加：【异常抛出】申明语句。

2-4.1. 在方法尾部增加【异常抛出】申明语句（**throws**）。

+ 业务说明

1）当某个函数需要抛出一个继承了**Exception**的自定义异常时，**Java**语法规定必须在函数尾部通过**throws**关键字申明该异常。

2-5. 根据业务适时抛出异常：

2-1.1. 定位到方法**tobeClearedStars**注释处。

2-1.2. 在注释处，抛出自定义异常**NoClearedStarsException**对象。

+ 提示 ：If判断结果为**true**，表示无可消除的泡泡糖，这就是异常抛出点。

2-6. 捕捉并处理**NoClearedStarsException**异常。

2-1.3. 定位到类：**cn.campsg.practical.bubble.MainForm**

2-1.4. 找到事件处理类**StartEventHandler**的处理方法**handle**

2-1.5. 在**tobeClearedStars**函数调用处编写异常捕捉代码。

2-2. 进行异常处理：

2-2.1. 界面如果发现用户点击的泡泡糖四周无【待消除的泡泡糖】，那么应该终止所有业务逻辑处理，直接退出点击事件。

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，进行以下测试：

3-2.1. 点击可消除泡泡糖时完成消除；

3-2.2. 点击不可消除泡泡糖时，程序无任何操作；

3-2.3. 继续点击可消除泡泡糖能完成消除功能。

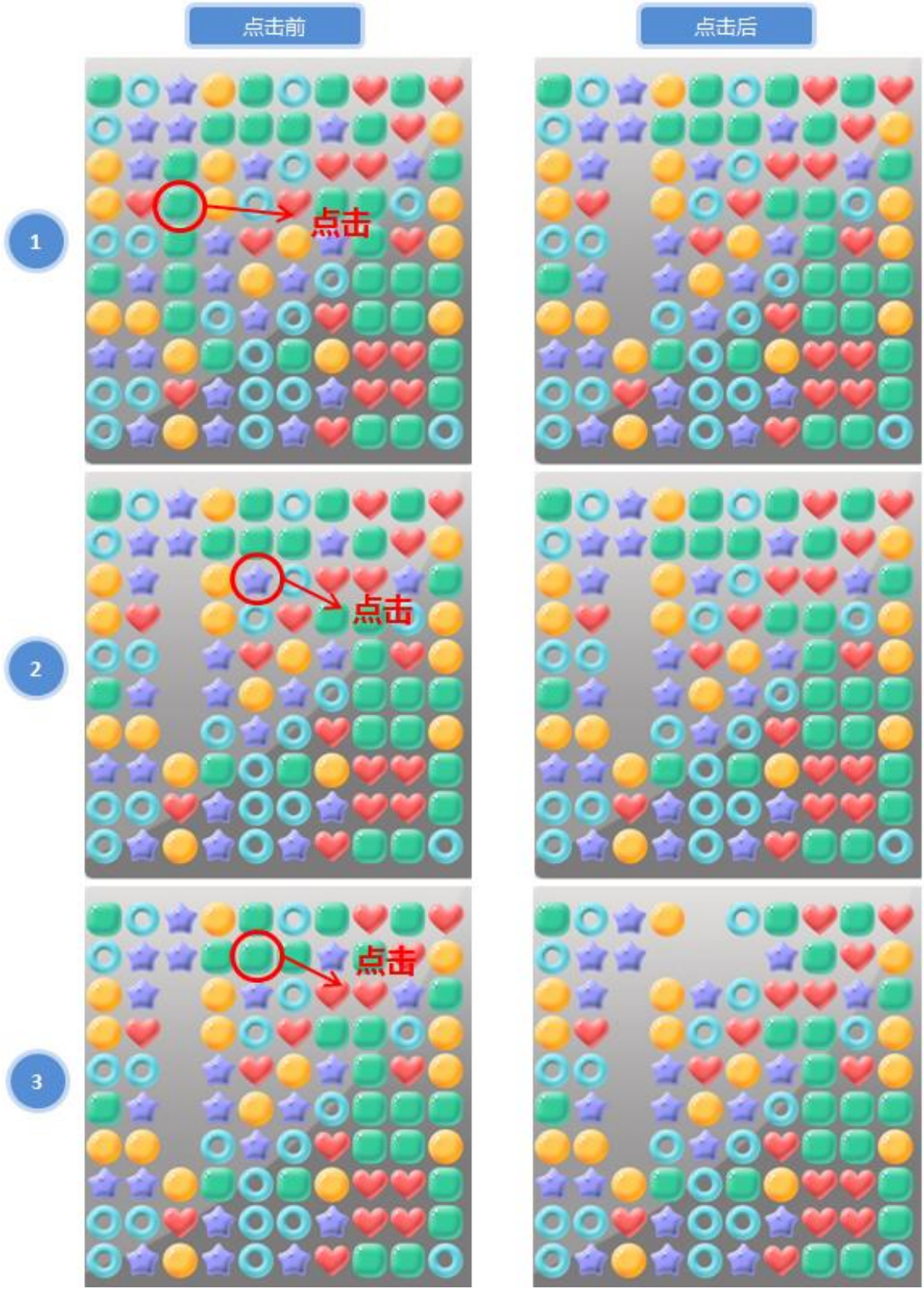


图 **8-1**

9、场景总结

Q1. Java中的异常有何作用？如何捕捉？

1. Java中的标准函数具有四大重要特性：

1-1. 函数名：表达函数存在的意义。

1-2. 参数列表：表达函数计算过程中的**"**已知条件**"**。

1-3. 返回值：表达函数计算业务后的结果。

1-4. 异常：如果计算过程发生了错误需要告知函数的调用者（千万别自己随意处理）。

例如：我们将电脑开机按钮理解为函数，按下开机按钮属于函数调用，看到**windows**界面属于返回，那么因为硬件问题导致电脑发出蜂鸣声就属于异常，电脑并不会主动处理硬件故障，而是通知用户修理。

2. 为了保证程序运行的健壮性，公共函数的调用者都需要了解被调函数可能抛出的异常，并做好相关处理。

Q2. 在您的项目中何时选择捕捉异常？何时选择抛出异常？

1. 以下两个场景是抛出异常的最佳实践：

1-1. 公共函数，该函数开发完毕后可能被多个程序员使用，为了应对不同程序员的需求，功能函数遇到错误情况，可以直接抛出异常告知调用方，函数运行出现了错误。

例如：本场景中的**ServiceInitException**异常

1-2. 层级间的交互，业务层遇到计算错误的情况需要通知软件使用用户，由于业务层无法显示美观的错误提示对话框，需要将错误信息传递给界面层，界面层的职责是显示人机交互界面，因此在获得错误异常后，可以将异常转换成美观的错误提示对话框。

例如：本场景中的**NoClearedStarsException**异常

2. 总结：异常往往在函数与函数、层与层之间使用，函数调用方一般扮演异常捕捉者、函数定义方一般扮演异常的抛出者。

注意 :异常不同于返回值 ,它能更确切的表达计算错误的意图 ,返回值则表示函数的计算结果。

Q3. 您是否具有自定义异常的创建经验？它有何作用？

1. 自定义异常具有以下两个作用：

1-1. 异常名用于描述该异常的作用。

1-2. 异常也是类，为了保证异常捕捉方理解，异常中可以增加额外的属性。

2. 说明：自定义异常可以分别继承自**Exception**和**RuntimeException**。

2-1. 继承**Exception**，表示该异常属于业务异常，函数调用方必须捕捉。

2-2. 继承**RuntimeException**，表示该异常属于语法异常，一般不必捕捉。

2-3. 自定义异常中，较为常见的是继承**Exception**。