Java 大作业报告

1. 需求分析

本次 Java 大作业要求设计一个简易五子棋游戏，并至少实现本地单机对战等一种对战方式。最终本次大作业在完成基本功能的基础上实现了联机对战、复盘等功能。

2. 游戏方式

游戏界面如图 1 所示：

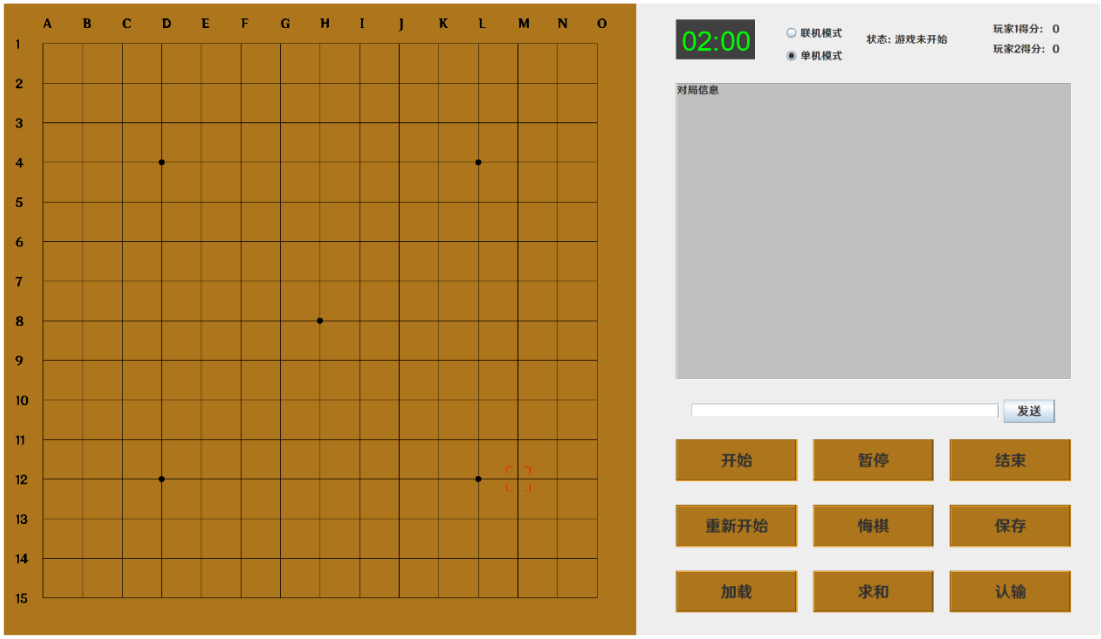

图 1 游戏界面

下面对游戏的各个功能进行逐一介绍。

2.1 单机和联机模式的切换

程序打开后默认是单机模式，在单机模式下，可以进行本地的双人对战。

如果要切换到联机模式，点击联机模式按钮，弹出如下对话框：

图 2 设置联机模式的对话框

在该对话框下输入对方服务器的 IP 地址、对方服务器监听的端口，并为本机也设置一个监听的端口。在有两台电脑的情况下，对方服务器地址即为对方电脑的 IP 地址，可以在 Windows 命令行模式下通过 ipconfig 命令查得，如果是本机运行两个该程序模拟联机情况，对方服务器需要填入“localhost”。

如果想切换到单机模式，直接点击单机模式按钮即可。

如果在进入联机模式后想要修改对方服务器的 IP 地址或端口，需要重新回到单机模式，然后再次进入联机模式即可。

2.2 开始游戏

在游戏尚未开始时，单机模式下，点击开始按钮比赛直接开始，程序将会为玩家 1 和玩家 2 随机分配黑棋和白旗。联机模式下，点击开始按钮后表明本机已经做好准备，此时将等待对方玩家。当对方玩家也点击了开始按钮后，比赛会正式开始。程序随机为本机和对方玩家分配玩家序号，并随机分配棋子颜色。

此外，在游戏已经开始的情况下，点击重新开始按钮可以重新开始一局比赛。单机模式下点击重新开始按钮后游戏直接重新开始，联机模式下则需要征得对方的同意。

每场比赛都是黑棋方先下。

2.3 悔棋、暂停和继续游戏

在比赛进行过程中单击暂停按钮，计时器将会暂停。同时暂停按钮变为继续按钮，单击继续按钮计时器继续计时，比赛继续。

在比赛进行过程中可以点击悔棋按钮进行悔棋，联机模式下得悔棋需要征得对方同意。

2.4 结束游戏

在单机模式下，点击结束、求和或认输按钮，都可以结束游戏；在联机模式下，只能通过求和或认输结束比赛。

2.5 保存和加载棋局

点击保存选项，程序将会将当前棋局和历史信息栏里的对局信息进行保存，保存的后缀名默认为.che。

在游戏没有开始的情况下，点击加载按钮，将会弹出选择文件的对话框，在该对话框中选择一个 che 文件，程序会弹出一个新的对话框，如下图所示：

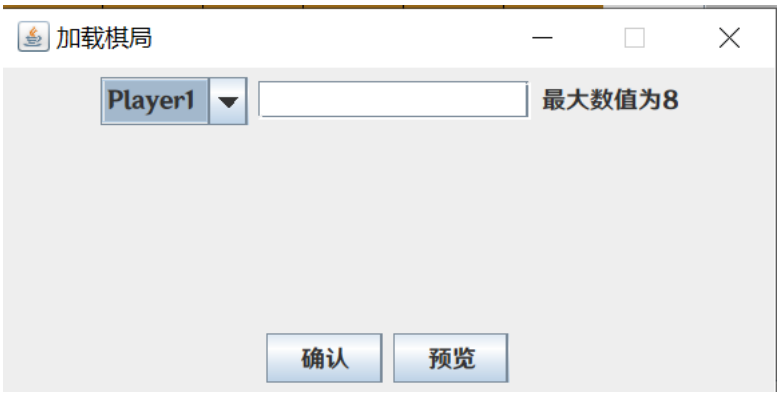

图 3 加载棋局的对话框

在该对话框中可以选择某一个玩家的某一次落子，点击预览按钮棋盘上将会显示出对应的棋局，点击确定按钮则会从选择的历史棋局开始比赛。如果在文本框中输入的不是自然数或输入的数字过大，程序都会弹出相应的对话框进行提示。

2.6 发送消息

在联机模式下，点击发送按钮可以将输入文本框中的消息发送给对方。

3. 类设计

本次大作业主要有三个大类。

3.1 MainWindow 类

该类用来画主窗口和实现主窗口的逻辑。该类的 UML 图如下：

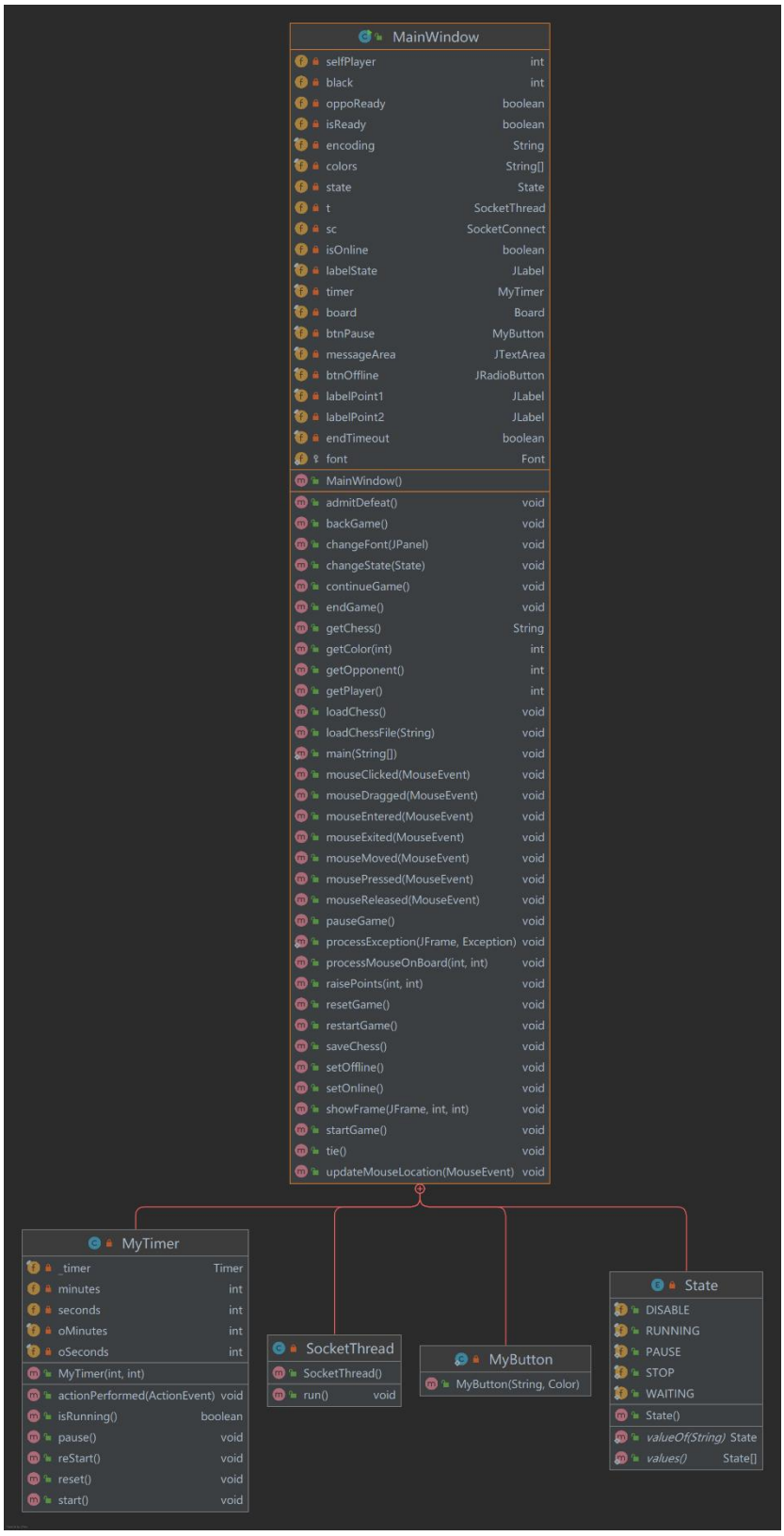

图 4 MainWindow 的 UML 图

3.1.1 主要字段

```
private int selfPlayer = 1;                    // 联机模式时的己方玩家
private int black = 1;                         // 黑棋方的玩家代号
private boolean oppoReady = false;             // 对方是否准备好
private boolean isReady = false;               // 自己是否准备好
private final String encoding = "UTF-8";       // 设置保存文件和网络传输的编码方式
private final String[] colors = {"黑", "白"};  // 棋子颜色
private State state = State.STOP;              // 当前状态
private SocketThread t;                        // 套接字线程
private SocketConnect sc;                      // 发送信息和接收信息
private boolean isOnline = false;              // 当前是否是在线模式

private final JLabel labelState;               // 状态标签
private final MyTimer timer;                   // 计时器
private final Board board;                     // 棋盘
private final MyButton btnPause;
private final JTextArea messageArea;           // 显示对局信息的区域
private final JRadioButton btnOffline;
private final JLabel labelPoint1;              // 玩家1得分
private final JLabel labelPoint2;              // 玩家2得分
private final boolean endTimeout;              // 超时时是否结束比赛
protected static Font font;                    // 各种组件的字体
```

3.1.2 主要方法

admitDefeat：该方法实现认输功能。

backGame：该方法实现悔棋功能。

changeState(State s)：该方法将当前游戏的状态更改为 State。

getColor(int i)：该方法得到玩家 i 所持有的棋子颜色。

getPlayer：得到当前下棋的玩家的序号。

loadChessFile(String s)：根据字符串 s 加载棋盘。

showFrame(JFrame f, int w, int h)：将窗口 f 以宽度 w、高度 h 显示在屏幕中央。

raisePonits(int i, int p)：将玩家 i 的分数增加 p 分。

processException(JFrame f, Exception e)：将异常 e 以对话框的形式显示。

3.1.3 内部类

State：

该变量是枚举类型，指示当前游戏的状态。其中 DISABLE 表示当前有二级页面打开，主窗口不能被操作；RUNNING 表示当前游戏处于运行状态；PAUSE 表示当前游戏处于暂停状态；STOP 表示当前游戏尚未进行。WAITING 表示当前处于联机状态且在等待对方回应。

SocketThread：

该类继承自 Thread，在联机状态下用于发送和接收消息。

MyTimer:

该类继承自 JLabel，实现定时器功能。

构造函数 MyTimer(int minutes, int seconds)表明创建定时 minutes 分钟 seconds 秒的定时器。方法 reset()可以重置定时器，start()开始计时，pause()暂停计时，reStart()在暂停状态下重新开始计时，isRunning()返回当前计时器是否在运行。

MyButton：

该类继承自 JButton，实现一个自定义的按钮。

3.2 Board 类

该类用来实现棋盘的可视化以及和棋盘相关的逻辑。UML 图如下：

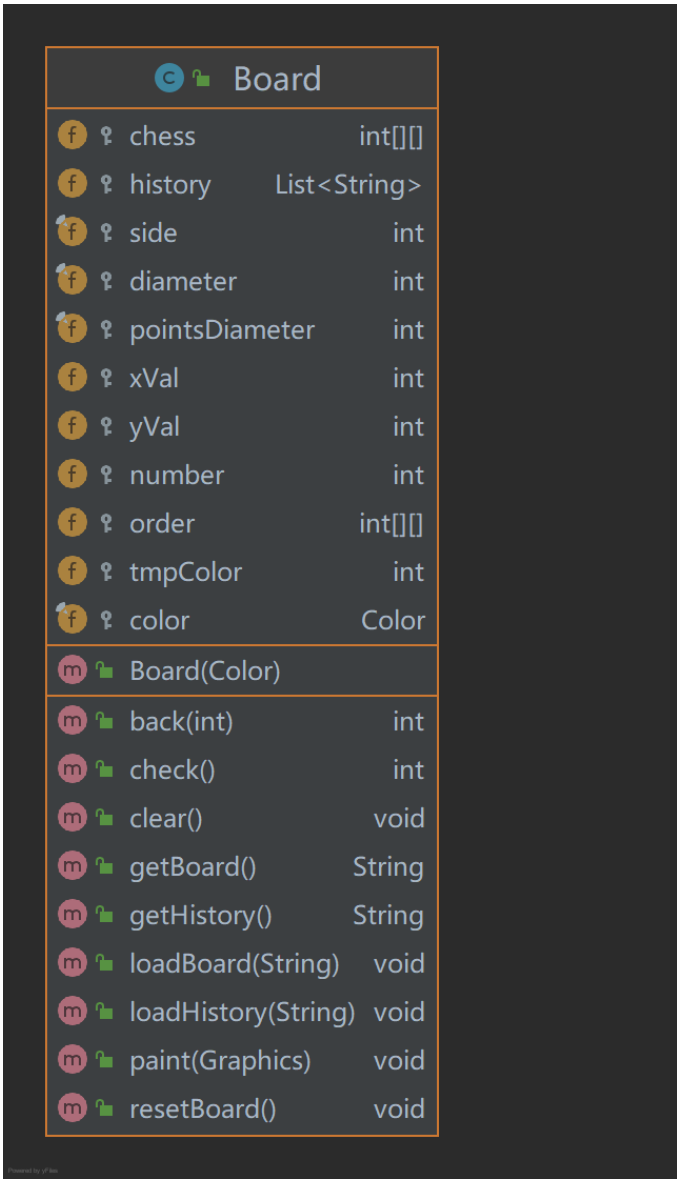

图 5 Board 的 UML 图

3.2.1 主要字段

```
protected int[][] chess = new int[15][15];            // 1：黑色，2：白色，0：没有棋子
protected List<String> history = new ArrayList<>();   // 对局顺序
protected final int side = 50;                        // 一个格子的边长
protected final int diameter = 30;                    // 棋子的直径
protected final int pointsDiameter = 8;               // 小圆点的直径
protected int xVal = -1, yVal = -1;                   // 当前鼠标在棋盘的落棋点的横坐标和纵坐标
protected int number = 0;                             // 当前棋子总数量
protected int[][] order = new int[15][15];            // 记录各个位置的棋子的下棋顺序
protected int tmpColor = 1;                           // 当前下棋方的棋子颜色
protected final Color color;                          // 棋盘颜色
```

3.2.2 主要方法

back(int i)：实现玩家 i 的悔棋，返回从当前棋局中删去的棋子数量。

check：检查当前棋盘是否有玩家获胜，如果没有返回 0，如果有返回获胜方的棋子颜色。

clear：重置各个变量但不重画棋盘

getBoard：获取当前的棋盘信息。

getHistory：获取当前对局顺序。

loadBoard(String s)：根据棋盘信息 s 加载棋盘。

loadHistory(String s)：根据历史对局信息 s 加载对局顺序。

resetBoard：重置棋盘。

3.3 SocketConnect 类

该类实现联机模式下的网络通信功能。该类的 UML 图如下：

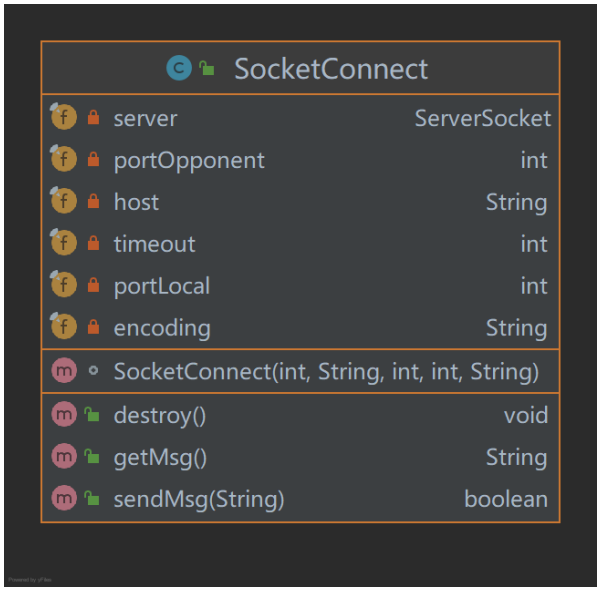

图 6 SocketConnect 的 UML 图

3.3.1 主要字段

```
private final ServerSocket server;      // 本地服务器套接字
private final int portOpponent;         // 对方服务器端口
private final String host;              // 对方服务器地址
private final int timeout;              // 超时时间
private final int portLocal;            // 本地服务器端口
private final String encoding;          // 编码方式
```

3.3.2 主要方法

destroy：关闭本地服务器。

getMsg：从对方服务器接收消息。

sendMsg(String s)：向对方服务器发送消息 s，发送失败返回 false，成功返回 true。

4. 附加功能实现

4.1 加载和保存文件的实现

4.1.2 文件保存格式

保存的棋盘文件由 4 个部分组成。第一部分是棋盘上各个位置的落子情况，0 代表未落子，1 代表落了黑子，2 代表落了白子，具体做法为依次遍历棋盘的每一行，得到长度为 225 的字符串。第二部分是棋盘的历史对局信息，按照落子顺序记录落子颜色、落子位置。第三部分是对局信息。第四部分记录 black、tmpColor 和 selfPlayer 变量，依次代表黑棋方的玩家代号、最后一次落子后下次落子的颜色和联机模式是的己方玩家代号。以上四个变量之间用符号“$”隔开。

为了将保存的文件和普通的 txt 文件区分开，保存的文件后缀设置为 che。保存的文件的一个示例如下：

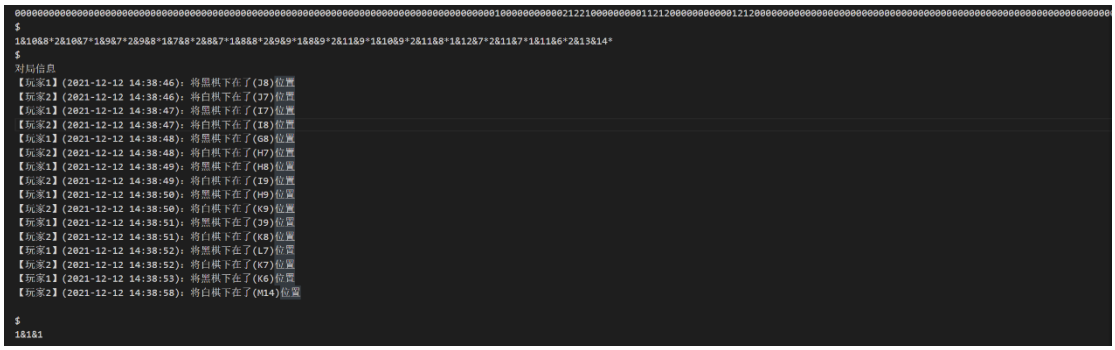

图 7 保存文件的一个示例

4.2 联机功能实现

4.2.1 网络架构

由于不可能架设一个专用的服务器负责客户端之间的通信，因此网络的结构上采用了双客户端-服务器结构，即每一方都作为服务器端和客户端。当一方想要发起通信时，该方会作为客户端去连接对方服务器，并将消息发送给对方服务器；当一方接收到消息时，该方是作为服务器。由于每一方的服务器都会持续监听事先设置好的端口，因此作为一个单独的线程而运行。

4.2.2 消息格式

消息格式采用消息类型加消息内容的方式。当服务器接收到消息时，会先将消息类型提取出来，确定消息类型，然后将消息内容送入处理该消息类型的程序块中进行处理。具体的消息类型有如下几种。

READY：对方做好开始游戏的准备。

PAUSE：对方暂停游戏。服务器接收到该消息后游戏会暂停。

CONTINUE：对方继续游戏。服务器接收到该消息后游戏会从暂停状态恢复运行状态。

LOCATION：落子的坐标消息。服务器接收到该消息表明对方已经落子，己方将根据落子坐标更新棋盘和游戏状态。

TIMEOUT：对方玩家超时。己方获胜，游戏结束。该信息只有当 MainWindow 中设置的 endTimeout 字段值（默认为 false）为 true 时才会发送。

BACK：对方提出悔棋。服务器接收到该消息后会弹出对话框让己方玩家确认是否同意对方

的悔棋，如果同意发送“AGREE_BACK”消息，如果不同意发送“REFUSE_BACK”消息。

AGREE_BACK：对方同意了己方提出的悔棋。

REFUSE_BACK：对方拒绝了己方提出的悔棋。

TIE：对方提出求和。服务器接收到该消息后会弹出对话框让己方玩家确认是否同意对方的求和，如果同意发送“AGREE_TIE”消息，如果不同意发送“REFUSE_TIE”消息。

AGREE_TIE：对方同意了己方提出的悔棋。

REFUSE_TIE：对方拒绝了己方提出的悔棋。

DEFEAT：对方认输。己方获胜，比赛结束。

LOAD：对方想要加载一份历史棋局。

TIE：对方提出重新开始。服务器接收到该消息后会弹出对话框让己方玩家确认是否同意重新开始，如果同意发送“AGREE_RESTART”消息，如果不同意发送“REFUSE_RESTART”消息。

AGREE_ RESTART：对方同意了己方提出的重新开始请求。

REFUSE_ RESTART：对方拒绝了己方提出的重新开始请求。

MESSAGE：对方的聊天消息。

CLOSE：关闭联机模式的消息。当己方玩家在联机模式下点击单机模式按钮时，己方向己方服务器发送 CLOSE 消息，使得己方服务器接收消息的线程进入运行状态（之前由于等待对方连接而处于阻塞状态），接着关闭服务器，并关闭该线程。