俞楚凡 24302010004 OOP PJ-1

以下是我的 pj-1 实现思路。

首先，以下是我的项目结构图：

- assets
    - test1.cmd 演示文件
    - test2.cmd 演示文件
    - test3.cmd 演示文件
- src
    - main
        - java 代码目录
            - com.oop.pj1 软件包名
                - controller 控制器
                    - GameController 存放单局游戏实例的核心算法及调用逻辑
                - data 数据层
                    - Constant 全局静态常量
                    - GlobalEntities 为 GUI 设计的图形化静态全局变量集合
                    - Serializer 序列化器，用于状态储存
                - model 业务层
                    - Board 处理棋盘相关逻辑
                    - GameInputValidator 原本用于接受标准键盘输入和文件输入，过渡到图形化界面后仅用于处理文件输入
                    - GomokuBoard 继承自 Board，处理五子棋棋盘相关逻辑
                    - PeaceBoard 继承自 Board，处理简单棋盘相关逻辑
                    - ReversiBoard 继承自 Board，处理翻转棋棋盘相关逻辑
                    - Piece 棋子状态枚举类
                    - Player 处理玩家相关逻辑
                - view 视图层

- AbstractUIConstructor 抽象类图形化 UI 构建器，接口约定
- Messenger 游戏提示弹窗工具类
- UIConstructorPeace 继承自 AbstractUIConstructor，用于构建简单棋盘
- UIConstructorReversi 继承自 AbstractUIConstructor，用于构建翻转棋棋盘
- UIConstructorGomoku 继承自 AbstractUIConstructor，用于构建五子棋棋盘
- Game 游戏主类，程序入口点，同时处理一些全局逻辑

- resources 资源目录
  - icons 弹窗 icon
    - error.png
    - info.png
    - success.png
    - warning.png (unused)
  - save 存档目录
    - pj.game 游戏存档文件

在 lab6 中，我们已经做了一些工作。包括演示模式，五子棋盘的扩大，障碍物以及炸弹等。在 PJ-1 中，我们需要做的工作主要有三件：

1. 从字符界面向图形化界面过渡（基于 javafx）；
2. 游戏状态的存档与恢复;
3. 调整界面窗口大小时，UI 各部分需要动态调整大小。

下面依次来叙述三部分任务的实现思路。

为了实现图形化界面，除了学习 javafx 框架之外，在代码结构上：

添加了 GlobalEntites 类，用于存放一些诸如组件宽高，组件间隔之类的全局变量。由于图形化之后，虽然游戏的核心逻辑没有变化，但是很多与交互相关的逻辑都要与组

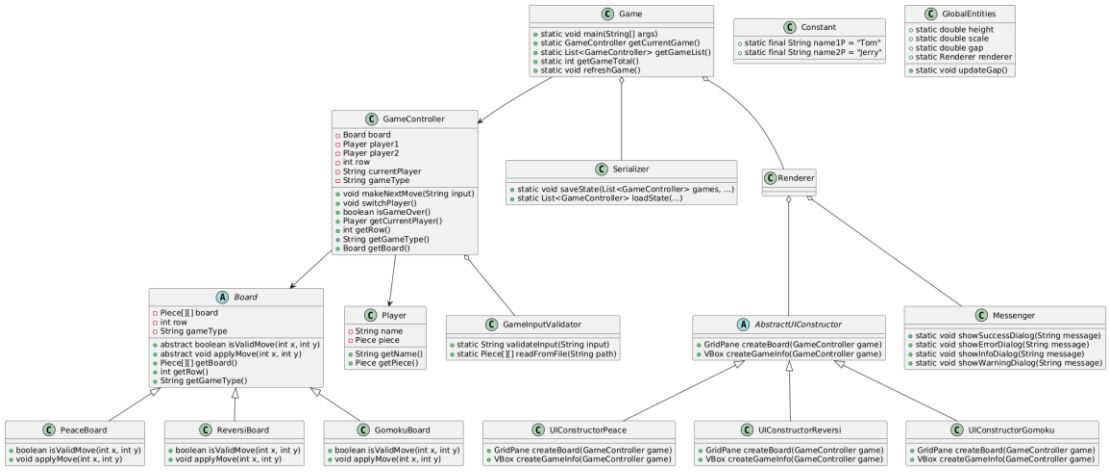
件强绑定，因此对于整体的重构是比较大的。具体来讲，每个指令对应的逻辑基本都挪到了按钮上，所以需要原本的主类 Game 来多承担一些不同层之间的联系任务。类似于前端的 fxcss 是一个非常神奇的东西，将 java 和 css 结合了起来，比较好用。此外，通过这个 PJ 的完成我能够学习到一些前端组件的嵌套的一些通用性的写法。

对于游戏状态的存档与恢复，java 有一个 Serializable 接口。这个接口的作用是将 class 对象转换为序列化的数据存储在文件中，下次读取再反序列化成对象。要使用 Serializable 接口，只需要在类之后加上 implements Serializable 即可。

在本项目中，只需要保存游戏列表 gameList，以及 ListView 相关的数据：当前游戏 ID currentGameID，总游戏数量 gameTotal，当前游戏 currentGame 即可。另外一个细节是，开头不加/斜杠的地址能够直接定位到源代码根目录，查找根目录下的文件。在反序列化之前首先应该判断一下文件非空，否则会抛 IOException。

UI 各部分需要动态调整大小，需要添加一个监听窗口宽度高度变化的监听器。由于在写代码之前不知道有个东西叫做 scale，所以我是设置变量手动设置各组件的宽高来实现的，包括组件之间的间隔。具体细节请详见代码。

有关于代码复用的问题，我认为我对于原本代码的改动是很小的，由于 lab6 的彻底重构提高了代码的解耦性，因此我基本完全没动 GameController 下的核心逻辑，除了一些与键盘交互的部分做了修改。新内容基本上都是以添加为主。添加时尽量做到代码业务的抽离。原代码中所有 System.out.println 控制台输出提示的部分全部改成了 Messenger 弹窗类提示，提升了用户友好性。此外，对于输入指令判断的代码全部可以简化到具体的按钮组件绑定。我做了少量的异常处理，能够处理一些抛出的基本异常。

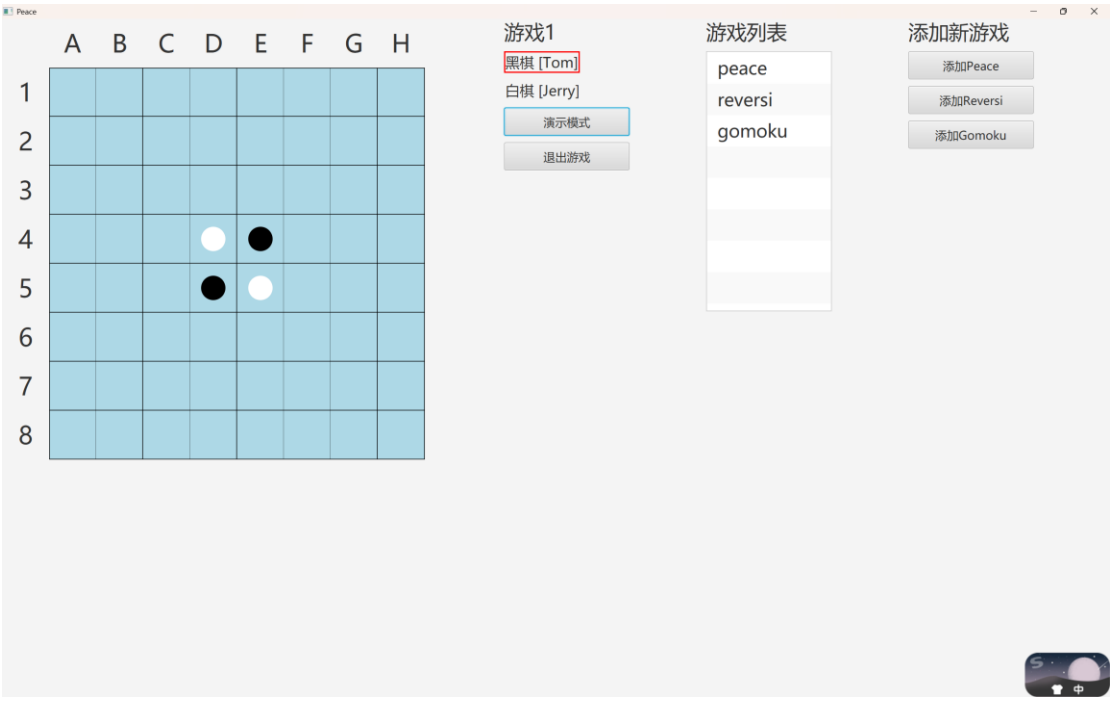


以上是项目主要类的 UML 类图。图中可以清晰的看见各部分之间的关系。Game 主类控制 controller，controller 调度 model 层的业务逻辑，model 层与数据交互，最终返

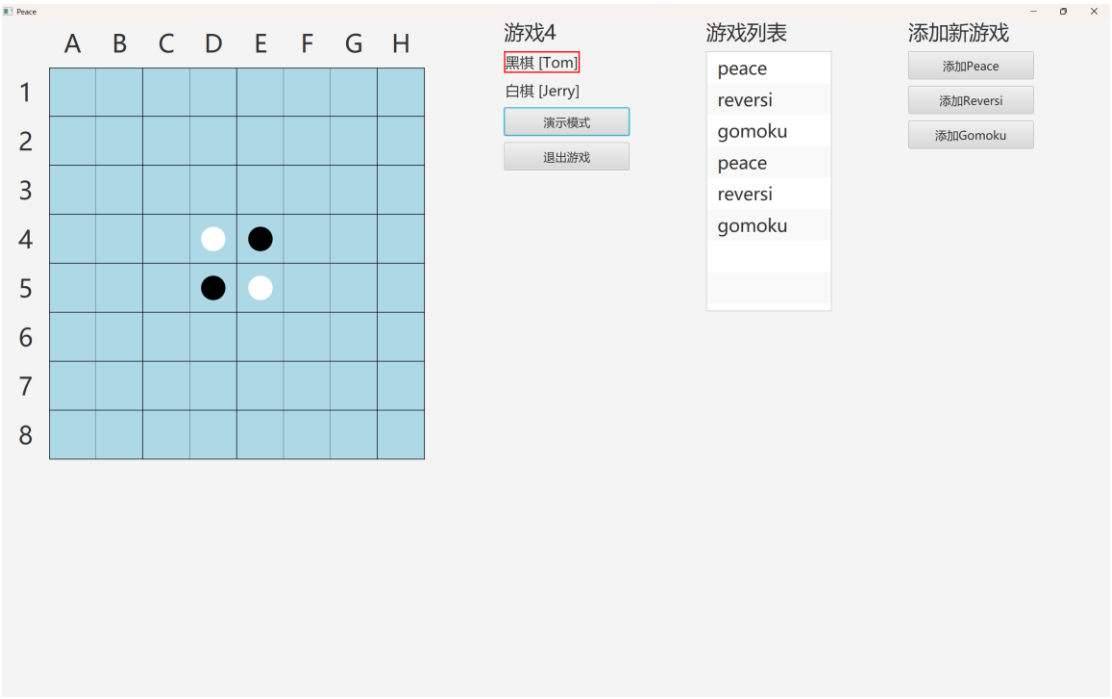
还给主类 Game，再与视图层进行交互。Game 和 GameController 均应该被视为控制器。由于图形化界面逻辑的分散，Game 也同样负责全局的调度。所以，Controller 和 View 与 Model 交互，Model 与 Dao 层交互。是一个比较典型的 MVC 架构。

如果需要加一个新的 2048 游戏，那么首先，2048 的棋盘只有 4*4。需要进行一定的适配，其次，游戏操作的逻辑是上下左右四个按键，棋盘更新的逻辑也需要重写。虽然如此，基本的框架结构是不变的，若模块化做的好，应该只需要修改类似 makeNextMove, updateBoard, checkGameEnd 这样的方法即可。需要扩展新的 Board, AbstractUIConstructor 子类来实现对应的功能。此外，棋子的种类不止黑白两种，而是根据数字的变大会有十几种，可能需要开一个数组来维护棋子类型，并做出相应的修改。

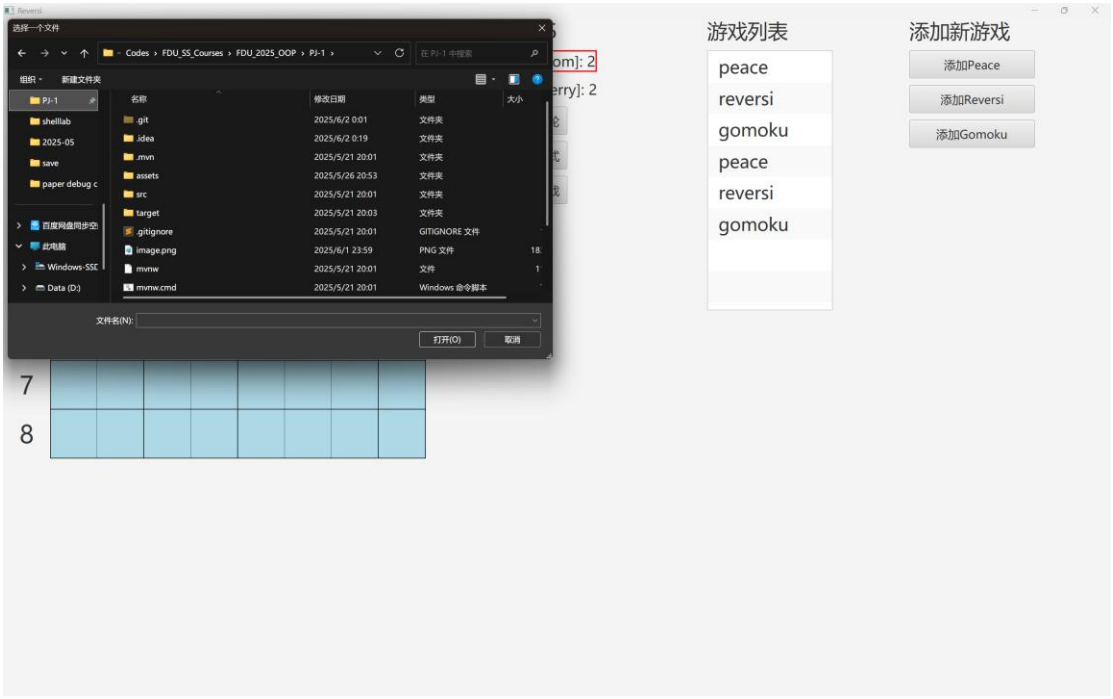
运行过程截图如下。



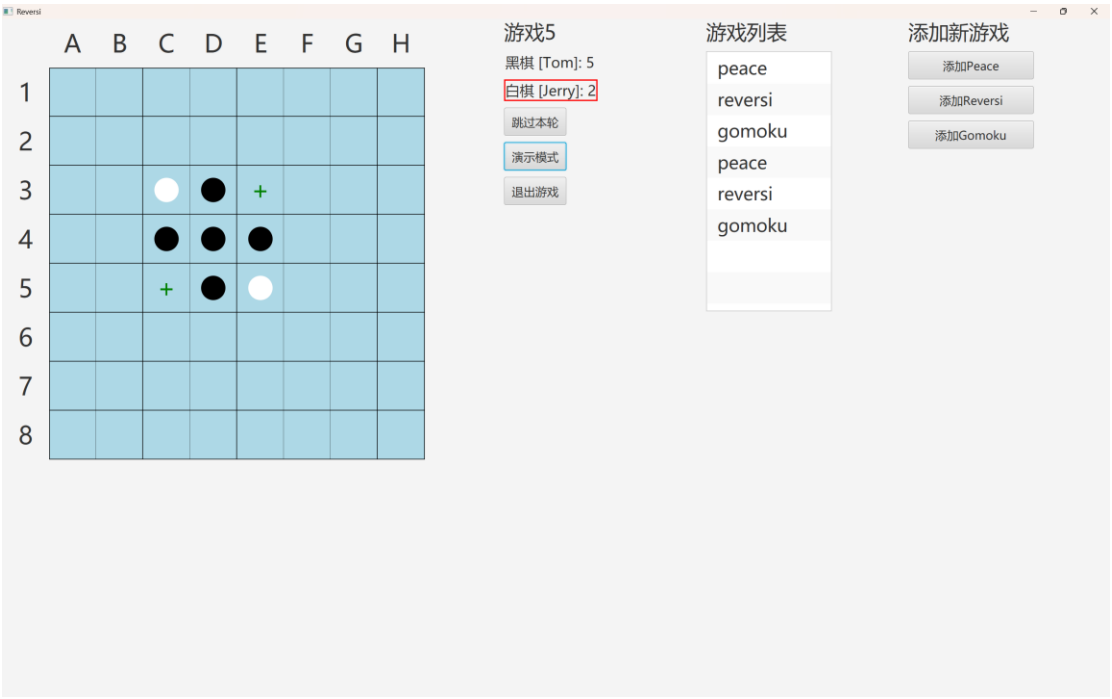
运行主类 Game，弹出游戏主界面。



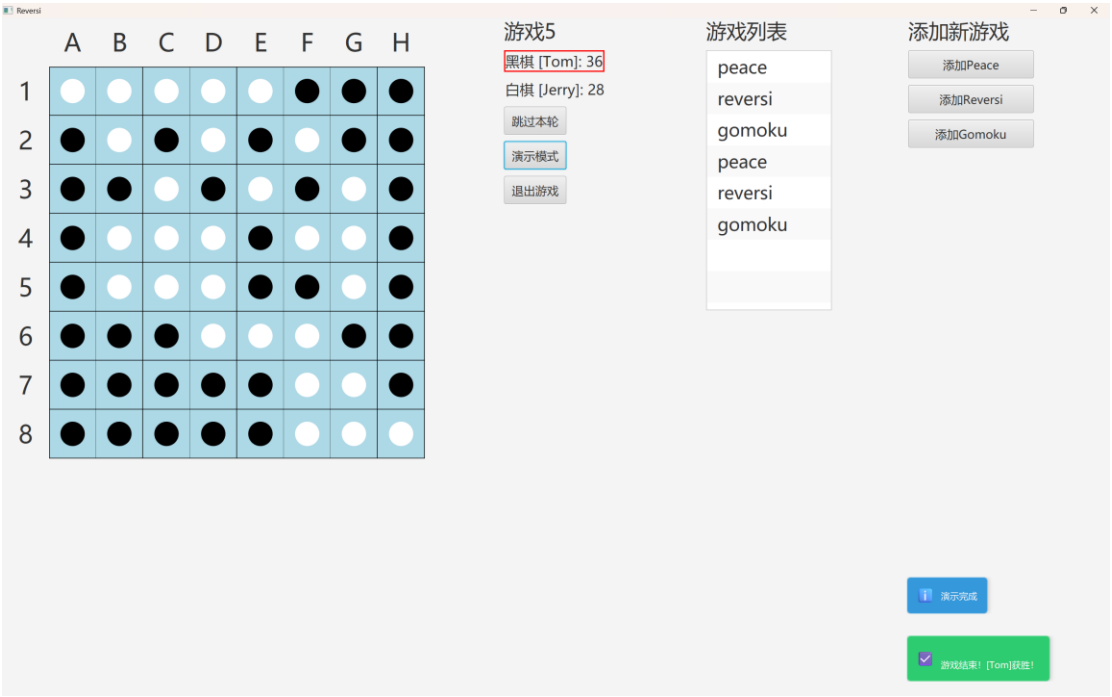
点击 ListView 切换对应棋盘。点击旁边的添加棋盘按钮添加游戏。



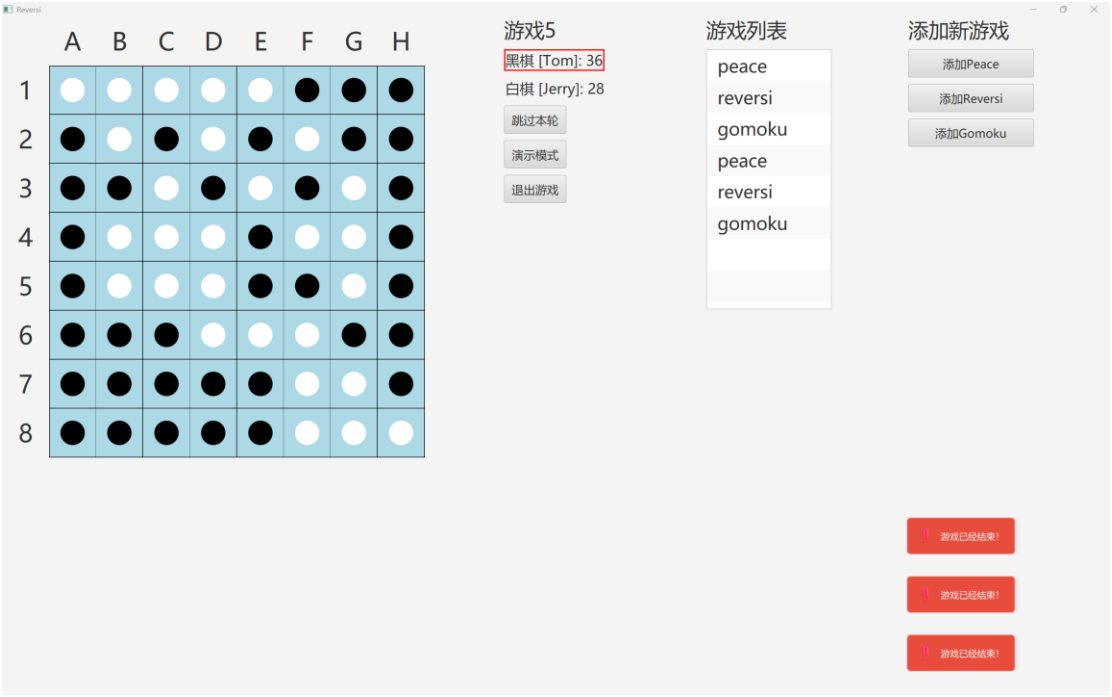
选择演示模式按钮，选择对应演示文件。



系统自动开始播放选定的文件。



播放完毕后，弹出游戏获胜和演示完成提示。


A B C D E F G H
1
2
3
4
5
6
7
8
游戏5
黑棋 [Tom]: 36
白棋 [Jerry]: 28
跳过本轮
演示模式
退出游戏
游戏列表
peace
reversi
gomoku
peace
reversi
gomoku
添加新游戏
添加Peace
添加Reversi
添加Gomoku
游戏已经结束!
游戏已经结束!
游戏已经结束!

非法操作时的错误提示。