**tzl扫雷项目**

**程序开发说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| 日 期 | 版 本 | 说明 | 作 者 |
| 2023.04.01 | V1 | 第一次迭代 | 侯皓斐，刘铭宸，甘凤轩 |
| 2023.04.03 | V2 | 第二次迭代 | 侯皓斐，刘铭宸，甘凤轩 |
| 2023.04.05 | V3 | 第三次迭代 | 侯皓斐，刘铭宸，甘凤轩 |
|  |  |  |  |

**目 录**

[1引言 3](#_Toc131625035)

[1.1编写目的 3](#_Toc131625036)

[1.2定义 3](#_Toc131625037)

[1.3参考资料 4](#_Toc131625038)

[2产品设计 4](#_Toc131625039)

[2.1版本1 5](#_Toc131625040)

[2.1.1 需求-测试-产品-重构 5](#_Toc131625041)

[2.1.1.1 需求 5](#_Toc131625042)

[2.1.1.2 测试设计 6](#_Toc131625043)

[2.1.1.3 代码设计 6](#_Toc131625044)

[2.1.1.4 重构 7](#_Toc131625045)

[2.2版本2 7](#_Toc131625046)

[2.2.1 需求-测试-产品-重构 7](#_Toc131625047)

[2.2.1.1 需求 7](#_Toc131625048)

[2.2.1.2 测试设计 7](#_Toc131625049)

[2.2.1.3 代码设计 7](#_Toc131625050)

[2.2.1.4 重构 8](#_Toc131625051)

[2.3版本3 8](#_Toc131625052)

[2.3.1 需求-测试-产品-重构 8](#_Toc131625053)

[2.3.1.1 需求 8](#_Toc131625054)

[2.3.1.2 测试设计 8](#_Toc131625055)

[2.3.1.3 代码设计 9](#_Toc131625056)

[2.3.1.4 重构 9](#_Toc131625057)

[3小组分工 9](#_Toc131625058)

# 1引言

## 1.1编写目的

本Scrum项目组旨在**二次开发**一款名为“扫雷”的益智小游戏，我们将采用**Scrum开发和测试驱动开发（TDD）**结合的方法，**为现有的扫雷游戏加入排行榜功能和难度设置**。游戏的主要目标是在最短时间内找出所有非雷格子并避免踩雷。游戏包括初级、中级、高级和自定义四种难度，玩家可以自定义雷区大小和雷数。该游戏在计算机上单机运行，玩家可以通过游戏界面与游戏进行交互操作。

我们为了保证Scrum和TDD开发方法的顺利进行，将进行以下的保证活动：

* 采用迭代开发方法，每个迭代周期一般为1-2天
* 每个迭代周期包含需求分析、设计、编码、测试、回顾等环节
* 采用Scrum仪式，包括每日站立会议、迭代回顾会议、迭代计划会议等
* 先编写单元测试用例，为通过单元测试用例而编写程序
* **积极的重构原先紧耦合的代码（UI和逻辑），使之能够进行单元测试，回归测试**

开发工具使用IDEA 2022 Ultimate与Java 1.8。测试使用Junit4。

由于作业时间紧张，我们应该在一周内完成规划的任务。

## 1.2定义

* Scrum：一种敏捷软件开发方法，主要用于快速开发高质量的软件。
* 迭代开发：将开发工作分为若干个迭代周期，每个迭代周期都包含需求分析、设计、编码、测试和回顾等环节。每个迭代周期都会交付可工作的软件部分，不断重复这个过程，逐步完善软件功能和性能。
* 每日站立会议（Daily Scrum）：每日站立会议是Scrum的一个仪式，团队成员在此会议上汇报前一天的工作、今天的工作计划和遇到的问题，以便团队成员之间相互了解和协作。
* 风险管理：是一种管理和控制风险的方法，旨在避免或减轻潜在的负面影响。
* 扫雷棋盘（Minefield）：游戏界面上的方格网格，每个网格中可能包含地雷或数字。
* 地雷（Mine）：游戏中的障碍物，如果玩家打开一个地雷，游戏将立即结束。
* 数字（Number）：游戏中非地雷网格上显示的数字，用于指示周围8个网格中地雷的数量。
* 标记（Flag）：玩家用于标记地雷的红旗，以避免误点。
* 爆炸（Explosion）：如果玩家误点了一个地雷，游戏将结束，并且屏幕将显示爆炸效果。
* 挑战排行榜（Challenge leaderboard）：记录玩家在挑战模式下完成游戏的时间和排名的排行榜。

## 1.3参考资料

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 标题 | 文件名称 | 发表日期 | 资料来源 |
| 1 | TDD开发参考资料 | Test-Driven Development: By Example | 2002年 | Kent Beck，Addison-Wesley Professional出版社 |
| 2 | Scrum开发参考资料 | Agile Software Development with Scrum | 2001年 | Ken Schwaber  Prentice Hall出版社 |
| 3 | Java编程参考资料 | Effective Java | 2001年 | Joshua Bloch， Professional出版社 |
| 4 | JUnit参考资料 | JUnit Cookbook | 2005年 | J.B. Rainsberger  O'Reilly出版社 |
| 5 | 二次开发参考资料 | Head First Design Patterns | 2004年 | Eric Freeman,  O'Reilly出版社 |

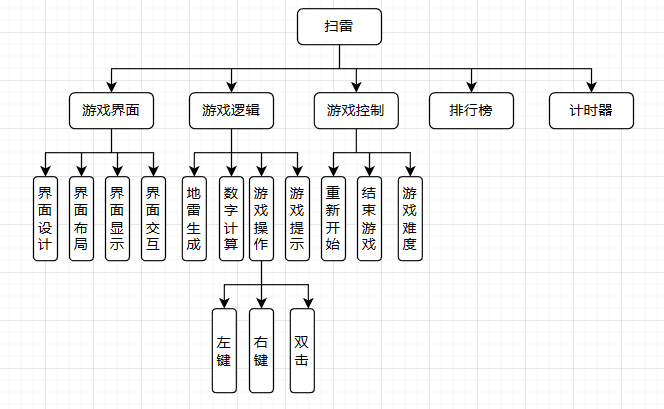
# 2产品设计

我们于2023年3月31日进行开发前Scrum组织会议，内容主要如下。

我们进行了迭代规划。原开源项目采取紧耦合的设计思路，导致后续加入排行榜和难度选择十分困难，我们为了后续迭代和测试的需要，我们需要对原项目进行重构将Java.awt编写的UI与游戏逻辑分开（第一轮迭代，使之能够进行单元测试）。后续每一轮迭代，通过不断需求细化，设计测试用例，编写代码，进行重构为其增添新的内容。

我们通过项目目录大致分析了我们现有项目的模块与需求。

我们大致分析了我们需要实现的各项功能需求，并和原先的功能需求结合。形成大致的模块划分图如下：



## 2.1版本1

基于原有的代码重构，实现游戏界面，和完整的游戏逻辑。

### 2.1.1 需求-测试-产品-重构

2.1.1.1 需求

|  |  |  |
| --- | --- | --- |
| 需求 | 需求描述 | 编号 |
| 显示主界面 | 显示游戏菜单栏。显示默认简单难度的雷区布局，地雷计数器 | 1 |
| 地雷生成 | 在雷区随机生成相应数量的地雷 | 2 |
| 数字计算 | 根据雷区中地雷的位置和数量，在周围的方格中计算出数字 | 3 |
| 打开方块 | 如果该方块为雷，则游戏结束。如果该方块为空，则递归打开与其相邻的所有空方块和数字方块，直到不能再打开。如果该方块为数字方块，则显示该数字 | 4 |
| 标记地雷 | 用户在雷区中右键单击该方块，标记为地雷 | 5 |
| 双击已经打开的数字方块 | 雷区更新，周围一圈变白。 | 6 |
| 判断游戏结束 | 判断是否有未被挖开的雷，有则判定游戏失败；无则判定游戏胜利。  若直接打开地雷所在的方格，直接失败。 | 7 |

2.1.1.2 测试设计

我们可以看到第一次迭代内的很多内容是UI相关的，UI相关的和IO部分的代码我们是无法进行测试的，所以我们围绕着地雷生成和数字计算以及游戏结束判断（非失败）进行了测试设计。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试设计 | 测试需求描述 | 对应测试脚本 | 需求（编号） |
| 地雷生成 | 随机生成默认难度特定大小下规定的地雷数目，数目对的上（保证指定的位置不是地雷，即第一个点的位置） | LayMineTest | 2 |
| 数字计算 | 计算方格周围地雷数没有错误 | ComputeBombTest | 3 |
| 判断游戏结束 | 判断所有地雷已经挖出来，验证符合题意 | CheckWinTest | 7 |

2.1.1.3 代码设计

重构时竭尽全力地将逻辑结构的数据和游戏逻辑相关代码与UI解耦合。

采取了数据仓库设计模式。

采取了如下方案：

* 分离数据：将逻辑结构的数据从UI代码中分离出来，形成一个独立的数据结构模块tools。数据结构模块定义清楚数据的类型、属性和操作，不涉及具体的UI操作。 将其设置为全局静态类，大家公用数据。
* 分离数据操作：将对逻辑结构的数据操作的方式顺带提取，形成静态方法（LayBomb，StaticToo中的方法）。
* 实现逻辑操作接口：在逻辑结构（bean）的数据模块中定义逻辑操作接口，例如将开启某个网格、标记某个网格转化为get或set的方法。这些接口应该是抽象的，不依赖于具体的UI实现。
* 实现UI操作接口：在UI模块中实现UI操作接口，例如在点击某个网格时，调用逻辑操作接口来进行get或set。
* 注册逻辑操作回调：在UI模块中注册逻辑操作回调函数（listenner），以便在用户点击网格时，调用逻辑操作接口实现对应的操作。

打开方块，标记地雷，双击已经打开的数字方块，在业务逻辑层面也是可以处理的，例如，针对“打开方块”这个需求，业务逻辑可以考虑：是根据该方块所对应的棋盘格的取值，给出相应处理。如果棋盘格取值是雷，则游戏状态设置为结束；如果方块棋盘格取值为空，则获取其邻域所有棋盘格，然后循环对每个棋盘格，继续执行本函数；如果该方块对应棋盘格取值为数字，则标记该方块为显示数字状态。这个业务逻辑是可以与界面无关。但实际上整个状态机并未实现，我们如果**如此重构，耗费的精力和时间**会远远超过一个迭代周期的范围，也会影响项目交付时期。我们反过来思考，**这里的结构混乱并不会影响我们后续功能的增加**，因为这里的逻辑和后续加入的逻辑相对分离，所以我们此处不考虑继续分离。

2.1.1.4 重构

我们本次迭代本身安排就是二次开发和TDD的重构迭代。

## 2.2版本2

实现排行榜功能（增加menu中的新UI，增加新的数据处理部分，加入计时器）

### 2.2.1 需求-测试-产品-重构

2.2.1.1 需求

|  |  |  |
| --- | --- | --- |
| 需求 | 需求描述 | 编号 |
| 游戏计时器 | 开始计时，记录游戏进行的时间 | 8 |
| 时间记录功能 | 记录玩家本次游戏的用时，并将其记录在对应难度的排行榜中 | 9 |

2.2.1.2 测试设计

游戏计时器的功能显然易见的无法进行单元测试。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试设计 | 测试需求描述 | 对应测试脚本 | 需求（编号） |
| 时间记录功能 | 并将其记录时间成功在对应难度的排行榜中 | HeroSetTest | 9 |

2.2.1.3 代码设计

经过第一轮的重构迭代，我们发现加入新的功能变得十分轻松。我们在迭代后的架构上（数据仓库模式）下进行第二次设计，为了实现新的功能在逻辑和UI上都要进行一定程度的修改。

|  |  |
| --- | --- |
| 代码 | 代码设计描述 |
| 游戏计时器  timer/TimerListener | 使用触发器的方式进行计时，在数据仓库中加入timecount。每过1s自动更新时间，从而实现游戏的计时。 |
| 时间记录功能  tool/StaticTool | 利用静态TreeSet记录游戏记录 |
| 请求姓名  dialog/HeroDialog | UI部分，获得用户姓名，从而记录入游戏排行榜。 |

2.2.1.4 重构

我们需要重构游戏胜利UI游戏结束跳转部分的代码（listenner/Listener/isWin），从而游戏结束时，进入我们新的Dialog，从而记录时间。

## 2.3版本3

实现难度切换功能（增加menu中的新UI，增加新的数据处理部分，进行较大规模重构）

### 2.3.1 需求-测试-产品-重构

2.3.1.1 需求

|  |  |  |
| --- | --- | --- |
| 需求 | 需求描述 | 编号 |
| 游戏难度选择 | 用户选择游戏难度  根据用户选择的难度确定雷区大小和地雷数量，生成相应的雷区和地雷  用户选择自定义难度时，需在弹出的设置对话框中设置雷区大小和地雷数量，若设置不合法，弹出提示信息并要求重新设置 | 10 |

2.3.1.2 测试设计

|  |  |  |  |
| --- | --- | --- | --- |
| 测试设计 | 测试需求描述 | 对应测试脚本 | 需求（编号） |
| 游戏难度选择 | 用户选择游戏难度  根据用户选择的难度确定雷区大小和地雷数量，生成相应的雷区和地雷 | SetLevelTest | 10 |

2.3.1.3 代码设计

实际上难度与（行数，列数，雷数）三元组为等价关系。

所以我们需要提取出所有行数，列数，雷数成为公共可变参数，放入数据仓库。

|  |  |
| --- | --- |
| 代码 | 代码设计描述 |
| 新的UI  panel/BombJPanel | 行数可变的渲染 |
| 难度设定和判断  tool/StaticTool | 利用（行数，列数，雷数）三元组  UI中处理数据不应写死，而应访问数据 |
| 自定义Dialog  dialog/HeroDialog | UI部分，获得  判断格式合法，以及数目是否合法 |

2.3.1.4 重构

加入难度后，很多地方代码都要进行重构，测试也需要重构。

代码重构的区域，包括开始游戏，计时器和排行榜模块，还有很大一部分的UI要分出三种情况的讨论。

而我们测试的相关部分，比如排行榜模块的代码也应该重构。

# 3小组分工

本Scrum项目组开发成员共三人，包括程序员、测试人员、设计师。

其中侯皓斐同学兼顾Scrum master与程序员，承担了最重要的工作，做出了40%左右的贡献。

刘铭宸和甘凤轩同学先编写测试后编写代码，为项目做出了突出贡献。每个人做出了30%左右的贡献。