**代码重构**

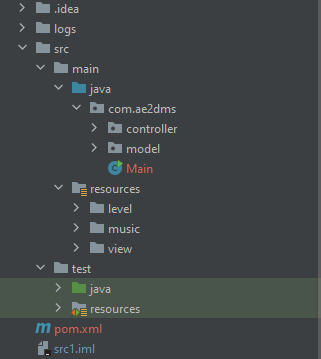
首先将项目文件重新组织，按照maven项目常用项目结构重构项目目录，所有文件位于src目录下

src/main/java 项目的源代码所在的目录

src/main/resources 项目的资源文件所在的目录

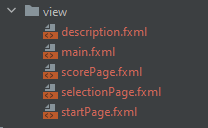
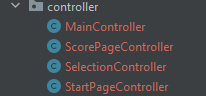
src/test/java 测试代码所在的目录

src/test/resources 测试相关的资源文件所在的目录

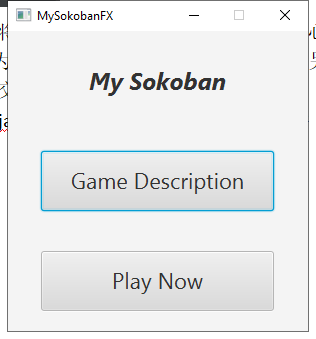


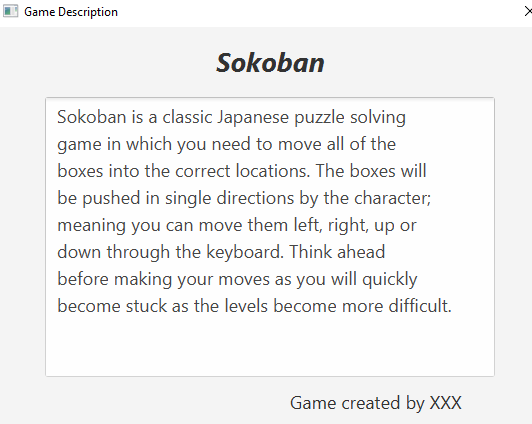
针对项目进行了MVC设计模式架构的重构，将原先的GameEngine等实现游戏核心功能的类归为Model，实现前端页面的fxml文件作为view存在resources/view目录下，另外对应每个fxml页面新建controller来负责前后端的交互。

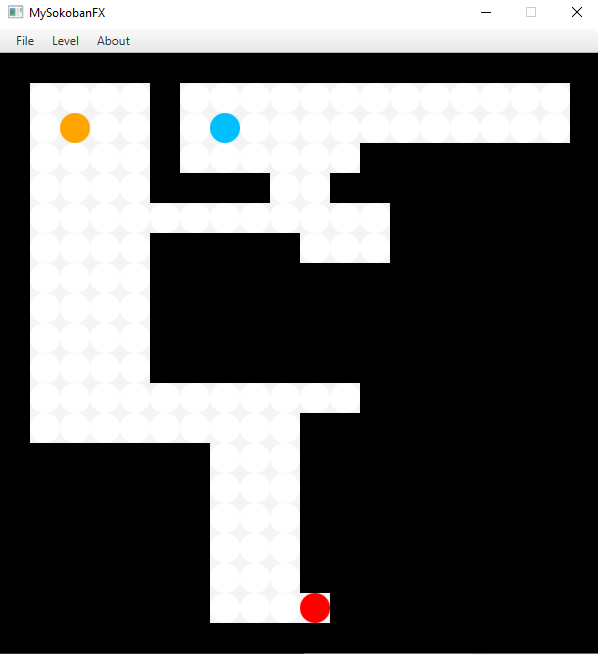
通过JavafxSceneBuilder2.0图形界面软件设计javafx页面并导出以下5个fxml文件

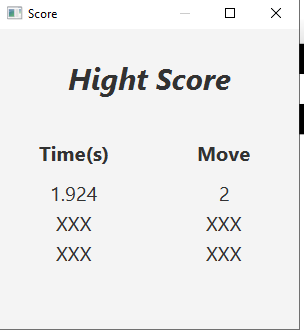
对应的controller

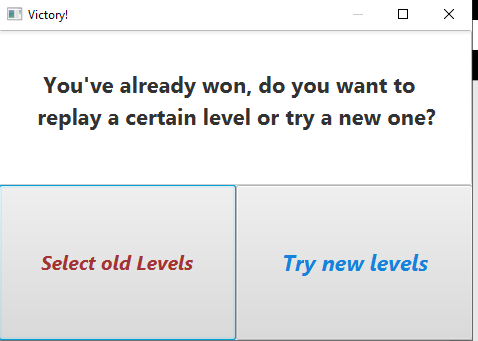
分别对应以下5个页面

main.fxml, controller:startPageController

游戏描述界面description.fxml

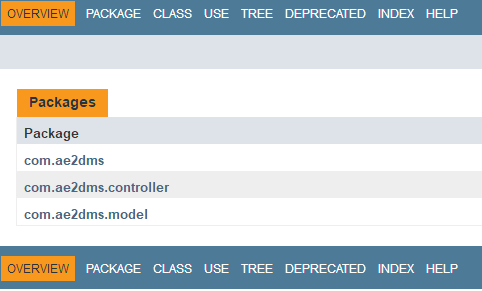
游戏主界面main.fxml controller: mainController

游戏结束后的分数界面scorePage.fxml controller：scorePageController

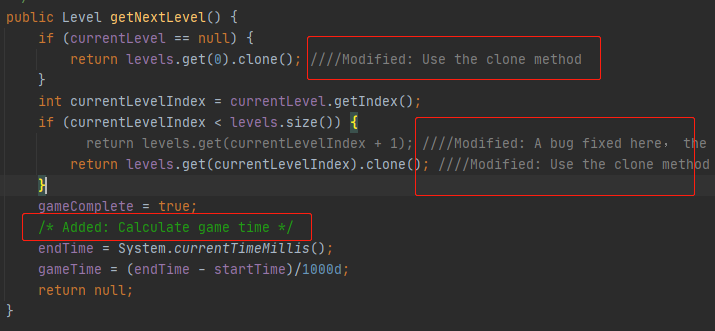


关卡选择界面 selectionPage.fxml controller: selectioncontroller

在此过程中，将原有的main函数拆分为了main和mainController以符合MVC架构和面向对象的设计原则。

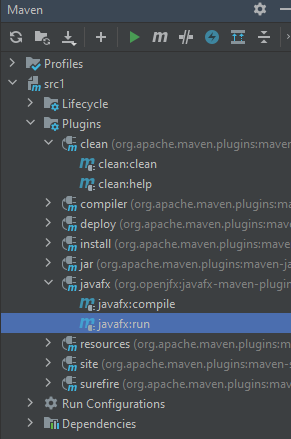
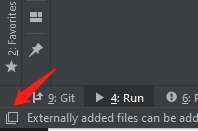
同时为了代码便于维护，为每个类，每个函数都做了javadoc的注释并生成了Javadoc文档，

在逻辑复杂的代码处和做了比较大修改的地方也添加了注释便于理解。



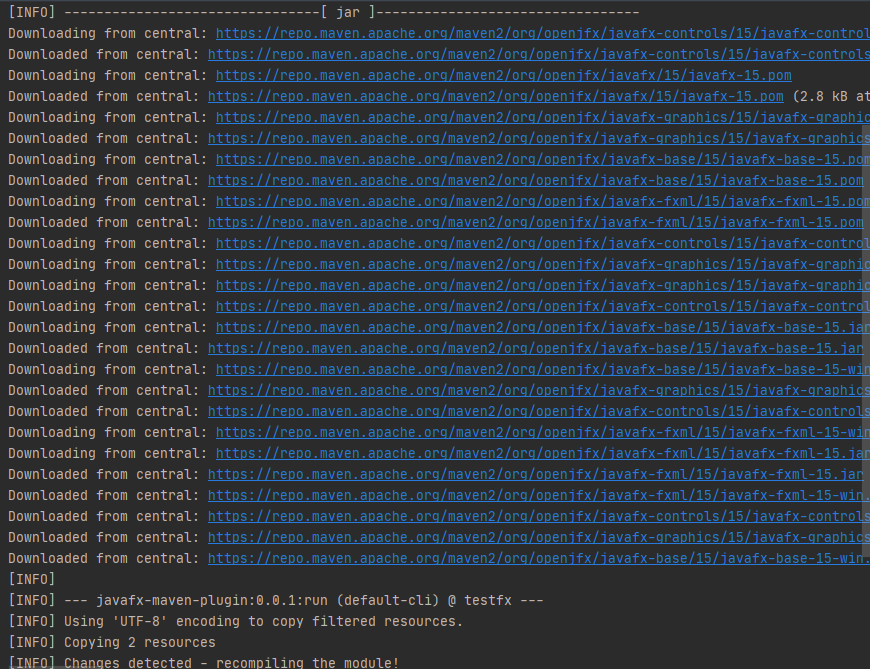
项目使用了Maven进行构建，详情见pom.xml文件。按照Javafx官网上的说明，通过IDEA向导使用模板新建了基于Maven构建的Javafx项目，除了添加Javafx依赖库以外

还使用了如下两个插件

IDEA窗口左下角点开小窗口选择maven，界面右侧会出现maven相关命令

使用Plugins下的clean->compiler:compile->javafx:run即可完成编译运行等操作

build项目之后，Maven会自动下载项目所需要的库，Javafx需要的库已添加到pom.xml文件中。



**修正bug**

**原本应有5个关卡，但只能玩到1,3,5三个关卡，问题出现在GameEngine的getNextLevel上**

**int currentLevelIndex = currentLevel.getIndex();**

**原本要获取的当前关卡的index是按1,2,3,4,5来存储的**

**return levels.get(currentLevelIndex + 1);**

**而levels是list结构是从0开始存储的**

**现在的代码如果currentLevelIndex+1则是2**

**Get（2）获取的是第三关的关卡**

**所以要获取下一个关卡将+1去掉就可以了**

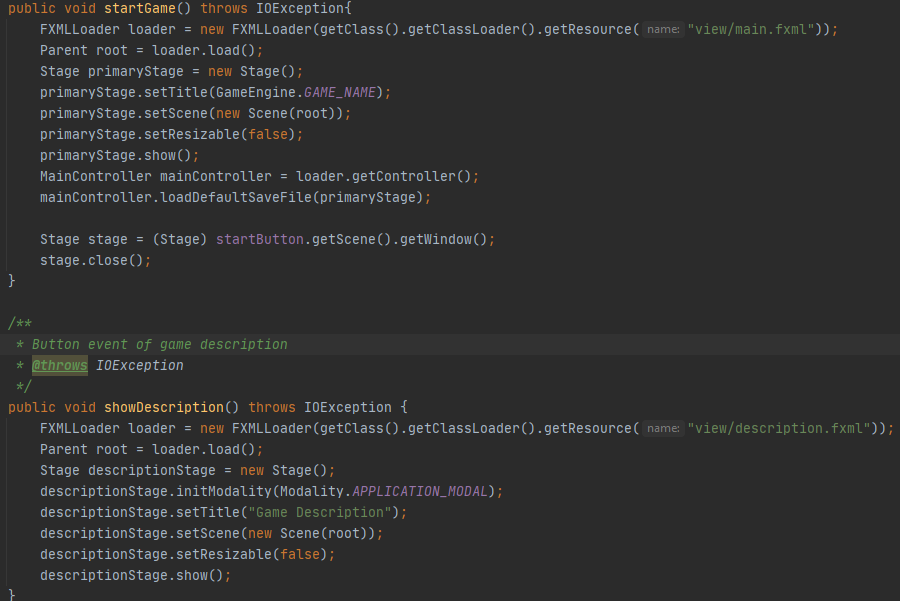
**return levels.get(currentLevelIndex)**

**新添加的功能**

1. TWO BUTTON: a button to start game and a button to provide information of the game.

涉及代码： Main.java 修改了程序入口点，以startPage.fxml页面作为起始页面。

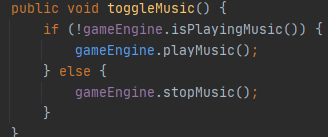
对应的新添加的StartPageController中的startGame()和showDescription()函数负责处理两个按钮的点击事件来打开新的窗口

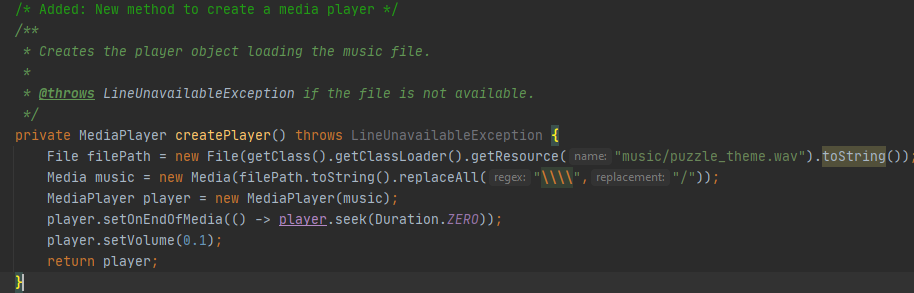


1. 添加背景音乐功能

在游戏主界面，点击Level菜单下toggleMusic可以开启背景音乐，再次点击暂停播放

具体实现代码为mainController下的



以及GameEngine类中createPlayer()函数初始化音乐播放器和startMusic(), stopMusic()函数。

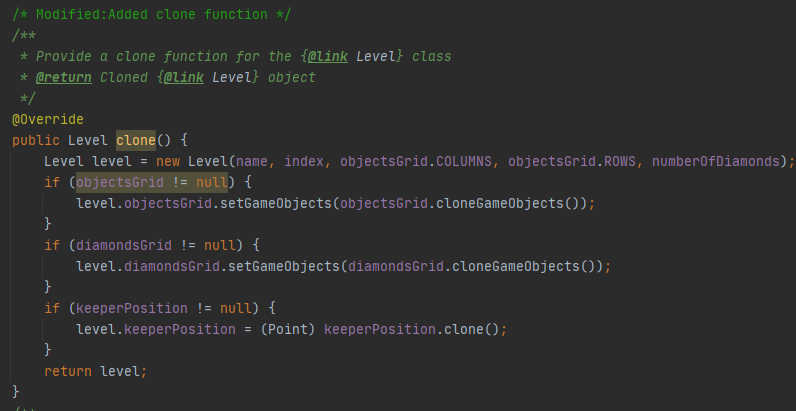
1. Reset Level功能

重置当前关卡为初始状态，快捷键为空格

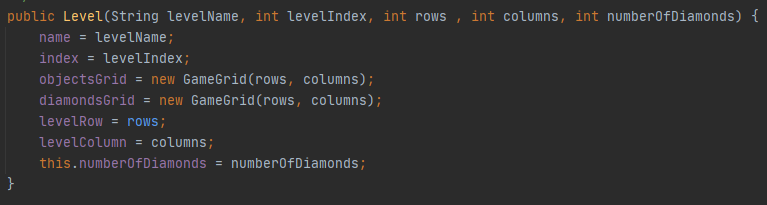
原理是在关卡载入阶段将存储关卡状态的对象存储到内存中，在点击reset Level时提取出来覆盖当前关卡状态（人物位置，箱子位置，钻石位置，地图信息），由于base代码中对关卡状态的操作直接在最初从文件读取出来的数据上进行的，所以我们需要为涉及到关卡状态的类（Level,GameGrid,GameObject等类）添加clone函数，为此我们为这些类添加了实现cloneable的声明，

并重写了clone类，由于在此情景下我们需要深拷贝而不是浅拷贝，同时又为了避免修改base代码中各个类各个属性的封装性（private->public, final），我们手动添加了如下clone函数

Level类

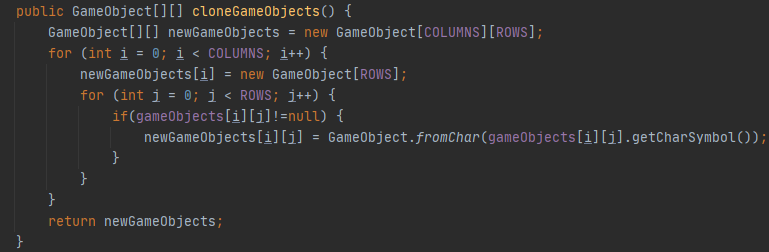


为此还新建了一个构造函数专门用于对属性直接赋值。

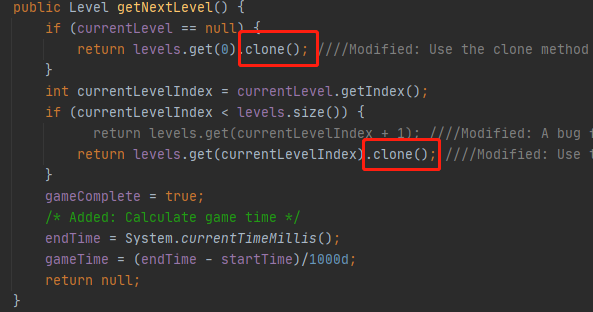


Gamegrid类

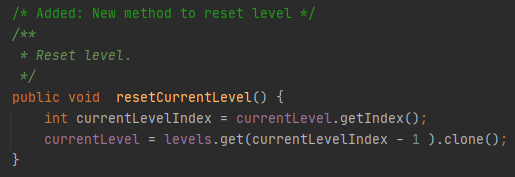
手动new一个GameObject数组对象并赋值



在getNextLevel()函数，即每次载入关卡的核心方法处调用clone，这样我们在内存中就保存了不会被修改的关卡状态数据供我们之后调用

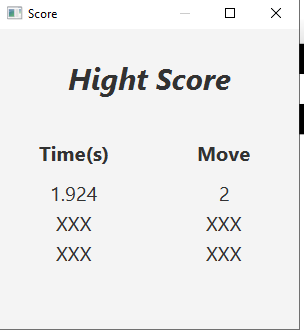


最后，在GameEngine中实现重置关卡的方法



3. A HIGH SCORE pop-up

**分数页面在提示游戏胜利时弹出**

分别显示用时和移动步数，排序由低到高，只显示最短用时和最少步数的前3个

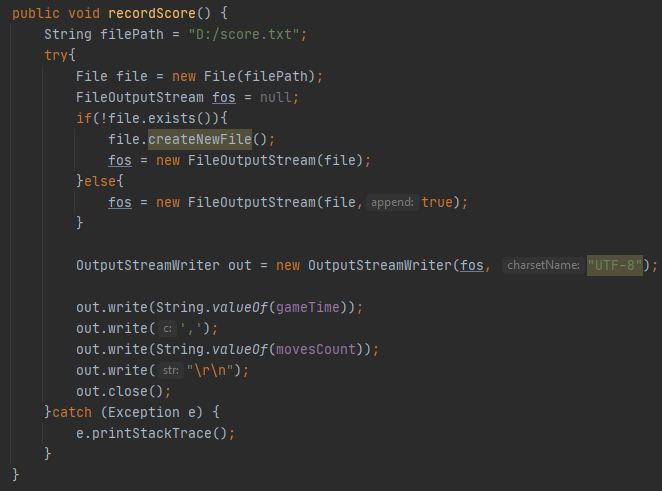
实现方法为mainController里的showHighScore()函数，负责载入scorePage页面的fxml文件且调用scorePageController中的initScorePage方法载入存在本地文件(D:/score.txt文件中的分数）

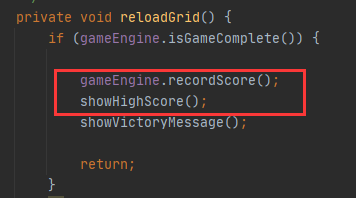


GameEngine中的loadScoreFile负责读取score文件



分数的记录则是在GameEngine类中的

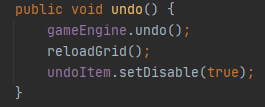


调用则是在mainController显示游戏通关信息的时候调用

1. Undo功能

允许撤回一步操作

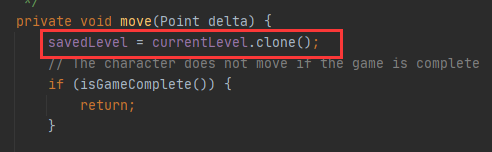
mainController



GameEngine



原理则是在GameEngine下新定义了Level savedLevel对象

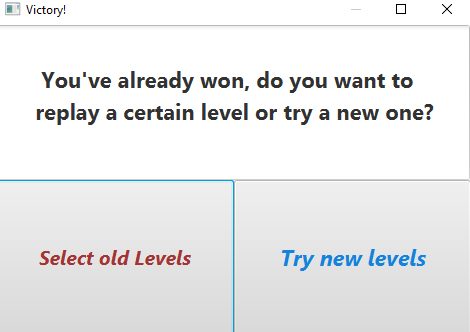
在每次move方法调用的时候使用clone方法临时存储当前关卡状态

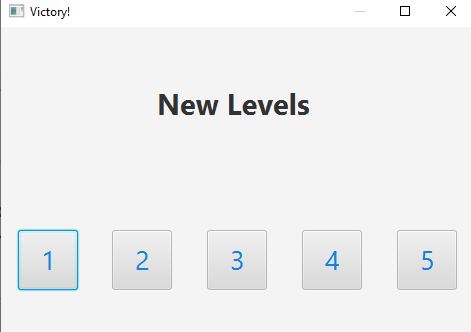
1. save game功能

直接由mainController中的saveGameFile函数实现，将当前关卡状态及之后的关卡全部存储到了skb文件中

1. SelectLevel界面及相应功能

在关闭分数界面后自动弹出

可选原有的5个关卡和新增的5个自定义关卡

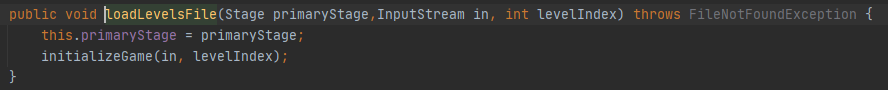


实现方法主要是selectionPage.fxml和selectionController

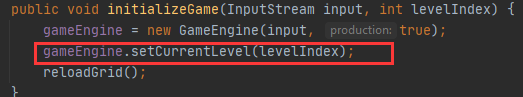
自定义的关卡文件在level/Mylevel文件夹下

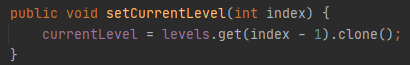
选择哪一关由mainController中的loadLevelsFile方法控制，传入了LevelIndex参数

并在原有的initializeGame函数添加了levelIndex形参

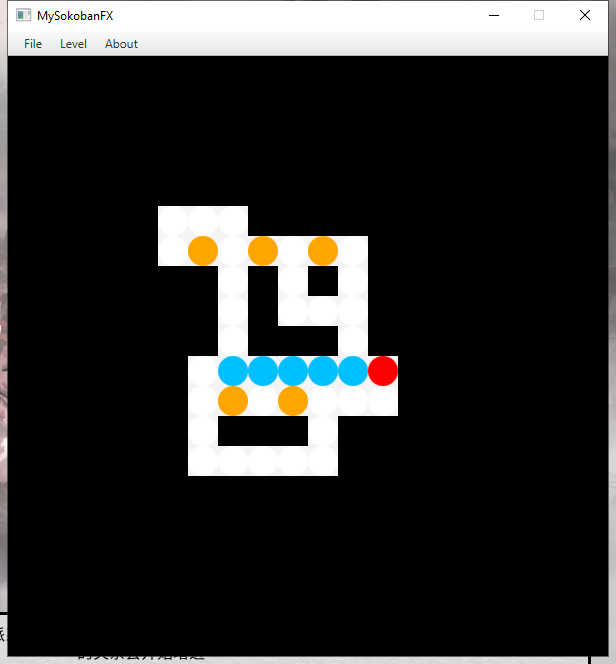
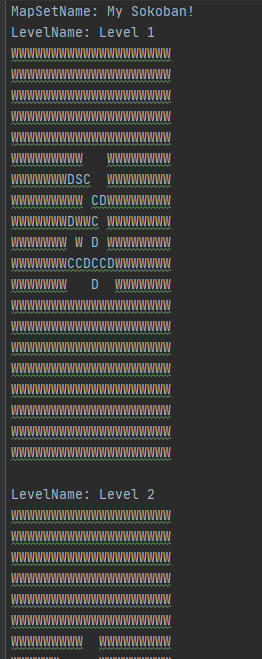


并调用了gameEngine的setCurrentLevel函数（新添加的）



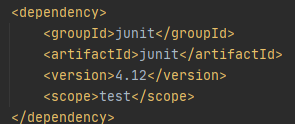


自定义关卡的skb文件

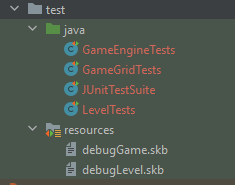


1. Junit

Pom文件添加依赖



文件结构



针对核心类的几个主要函数进行了简单测试

