北京工业大学

作业报告

课程名称:	面向对象程序设计		
学期:	2019 - 2020 学年 第 2 学期		
学号:	18021006	姓名:	吴天鹤
任课教师:	周珺	完成日期:	2020 年 5 月 17 日
教师评语:			

目录

一．题目的名称及要求

1. 做一个 Java GUI 应用程序，功能是四子棋游戏，横着、竖着或斜着连起来够 4 个棋子就赢了

2. 基本功能：棋盘的大小要求有 7 x 6 和 8 x 7 两种，做个菜单，在菜单里选棋盘规格；有一个 Play 按钮，点此按钮开始游戏；有一个文本框用于输入玩家姓名；单人游戏，人走一步，计算机走一步，计算机随机走，结束时要显示出谁赢了；游戏时，玩家点击最上一行的某处，就往相应的列中放棋子，鼠标点其他行不起作用

3. 扩展功能：棋盘的大小可定制；有 2 种游戏模式，人-机，人-人；计算机有智能；或其他自己扩展的功能

二．完成的功能及特色

1. 程序功能的详细说明

本次四子棋大作业，我完成了如下功能：

1> 有默认初始棋盘规格 7 x 6 和 8 x 7 两种。同时可以在文本框内输入棋盘的长和宽，自定义棋盘的大小，在输入时有异常处理
2> 有一个 start 按钮，当我按下后开始游戏，按下后 start 按钮变为 restart，可以在设计模式后重新开始游戏
3> 有一个文本框输入玩家姓名
4> 单人游戏人机模式时有两种模式，分别为计算机简单 simple 模式和计算机困难 hard 模式，对应计算机智能的不同。在简单 simple 模式下计算机只会自己连成 4 个子。在困难 hard 难度下，计算机会拦截玩家，同时会自己连成 4 个子，玩家很难战胜计算机
5> 在人机对战时，不管是 simple 难度还是 hard 难度，都可以设置玩家的先后手，player first 和 computer first
6> 可以实现双人对战模式，即玩家和玩家对弈的模式
7> 结束时会显示出哪一方获胜
8> 游戏时只有点击最上面一行才会下棋，点击其余行不会下棋
9> 在点击最上面一行下棋时，有棋子下落的动画
10> 当鼠标移动时，移动到某一列，某一列就会高亮显示，方便玩家分辨清楚自己下到了哪一列上

2. 自己的特色

本次四子棋大作业，我自己的特色有：

1> 计算机有两种智能模式，一种是简单 simple 难度，还有一种是困难 hard 难度
2> 人机对战时，可以选择先后手，即电脑先下，还是玩家先下
3> 在下棋时，有棋子下落的效果动画
4> 鼠标移动到某一列，某一列就会高亮显示

3. 运行界面截图

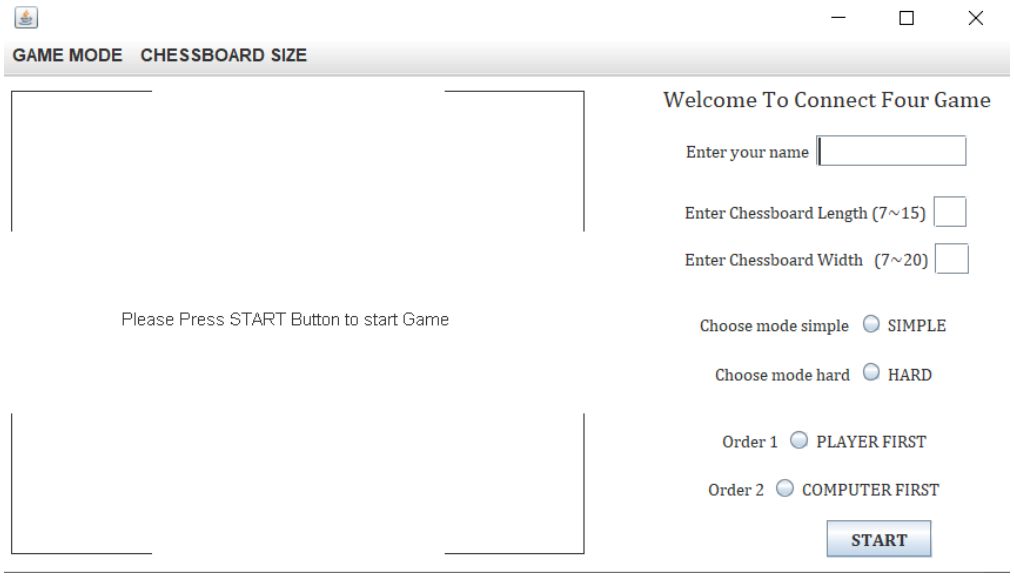

图 1 运行界面截图

三．设计与实现

1. 类之间的关系图

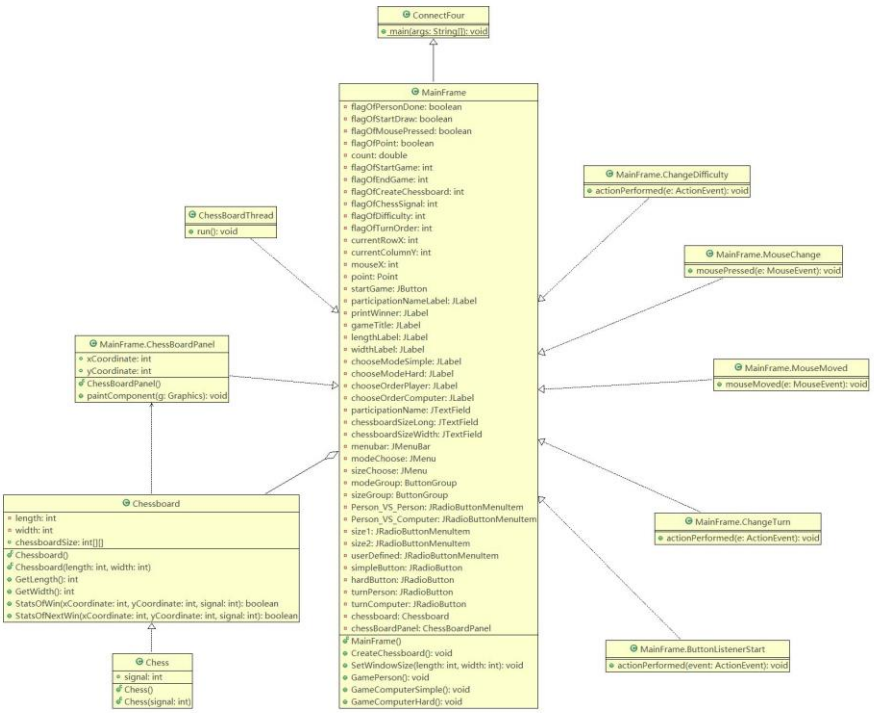

图 2 类之间的关系图截图

2. 类的说明

1> Chess 类

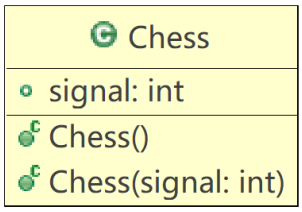

图 3 Chess 类 UML 图

Chess 类里有一个 int 型的变量 signal，为 public 公有属性。主要用来标记棋子的颜色，1 为 red，2 为 yellow，初始化时赋予棋子颜色。

2> Chessboard 类

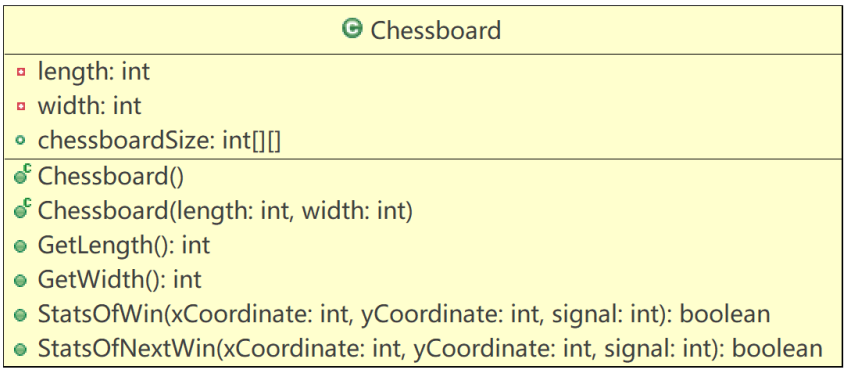

图 4 Chessboard 类 UML 图

Chessboard 类里有私有属性 int 型的 length 和 width，分别代表棋盘的长和宽，建立了一个二维数组，用来代表棋盘的每一个点。在棋盘中如果下了红色棋子，则被赋值为 1，如果下了黄色棋子则被赋值为 2，其余默认赋值为 0。

Chessboard 类里的方法有 GetLength 和 GetWidth 分别用来获取棋盘的长和宽，StatsOfWin 和 StatsOfNextWin 两个方法分别用来判断当前这步棋下完以后是否已经胜利，以及下一步棋下完是否会赢，主要应用在判断输赢和计算机 hard 难度下。

3> MouseChange 类

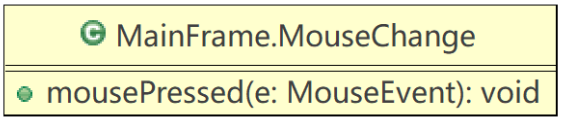

图 5 MouseChange 类 UML 图

MouseChange 类里主要实现了鼠标点击 mousePressed 方法，每点击一次，获取该点击处的坐标，并根据所选模式判断调用哪一种方法。

4> MouseMoved 类

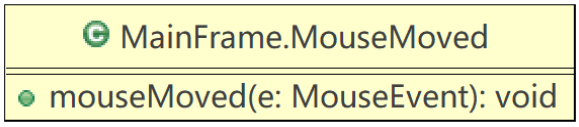

图 6 MouseMoved 类 UML 图

MouseMoved 类里面有 mouseMoved 方法，该方法主要实现移动鼠标的位置，获取鼠标移动位置处 X 坐标，并且不断地调用 repaint 方法来实现，移动到哪一列，哪一列高亮显示的功能。

5> ChangeTurn 类

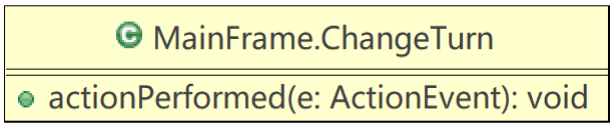

图 7 ChangeTurn 类 UML 图

ChangeTurn 类主要根据 User 选择的玩家先手还是电脑先手进行 flagOfTurnOrder 的赋值，为实现先后手的功能提供了帮助。

6> ChangeDifficulty 类

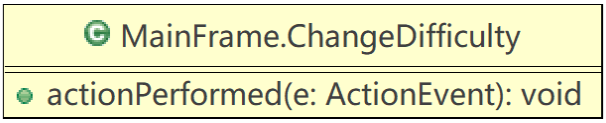

图 8 ChangeDifficulty 类 UML 图

ChangeDifficulty 类主要根据 User 选择的难度模式来决定在人机模式下 Computer 进行 Simple 难度还是 Hard 难度，为人机功能提供了帮助。

7> ButtonListenerStart 类

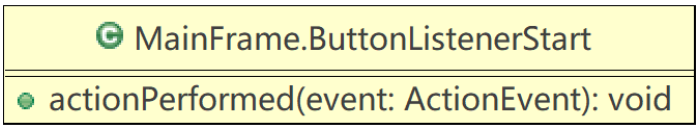

图 9 ButtonListenerStart 类 UML 图

ButtonListenerStart 类是实现开始游戏功能，在 actionPerformed 方法里为一些 flag 变量赋初值，并且调用 CreateChessboard 方法来创建棋盘，并根据玩家选择的先后手顺序调用一些方法来实现相应的功能。

8> ChessBoardPanel 类

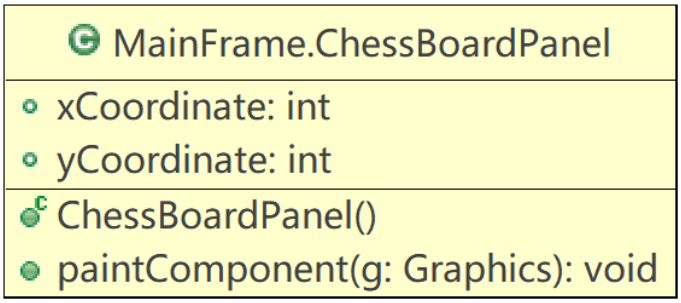

图 10 ChessBoardPanel 类 UML 图

ChessBoardPanel 类里主要是实现 paintComponent 方法，对棋盘不断重新绘制，以实现棋子下落，绘制棋子，绘制棋盘功能。其中定义了两个 int 类型变量 xCoordinate 和 yCoordinate 分别为棋盘的起始位置对应坐标(0, 0)。

9> ChessBoardThread 类

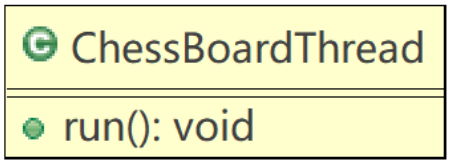

图 11 ChessBoardThread 类 UML 图

ChessBoardThread 类是实现了多线程的一个类，该类主要作用是绘制棋子下落的动画。run 方法里面有一个 while 循环不停地对 count 变量进行 count += 0.03 和 repaint 操作，以此来实现对棋子 y 坐标的不停更新，来实现棋子下落动画。

10> MainFrame 类

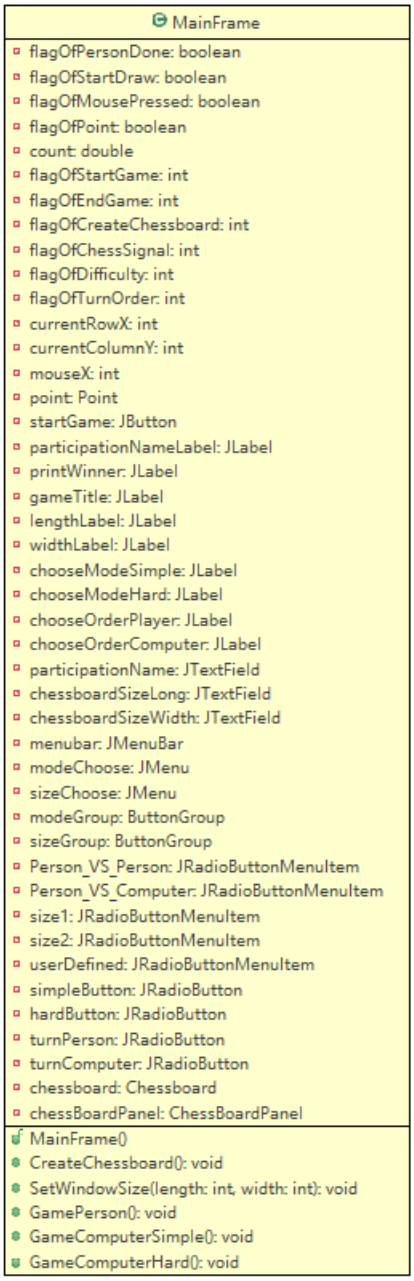

图 12 MainFrame 类 UML 图

上述大部分类都是在 MainFrame 类里实现的，MainFrame 类里面包含了所有的 flag，通过 flag 的变换以及更改来实现不同功能的切换以及实现。还有定义的点坐标，在不同的方法中调用它们来实现相应的功能。也定义了一些 boolean 变量，在面对是否 repaint 和对一些游戏方法调用的时候起到了关键作用。

界面里的 Label, TextField, Button 等元素也是定义在了 MainFrame 里面，并且在 MainFrame 构造方法里面实现元素的定义。并且在该构造方法内实现了菜单的定义与创建。MainFrame 构造方法里面对所有的变量均进行了初始化，同时在其中实现了 ChessBoardThread 类，意味着类创建以后，另一个线程就同时开始运行。

在 MainFrame 类里面还有 CreateChessboard, SetWindowSize, GamePerson, GameComputerSimple, GameComputerHard 等方法的实现，这些方法在一般都是在监听类里面进行调用，当事件发生被监听到，立刻调用方法。这些方法也是整个工程的核心方法，其中实现了游戏逻辑，实现了简单 Simple 模式，以及困难 hard 模式，也有双人对战的 GamePerson 方法，根据所选择的不同，调用不同的方法，来实现功能。

MainFrame 类是该工程的核心类，所有其余的类都是在其中进行实现和调用的，在该类中创建了棋盘类的对象，也创建了棋子类的对象，经过不同的 flag 选择，以及事件监听检测，来调用不同的方法，充分体现了面向对象的思维，还有多线程的思想，以及 repaint 刷新时利用人眼视觉暂留来实现动画的方法。各个类相互协调，相互调用，最终实现了四子棋的功能。

3. 主要方法的说明

1> CreateChessboard

该方法的主要目的是创建一个棋盘，根据 User 选择的模式，是 7 x 6 还是 8 x 7 或是自定义，来创建一个 Chessboard 对象

2> SetWindowSize

该方法的主要目的是在 User 选择自定义棋盘大小的时候，来改变窗口大小以适应棋盘大小。

3> paintComponent

该方法是对棋盘进行绘图，也是重要的方法之一。在开始游戏的时候，先绘制矩形，再利用两个 for 循环绘制白色的圆，每当我下一步棋时，该方法会判断该步棋对应的 signal 是多少，对应 1 就在该位置处画红色的圆，对应 2 就在该位置处画黄色的圆，0 就画白色的圆，意味着该处没有落子。同时再加入一个判定，当滑到该位置处是我最新落子的坐标位置，那么利用另一个线程里的 count，进行棋子下落时的绘画。同时还要判定我鼠标移动的 X 坐标的位置，在该位置处绘画透明度高的矩形，实现高亮功能。在游戏结束时，即 flagOfEndGame 为 1 时如果此时是黄色方玩家胜利，则绘制黄色背景，显现出胜利信息，如果红色方胜利，则绘制红色背景，显现出胜利信息。

4> GamePerson

该方法主要是实现人来下棋，判断下在了哪一个位置，将该位置数组对应值，赋值为 chess.signal，并且调用棋盘类里的方法 StatsOfWin 判断当前是否获胜。

5> GameComputerSimple

该方法是实现电脑简单模式下棋功能，设置随机从左至右下棋，并判断有没有获胜。

6> GameComputerHard

该方法是电脑困难模式下棋功能，该方法会先调用棋盘类里的方法 StatsOfNextWin 来检测下一步自己会不会赢，如果会赢的话，那么电脑自己就走这步，如果下一步不会赢，则判断下一步玩家会不会赢，如果会赢，则在相应地方进行拦截落子。如果对方也没有赢得地方，则在玩家棋子上方或左右进行落子，来减小玩家获胜概率，如果玩家上方或左右，已经有子，则连接自己颜色的棋子。

7> StatsOfWin

该方法是用来判断落子后是否已经胜利，经过分析一共上下左右左斜右斜 16 种情况进行判定即可。

8> StatsOfNextWin

该方法是用来判断下一步落子后是否已经胜利，经过分析一共上下左右左斜右斜 16 种情况进行判定即可。

4. 关键问题的解决

在做四子棋工程时，遇到以下几点问题：

1> 切换先后手：需要在调用方法时注意调用顺序，并且添加相应的 flag，在调用的时候作为条件判断是否调用某些方法

2> 计算机智能：hard 难度需要考虑的因素很多，因此需要明确优先级，判断何时去拦截，何时自己下，再进行方法的调用

3> 棋子下落动画：制作下落动画时需要明确多线程的思想，添加适当的 flag 进行条件判断，还有如何确保画到棋子想落在的位置就停止，需要加入 flagOfStartDraw 进行判定 count 的值，为 1 时停止 repaint。同时再写完这个新的线程以后，之前的 repaint 部分也要相应的修改，不然会出现很多其余的 bug，利用这个线程的 while 循环来不间断的 repaint 即可实现绘画

4> 判断胜负条件时：判断胜负条件时要充分考虑所有情况，不然会导致程序出现连成 4 子后，没法判定输赢的情况

5> 绘制图案：在绘制图案的时候，要调整窗口比例，需要进行参数的修改等等

四、测试

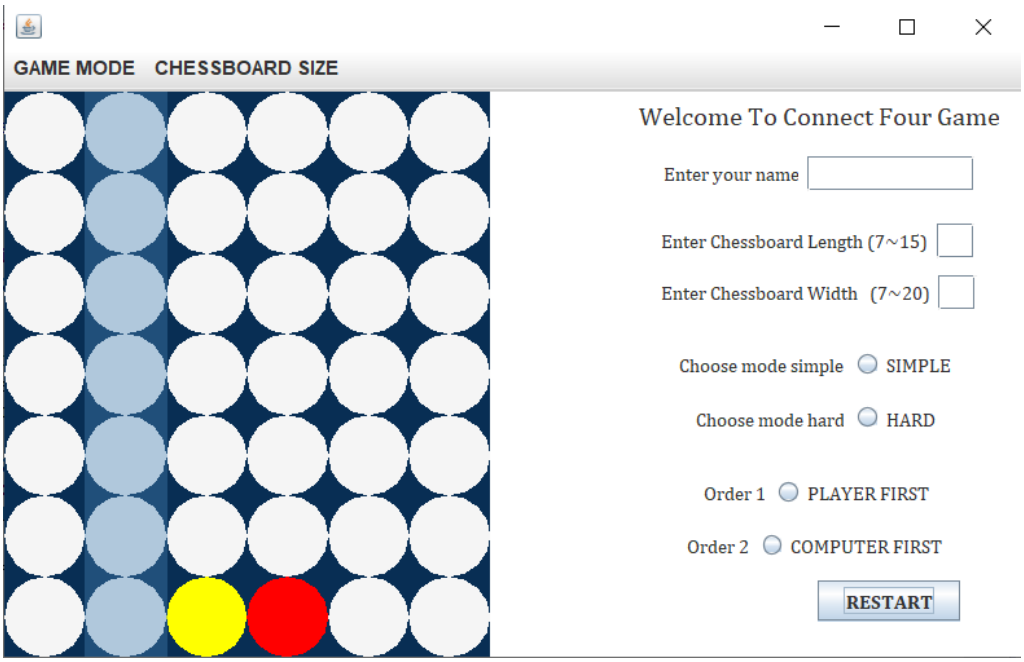

图 13 红子与黄子以及高亮显示截图

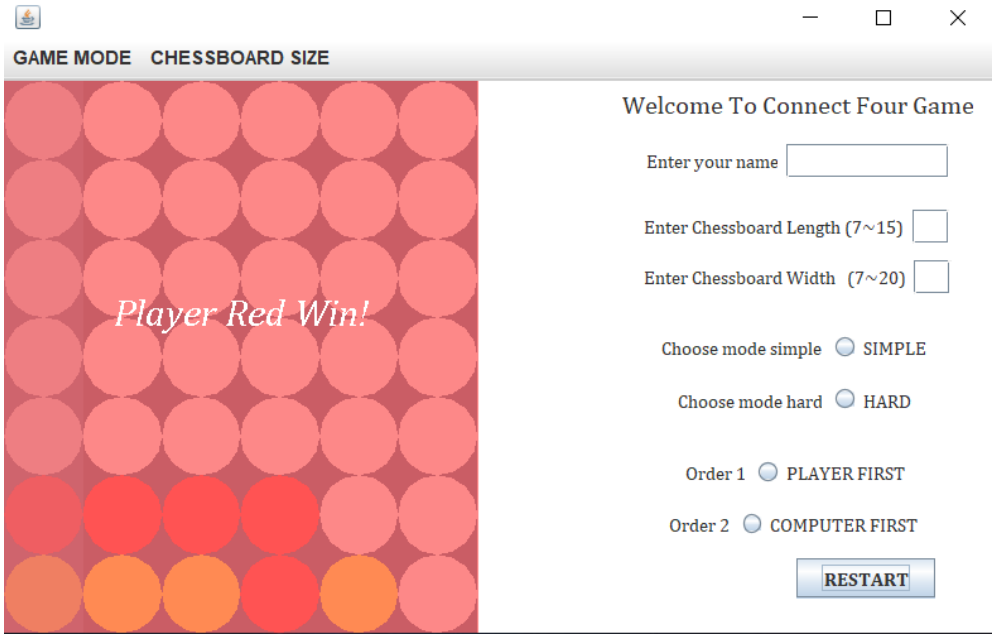

图 14 红方获胜截图

PS：由于人机模式先后手，以及困难简单模式截图后无法体现功能，还有棋子下落动画无法截图体现功能，因此这里不对上述功能进行截图了。

图 15 黄方获胜截图

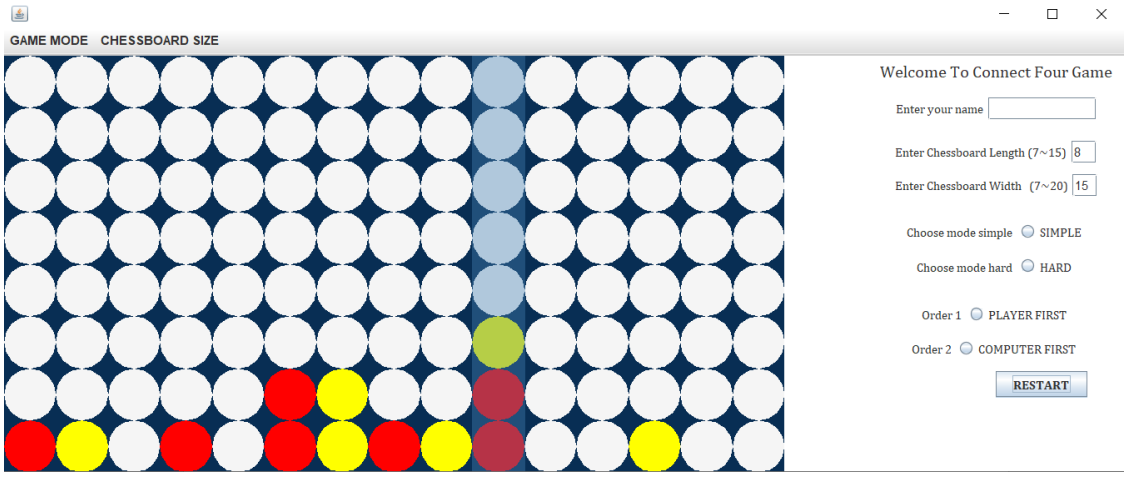

图 16 自定义棋盘截图

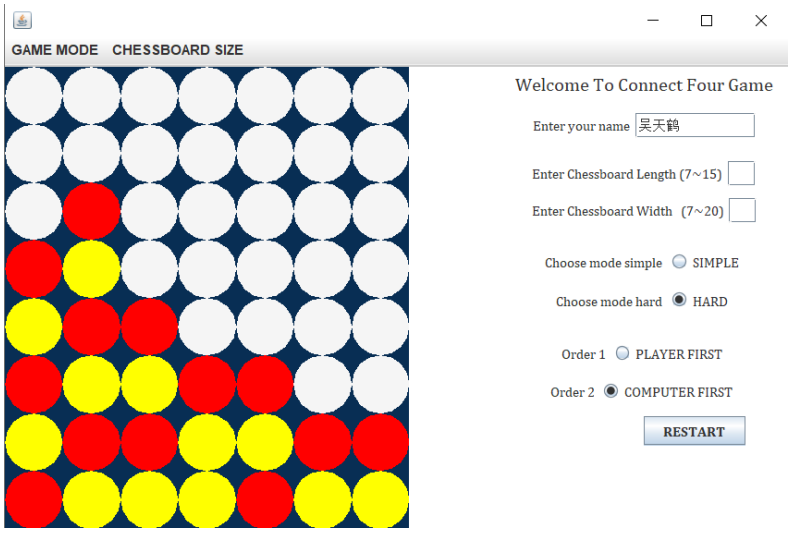

图 17 人机模式截图

五、总结

本次面向对象课程大作业四子棋开发，是我本人第一次做小开发。一开始的时候毫无头绪，后来自己先列了想实现的功能和以及相应的方法需求后才成功实现。在我看来这次的开发，又加深了面向对象的思维，这与现实中很相似。现实中有棋盘，因此我定义棋盘类，现实中有棋子，因此我定义棋子类。在别的地方创建棋盘对象，创建棋子对象，来进行相应方法的调用。

在做开发的时候，一定要写好大纲，千万不能做一步看一步，不然到后期要改的东西很多，并且自己会没有头绪。写好大纲能解决很多问题，让自己有思路，不会不知道自己做什么。

在写程序的时候会遇到很多 bug，这次做开发，让我沉静下了心，一步一步进行单步调试，找到哪里出现了问题。并添加相应的 flag。做图形界面部分要明确的思想就是多线程，以及这些是事件驱动型，它们一直在监听，与之前写的程序的思路和思维是有很大区别的。

感谢这次大作业，让我对计算机科学与技术这门专业有了更深的兴趣，以及对相应的思想，知识有了更深的理解。

附录

程序源代码清单：

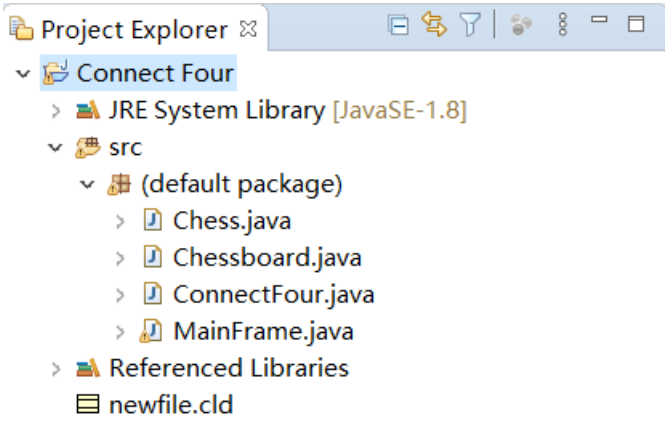

图 18 程序源代码清单截图