

Java 网络五子棋程序设计论文

软件工程 罗纶 2012666

一、程序简介

本程序以单机版五子棋为基础，利用 Java 语言实现了一个多功能的 Client/Socket 模式的网络对战游戏。其包含的内容如下：

- 本地对战、人机对战、网络对战三大模块
- 登陆系统、账号的注册注销
- 复盘、悔棋、重新开始、保存对局、读取对局
- 聊天、对局信息记录
- 人性化的弹窗提醒与消息提示

二、程序设计

明确了需求后就要选择合适的技术栈并探索实现思路：

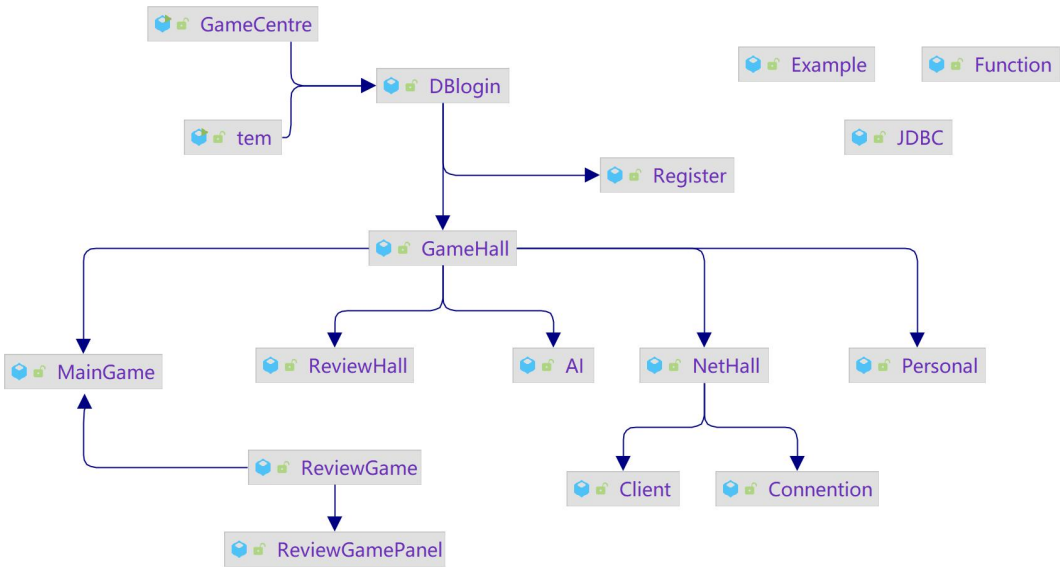
- (1) 主体界面选择使用 Java GUI 库中的 **Swing** 图形化工具。
- (2) 登陆系统与账号的注册注销使用 **JDBC** 完成与数据库的信息交互。

(3) 经过分析发现，使用集合中的链表就可以完美实现信息的存储并有效保证数据的顺序与完整。总结来讲：一局棋的本质就是一个以**位置类 (Point 类)**为节点类型的**链表 (LinkedList<Point>)**。在此基础上，复盘、悔棋、重新开始、保存对局等功能的实现就有了方向。复盘就是将此局的链表再可视化遍历一次；悔棋是退出链表尾节点；重新开始指清空链表；保存对局就是将该局的位置链表序列化，以“Save”加时间戳为名称，以**【.out】**为文件后缀 存储于程序的同级目录下，并在数据库中将账号与文件名达成匹配。读取时只读取与当前账号匹配的**【.out】**文件，将内容**反序列化**成链表对象后就可以进行可视化操作了。

(4) 人机对战、本地对战下的聊天更多是自言自语，网络对战状况下的聊天才是真正意义上的聊天。网络模式下，程序既可以作为主机等待他人接入，也可以作为客户端接入他人主机。主机声明 `ServerSocket` 对象、客户端声明 `Socket` 对象。网络中要维护**两条线程**：一条用于游戏中胜负判定、悔棋、保存等普通功能的响应，另一条则用于与对方端的信息实时交互。聊天就是交互的内容之一，另外交互的信息还有：每一步的位置信息，悔棋、重新开始等特殊信息。

(5) 大量的信息提示是改善玩家体验的重要因素，所以在适当的时候加入信息提示与信息反馈是重要的，比如加入房间失败后的失败提醒，保存对局成功后的成功提醒等都可以使用 `JOptionPane` 来实现。

三、模块介绍



以上为各类的一个大致流程图。

`GameCentre` 和 `tem` 是两个实例类，用于声明两个实例，运行整个程序。

`DBlogin` 类是整个程序的**起点**。`DBlogin` 类是用户登录界面类（图 3.1），界面内点击下方注册按钮则会进入 `Register` 类的注册界面（图 3.2），正确输入账号密码后会进入 `GameHall` 类所展示的游戏大厅中（图 3.3）。游戏大厅包括五个模块：本地对战界面（`MainGame` 类，图 3.5）、历史对局界面（`ReviewHall`，图 3.6）、人机对战界面（`AI`）、网络大厅界面（`NetHall` 类，图 3.7）和个人信

息界面（Person 类，图 3.4）。除历史对局界面下的 GUI 布局有所不同外，其他模式下的布局均沿用 MainGame 类。NetHall 类下还包含一个创建房间类（Connection）和一个客户端类（Client）。



图 3.1

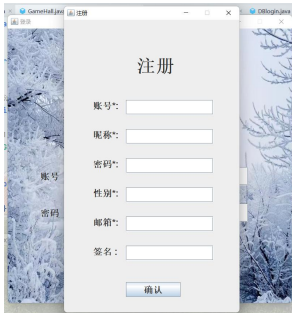


图 3.2



图 3.3



图 3.4

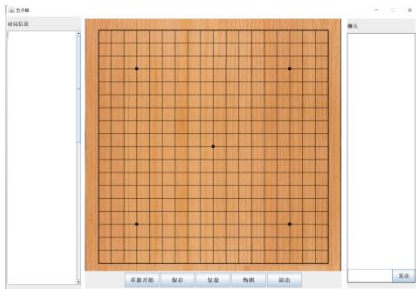


图 3.5

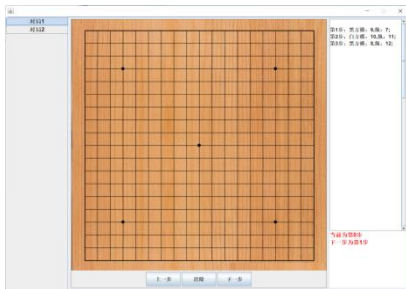


图 3.6



图 3.7

右上的三个类（Example、Function、JDBC）是需要应用于全局的三个类。Example 类用于实现**单例模式**，其中包含了几乎所有类的全局实例以及一些全局应用的参数，其属性均为**静态**。Function 类中是重要方法的集合，比如五子棋胜负的判定与各类的刷新等。JDBC 类封装的是所有与数据库连接有关的方法。包括注册注销，序列化保存等。

五子棋游戏的实现逻辑是：依照棋盘图片，设置一个 19x19 的二维数组，初始为 0。设置**鼠标点击事件**，计算距点击点最近的坐标，并将此对应的二维数组置为相应棋色（置 1 为黑，置 2 为白），每次点击后按二维数组的信息重绘棋盘，分别在相应点处绘制直径为 40 像素的圆。在点击事件发生后还需进行的操作有：判定是否有五个相连同色子，以及将该点击坐标加入链表便于以后的悔棋与保存操作。

下面将通过以下思路介绍实现过程：注册注销、基本功能实现、网络通讯、

AI 实现。

(1) 注册注销

注册注销操作主要依赖于数据库，因此需要使用 JDBC 相关技术。

```
JDBC() throws ClassNotFoundException, SQLException {  
  
    Class.forName( className: "com.mysql.cj.jdbc.Driver");  
    conn= DriverManager.getConnection  
        ( url: "jdbc:mysql://localhost:3306/game_player",  
          user: "root", password: "123456.");  
}
```

图 3.8

如上图连接数据库。注册过程中需要玩家自主填入“账号”“昵称”“密码”“性别”“邮箱”等信息。信息填写完毕后要经过 Example.jdbc.verify_account() 方法校验是否账号已被注册过，再经过 Example.jdbc.verify_nickname() 检验昵称是否唯一，其他的密码、邮箱都要经过正则表达式的规范化检验。经过上述检验并合格后将调用 JDBC 中的 register_player() 方法，而此函数再调用存储过程将该新用户信息插入数据库玩家表中，标志着新用户的成功注册。验证过程中每一步不合格情况都会以弹框形式提醒玩家修改（如图 3.9）。



图 3.9

注销功能实现与注册的大逻辑一致：**Java 方法调用存储过程**。在个人界面点击注销后会出现弹窗让您输入密码以验证是否是玩家本人，成功验证后即可注销。其背后是调用 JDBC 类中的 delete_game_record() 函数，再在数据库中执行相应 SQL 语句完成玩家信息的删除（tips: 玩家数据库中各表均依赖于玩家账号这一主码，所以一旦账号信息消失其他各表中的信息也将被销毁，不用担心数据

闲置问题)。登录实现则是调用 `log_out()` 方法：首先验证输入的账号是否存在然后再判断是否与密码匹配，完成检验后即可登录。

(2) 基本功能实现

1、悔棋

如上文所讲，我们将每一局落子信息作为一个链表存储，当你下完一步棋且对方还没下下一步之前悔棋才被允许，如果对方已落子将无法再悔。因此悔棋操作思路就是退出链表**尾节点**再将棋盘**重绘**。

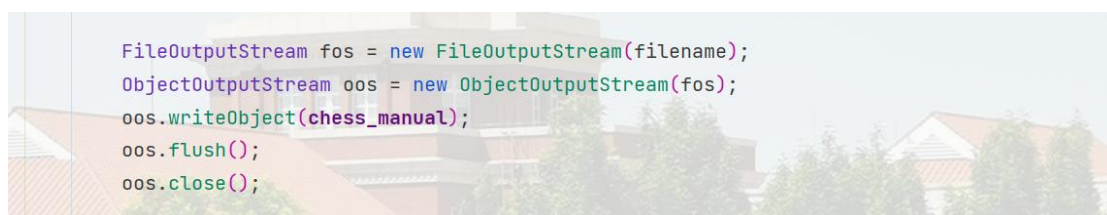
执行过程：首先判断“**位置链表**”是否为空，空的情况说明还没有下棋，弹出一个 `JOptionPane` 以示提醒；如不空且符合悔棋要求且不在网络对战的情况下将直接执行 `chess_manual.pollLast(); chessBoard.repaint();` 两行代码。同时回合信标摆回到下棋前的状态。

2、重新开始

在本地对战和人机对战的情况下（除网络对战外），会直接调用 `MainGame` 类内函数 `Restart()` 将棋盘数组重置为 0，重绘棋盘，回到黑棋先手状态。如果是网络对战则要通过 `ServerSocket` 与 `Socket` 的通讯征得对方同意后才能执行。

3、保存棋局

棋局的保存较复杂。由于一局棋的关键信息由一个链表承载，因此只需要保存此链表即可。利用链表的序列化，将 `Object` 对象转化为**字节码**，以“Save”加时间戳为名称保存在同级目录下，如图 3.10。



```
FileOutputStream fos = new FileOutputStream(filename);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(chess_manual);
oos.flush();
oos.close();
```

图 3.10

另外，为了方便下一次登陆时自动载。保存的过程中也调用 `JDBC` 类中的 `add_game_record()` 方法，把玩家账号与胜负、事件、序列化文件名等信息保存进一个表中。在下次进入“历史对局”模块时将调用 `JDBC` 的另一方法 `historycomp(String account)` 获取该玩家号下所有保存文件的文件名，并在

ReviewHall 类的 `init()` 方法中一一反序列化读取并生成新界面放入 `JTabbedPane` 中。

删除棋局是调用 `Example.jdbc.delete_game_record()` 方法删除数据库中的有关该棋局的记录，并使用 `File` 类中的 `delete` 方法删除该文件。

(3) 网络通讯

本程序的网络对战环节使用的是 TCP/IP 通讯以建立有连接的、可靠的、端对端的数据流的传输。在连接服务中，服务器进程声明一个 `ServerSocket` 用于接受客户 `Socket`，并绑定一个 `Socket` 来监听客户请求。

一个账号由于可以“创建房间”也可以“加入房间”，因此每个程序会有相应类：`Connection` 类用于服务端的创建、`Client` 类用于客户端的创建。此两类都继承自 `Thread` 类，作为主线程外的传输线程。

在点击“创建房间”后游戏主界面 `MainGame.f` 将展开，同时创建 `Connection` 类开启服务端线程实时监控服务端的通讯。客户端 `Socket` 的连入需要输入连入的 IP 地址，一旦连入，服务端的 `ServerSocket` 将 `accept()` 成功并开始正式通讯。

为了区别聊天内容、复盘、重新开始等信息与下棋位置信息，我们需要给传输的信息规定一些格式便于类型识别。如：除聊天信息外，其他信息会增加前缀“++++++”以示区分，位置信息的传输格式是“++++++ ‘横坐标’，‘纵坐标’；‘棋色’。”等。在接受信息时可配合字符串处理函数（`indexOf` 获取输入符号的第一个索引、`substring` 字符串的裁剪）准确快速的获取信息。

(4) AI 实现（有借鉴但修改）

AI 对战模块沿用的依然是主游戏界面。AI 类中封装的是 AI 的下棋逻辑。简单说就是通过递归为棋盘上的每一个空位进行局势判断并打分，遍历棋盘，在权重最大处落子。其中的“局势判断”取决于评分表的打分。评分表本质是一个哈希表，其中存储着各情况的权重，如图 3.11 和图 3.12。

如前文讲，棋盘数组中置 1 为黑棋，置 2 为白棋。因此“111”情况即黑子连成活三，“222”即白棋连成活三，在人工智能为黑棋的情况下：“111”权重

为 5000，而“222”权重为 8000，因此这时，AI 更偏向于围堵白棋而不是做活自己。

评分表的设计和 AI 主要代码都借鉴于网络，但为了接入原本的程序，我也对其进行了大量修改，例如接入对局程序、导入棋色、展示步数信息等。

```
//创建权值表
AI_sit_values.put("1", 10);
AI_sit_values.put("11", 100);
AI_sit_values.put("111", 5000);
AI_sit_values.put("1111", 8000);

AI_sit_values.put("12", 5);
AI_sit_values.put("112", 80);
AI_sit_values.put("1112", 3000);
AI_sit_values.put("11112", 10000);

AI_sit_values.put("21", 11);
AI_sit_values.put("211", 110);
AI_sit_values.put("2111", 1100);
AI_sit_values.put("21111", 11000);
```

图 3.11

```
AI_sit_values.put("2", 20);
AI_sit_values.put("22", 200);
AI_sit_values.put("222", 8000);
AI_sit_values.put("2222", 10000);

AI_sit_values.put("221", 100);
AI_sit_values.put("2221", 3000);
AI_sit_values.put("22221", 12000);
AI_sit_values.put("122", 5);
AI_sit_values.put("1222", 500);
AI_sit_values.put("12222", 10000);
```

图 3.12

此种 AI 模式是较为简单的模式，其棋艺高低主要依赖于评分表的人工设计。除此之外，还可以引入机器学习等更高级的技术和算法，如 AlphaGo 的“蒙特卡洛树”算法以及深度学习计算等。

四、遇到的重大困难与解决方法

（1）图片频闪与双缓冲技术

Swing 采用 **MVC 设计模式**，模型用于维护组件的状态，视图是组件的可视化表现，控制器用于控制各个事件发生组件做出怎么样的响应。模型发生改变，它通知所有依赖它的视图根据模型数据来更新自己。而更新时需要将 Canvas 画布进行清空再绘画，删除的瞬间就会出现闪烁，快速的重绘就引起了**频闪**。

而**双缓冲技术**就是再添加一张画布，两张图片交替前行，因此只需要展示已重新绘制好的图片就可以，避免了绘制过程的间隔，让观感更加舒适。

主要解决的代码就是重写 Canvas 中的 update 函数，如图 4.1。

```
public void update(Graphics g){
    // 双缓冲技术
    Image DbBuffer=createImage(getWidth(),getHeight());
    Graphics GraImage=DbBuffer.getGraphics();
    paint(GraImage);
    GraImage.dispose();
    g.drawImage(DbBuffer, x: 0, y: 0, observer: null);
}
```

图 4.1

(2) Socket 提前关闭与线程睡眠

在网络对战模式下，如果一方退出，那另一方也会退出并弹出一个弹窗：显示对方已经离线。但在调试过程中经常出现一方退出而另一方没有反应的问题。

首先讲解网络对战的程序思路，两条线程并行：**主线程**用于玩家本身的下棋、胜负的判定、局面的绘制等基本功能的实现，**另一条线程**就是专门用于通信。原程序的一方退出情况是，一方退出的点击事件发出后就将开启 Socket 通道的写数据功能，写入退出信息。对方端获取到退出信息指令后，同样退出己方界面并弹窗。问题就是一端点击关闭程序后，另一端没有反应，依然保持着连接状态。经过分析发现：主线程的退出事件发生后没有等到通信线程将**退出指令**发送就已经关闭了 Socket，导致另一端没有接收到退出信息指令。解决方法就是在退出事件发生后先让主线程睡眠一段时间，等待指令发出后再将其开启。



```
try {  
    Thread.sleep(100);  
} catch (InterruptedException ex) {  
    throw new RuntimeException(ex);  
}
```

图 4.2

五、优缺点总结

此五子棋游戏程序综合使用了**图形可视化**（Swing）、**网络通信**（TCP/IP）、**单例模式**（Example 类）、**多线程**（Connection 类和 Client 类）、**正则表达式**、**序列化反序列化**、**JDBC** 等技术，形成了一个较为完整的游戏程序结构。除了常见的单机模式外，还添加了人机对局和网络对局，而最重要的且最有趣的我认为**是历史对局部分**。以**对局信息链表**的序列化文件的形式进行存储，并通过数据库技术获取玩家账号下的所有序列化文件名，访问其文件并反序列化为链表对象后，再用 GUI 技术进行可视化**复盘**。其中涉及技术类型多，可以在技术层次上体现其重要性；从玩家层次上看，观看自己的历史精彩对局，既是“温故而知新”的过程，也是回顾自己高能时刻的体验，其重要性不言自明。另外程序中各种人性化的弹窗信息提示也应该算是一个亮点，精确的信息提示可以加快玩家对游戏的理解与操作，增加可玩性。

但程序背后的缺点还是非常明显的：代码比较臃肿，部分功能出现重复。原因是：设计之初没有考虑到使用 MVC 框架，只在五子棋单机游戏基础上进行功能叠加，导致设计没有全局观。如 MainGame 类和 ReviewGamePanel 类存在大量相似代码。如果早期全盘考虑的话应该可以避免，但现在程序已趋近完善不宜贸然修改。另外音效功能、更强大的 AI 功能、观战功能等都因时间原因或者一些其他原因还没有制作，以后有时间会加以完善。

六、游戏运行效果

(1) 主界面展示



图 6.1 登陆界面



图 6.2 游戏大厅

(2) 注册界面展示

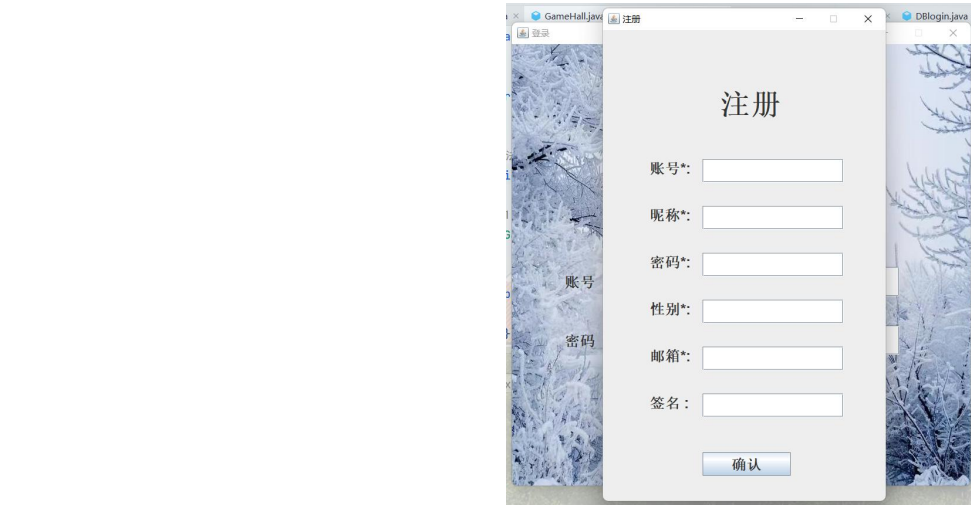


图 6.3 注册界面

(3) 本地对战界面展示

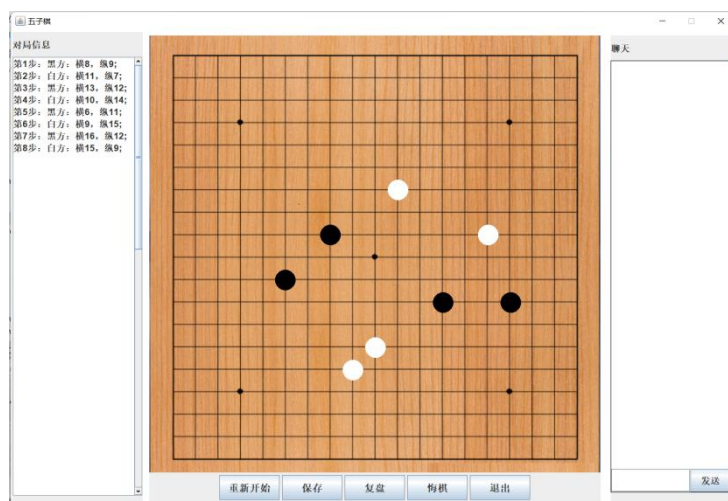


图 6.4 本地对战界面

(4) 人机对战界面展示

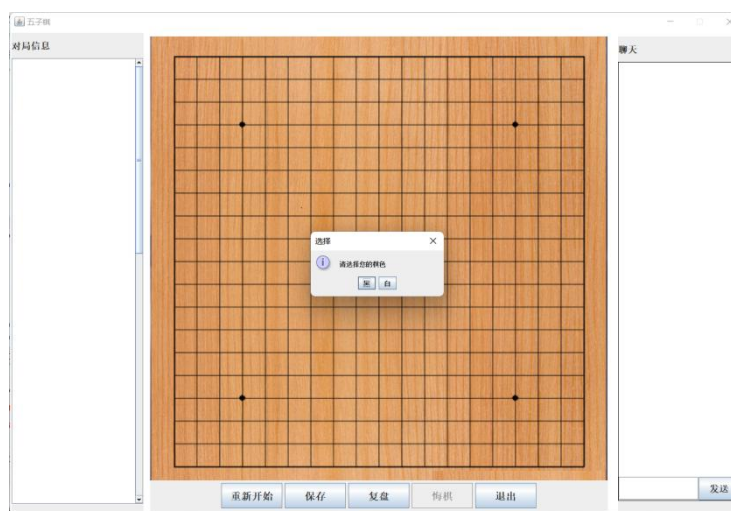


图 6.5 人机对战开局选色

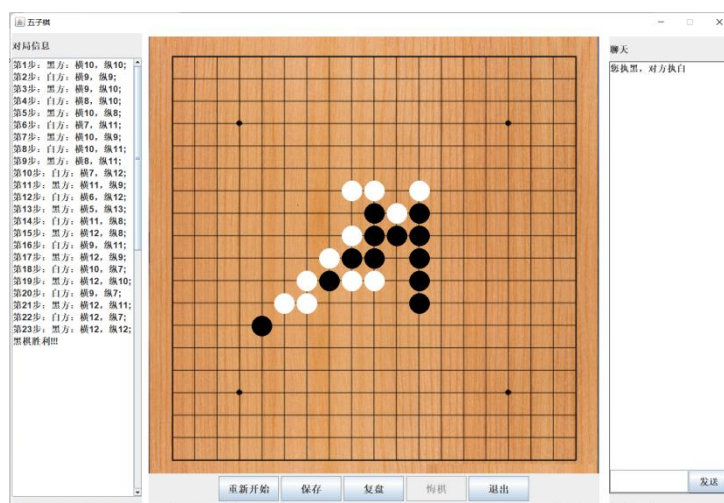


图 6.6 人机对战对局（AI 持白）

(5) 网络对战界面展示

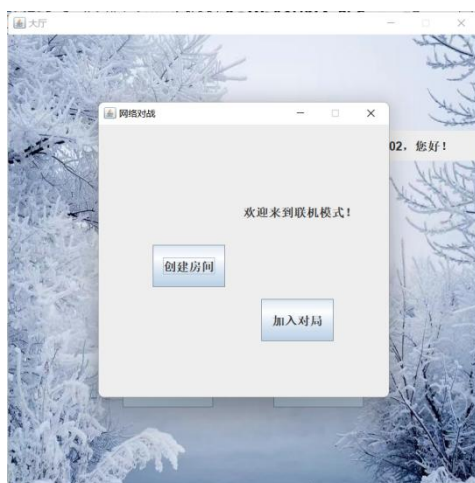


图 6.7 联机对战大厅

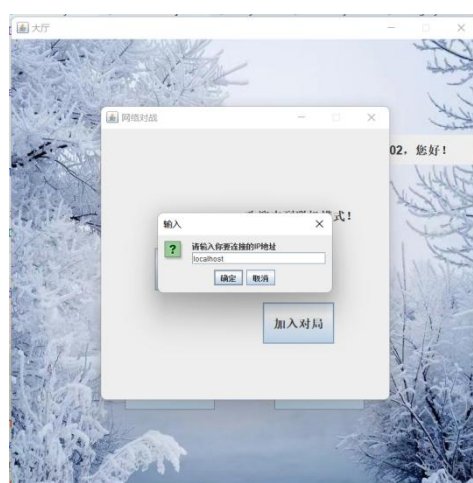


图 6.8 加入对局界面

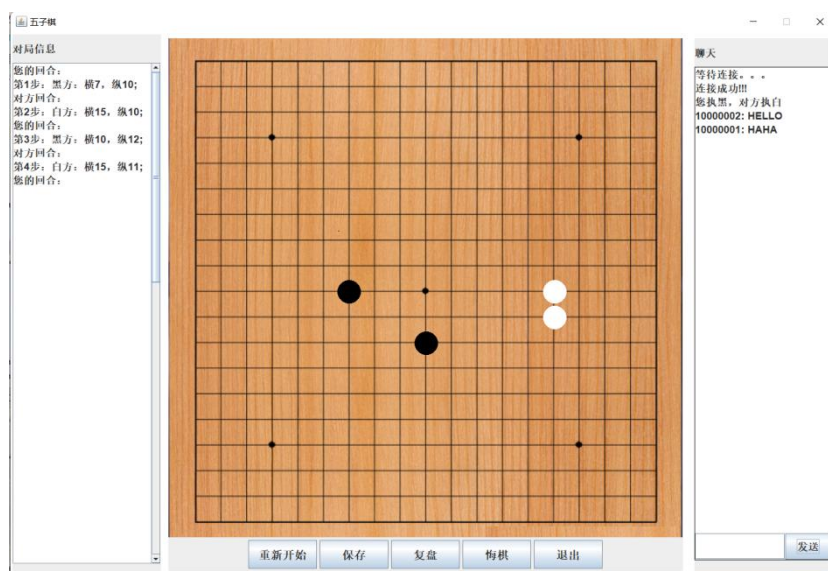


图 6.9 主机对战界面（主机持黑）

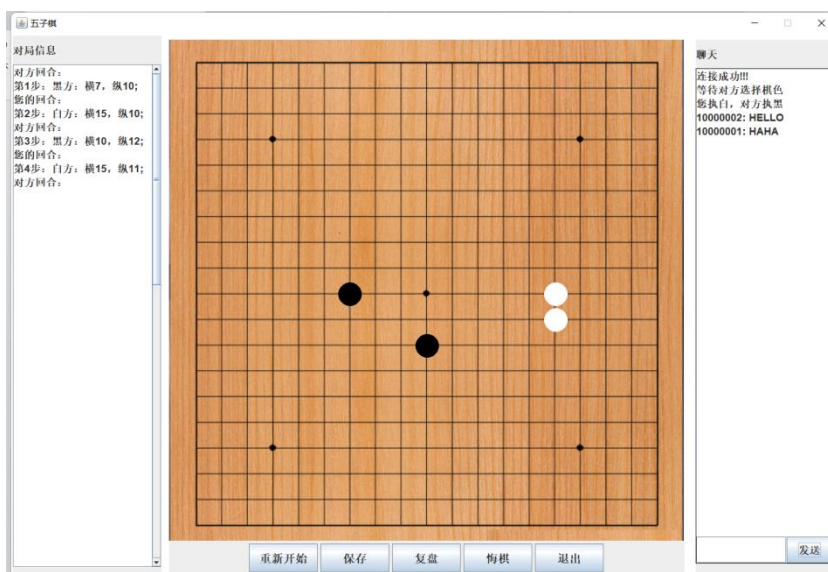


图 6.10 服务端界面（服务端持白）

（6）复盘界面展示

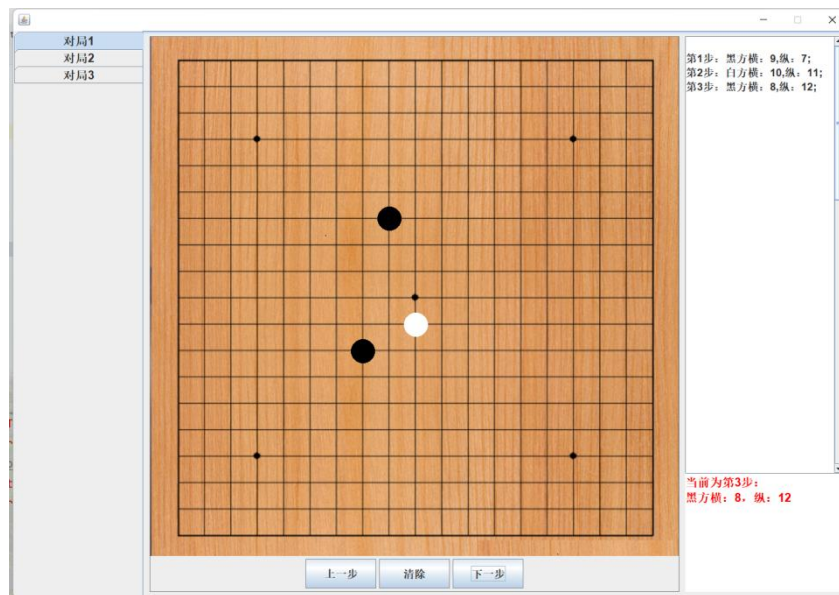


图 6.11 复盘界面

七、学习心得

经过一个学期的课程学习，我对 Java 语言的基础语法、包装类、集合框架、异常处理、IO 流、泛型、序列化、网络编程、多线程等方面的知识有了全新的理解、有了众多的收获。综合以上各知识点我才完成了此项目程序。在此需要感谢刘明铭老师的辛苦教学以及沈舒尹、朱静雯助教的辛勤付出。

一周多的大作业制作，让我发现我的实践能力比较薄弱。有时知道是使用某一方面的知识，但写起代码来就一头雾水，不得不耗费大量时间先去网络上查询一些课程或者查询一些文章来找到“砖”，经过自身的思考后才能将其垒成“墙”。所以在以后的课程学习中要注重积累实践经验，不仅要懂得理论，也要懂得代码操作，不然就是空中楼阁。

这一学期因为疫情的原因，课程时而线上时而线下。前期还能比较踏实地学习，但到了后期尤其是第二次线上课之后就已经十分的懈怠了，因此后面的两次课：网络编程和 JDBC 理解地都不是特别透彻，在制作大作业的过程中这也耗费了大量时间，希望能在以后的学习生活中极力避免。

最后再次感谢老师和助教的付出，感谢！

参考文献

【1】 棋盘图片素材来自网络

https://blog.csdn.net/weixin_46822367/article/details/121708862

【2】 双缓冲技术参考

https://blog.csdn.net/weixin_44552215/article/details/98748436

【3】 正则表达式参考

https://blog.csdn.net/weixin_43860260/article/details/91417485

【4】 AI 程序参考

<https://blog.csdn.net/thdgth/article/details/102837815>