浙江大学 计算机科学与技术学院

Java 程序设计课程报告

2024—2025 学年 秋冬学期

题目 五子棋网络对局程序

学号 xxxxxxxxxx

学生姓名 xxxxxx

所在专业 软件工程

所在班级 xxxxxxxx

目录

1 引言 **1**

1.1 设计目的 1

1.2 设计说明 1

2 总体设计 **1**

2.1 功能模块设计 1

2.2 流程图设计 3

3 详细设计 **6**

3.1 客户端设计 6

3.1.1 登录界面与功能的设计 6

3.1.2 注册界面与功能的设计 7

3.1.3 主界面与功能的设计 9

3.1.3.1 创建对局按钮 9

3.1.3.2 查看在线客户端 9

3.1.3.3 加入对局 10

3.1.3.4 查询历史对局按钮 10

3.1.3.5 退出按钮 11

3.1.4 五子棋棋盘界面与功能的设计 12

3.1.5 客户端接收信息线程的设计 13

3.2 服务端设计 15

3.2.1 与客户端交互模块的设计 15

3.2.2 数据库的设计 17

3.2.2.1 用户表设计 17

3.2.2.2 对局记录表设计 18

4 测试与运行 **18**

4.1 程序测试 18

4.2 程序运行 21

5 总结 **28**

1 引言

1.1 设计目的

为了巩固在 Java 应用技术课程中学到的面向对象设计方式、网络编程技术、GUI 应用设计技术和多线程编程技术，以及为了完成课程的最后程序设计作业，本人以五子棋为切入点，通过实现客户端与服务端交互，使得不同用户可以通过客户端程序进行五子棋在线对局。同时，也将数据库相关操作融合进服务端程序中，使得用户可以查询对局历史来进行复盘。

1.2 设计说明

本程序由本人独自开发设计，报告亦为本人独自撰写。以下为程序的运行环境、调用的库和开发工具的大致介绍。
程序设计的环境配置说明：

1. 使用的编程语言为 Java (OpenJDK v18.0.2)
2. 使用的 IDE 为 IntelliJ IDEA (v2024.2.4)
3. 使用的包管理工具为 Maven (v4.0.0)
4. 使用的数据库是 MySQL (v5.7.43)

程序设计中使用的主要库及其功能的说明：

1. 通过 Java 自带的网络库 (java.net) 中的 Socket 与 ServerSocket 类来实现服务端和客户端的通信。
2. 通过 Java 自带的多线程库 (java.lang.Thread) 实现服务端使用不同线程与不同客户端进行通信。
3. 通过 JavaFx (v18.0.2) 实现组件绘制、程序 GUI 显示和交互。

2 总体设计

2.1 功能模块设计

客户端程序设计达成的功能说明：

1. 用户的登录、注册和退出客户端功能
2. 用户查看其他在线用户功能
3. 用户主动创建五子棋对局并等待其他用户加入的功能
4. 用户加入其他用户的五子棋对局的功能
5. 用户和其他用户进行五子棋对局，并最终获取输赢结果的功能
6. 用户查看自己的历史对局记录以及详细的棋谱的功能

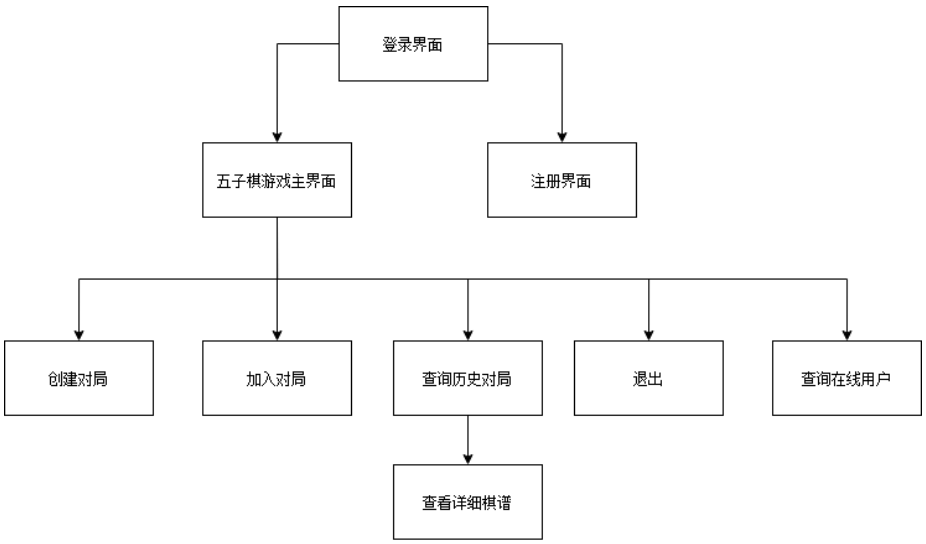

图 1: 客户端总体功能

服务端程序设计达成的功能说明：

1. 提供用户登录、注册服务
2. 提供用户创建对局服务
3. 提供用户加入对局服务
4. 提供用户查询历史对局服务
5. 提供用户查询对局棋谱服务
6. 提供用户查询在线用户服务

7. 提供对局的过程支持（包含放置棋子、回合转换和输赢判定）

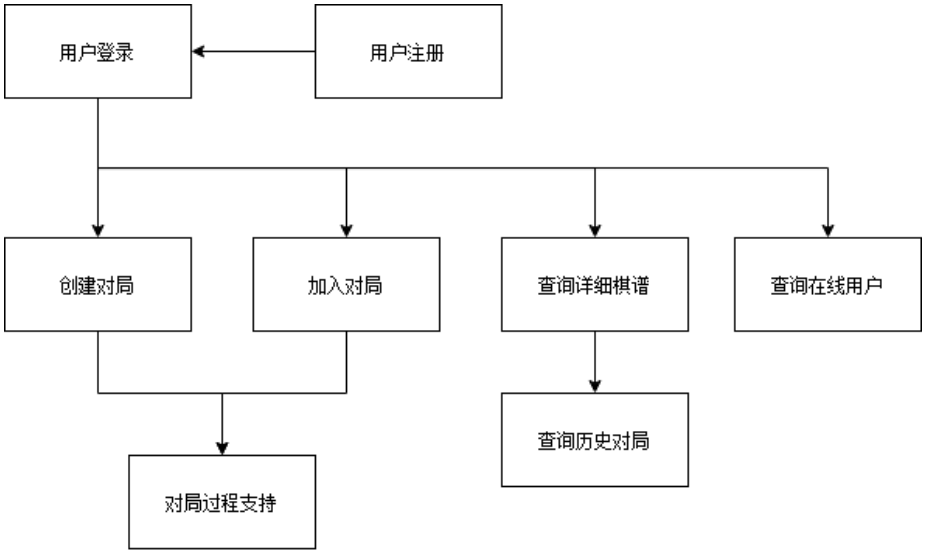

图 2: 服务端总体功能

2.2 流程图设计

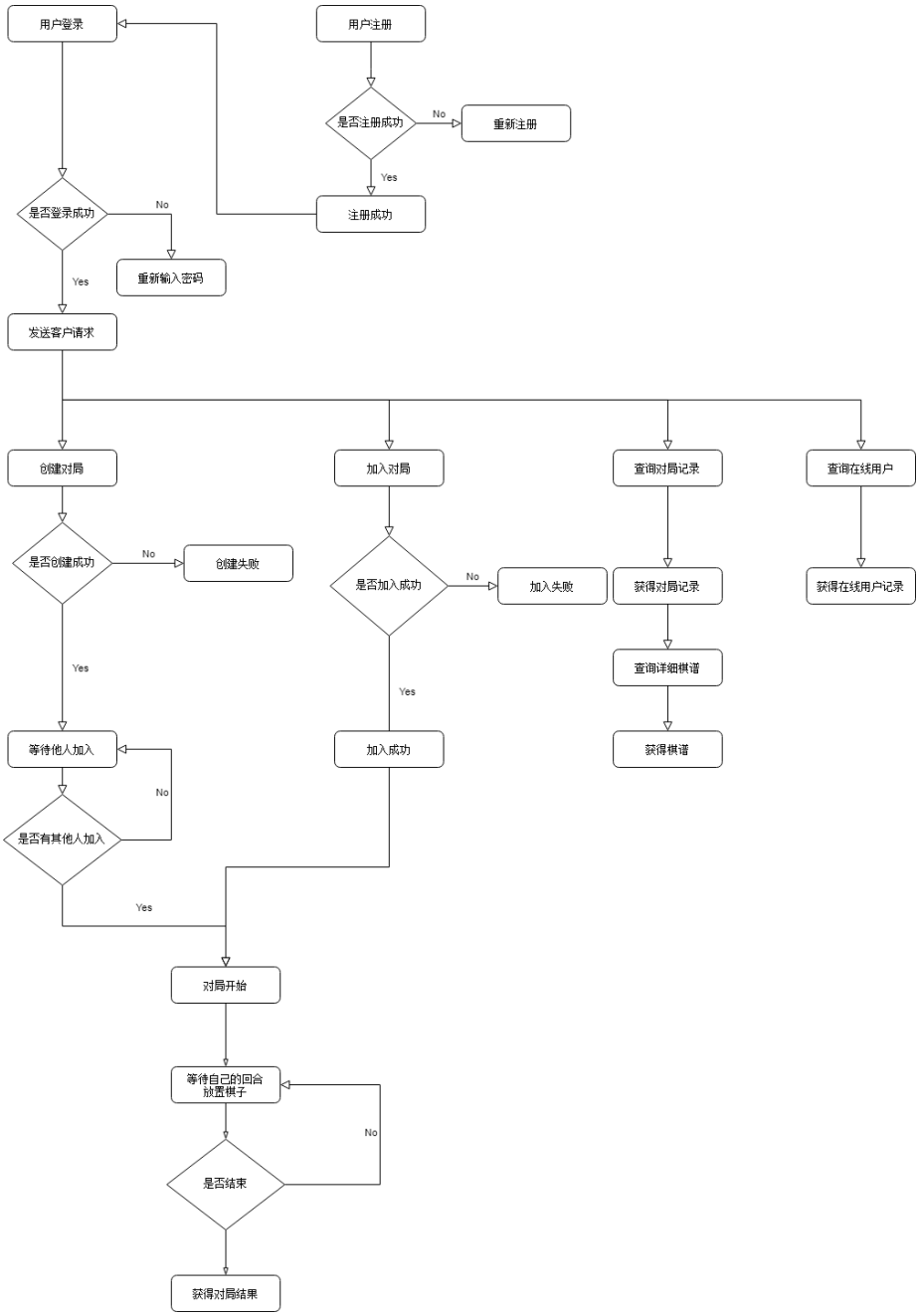
用户登录
用户注册
是否注册成功
No
重新注册
Yes
是否登录成功
No
注册成功
重新输入密码
Yes
发送客户请求
创建对局
加入对局
查询对局记录
查询在线用户
是否创建成功
No
创建失败
是否加入成功
No
加入失败
获得对局记录
获得在线用户记录
查询详细棋谱
Yes
Yes
等待他人加入
加入成功
获得棋谱
No
是否有其他人加入
Yes
对局开始
等待自己的回合
放置棋子
No
是否结束
获得对局结果

图 3: 客户端流程设计

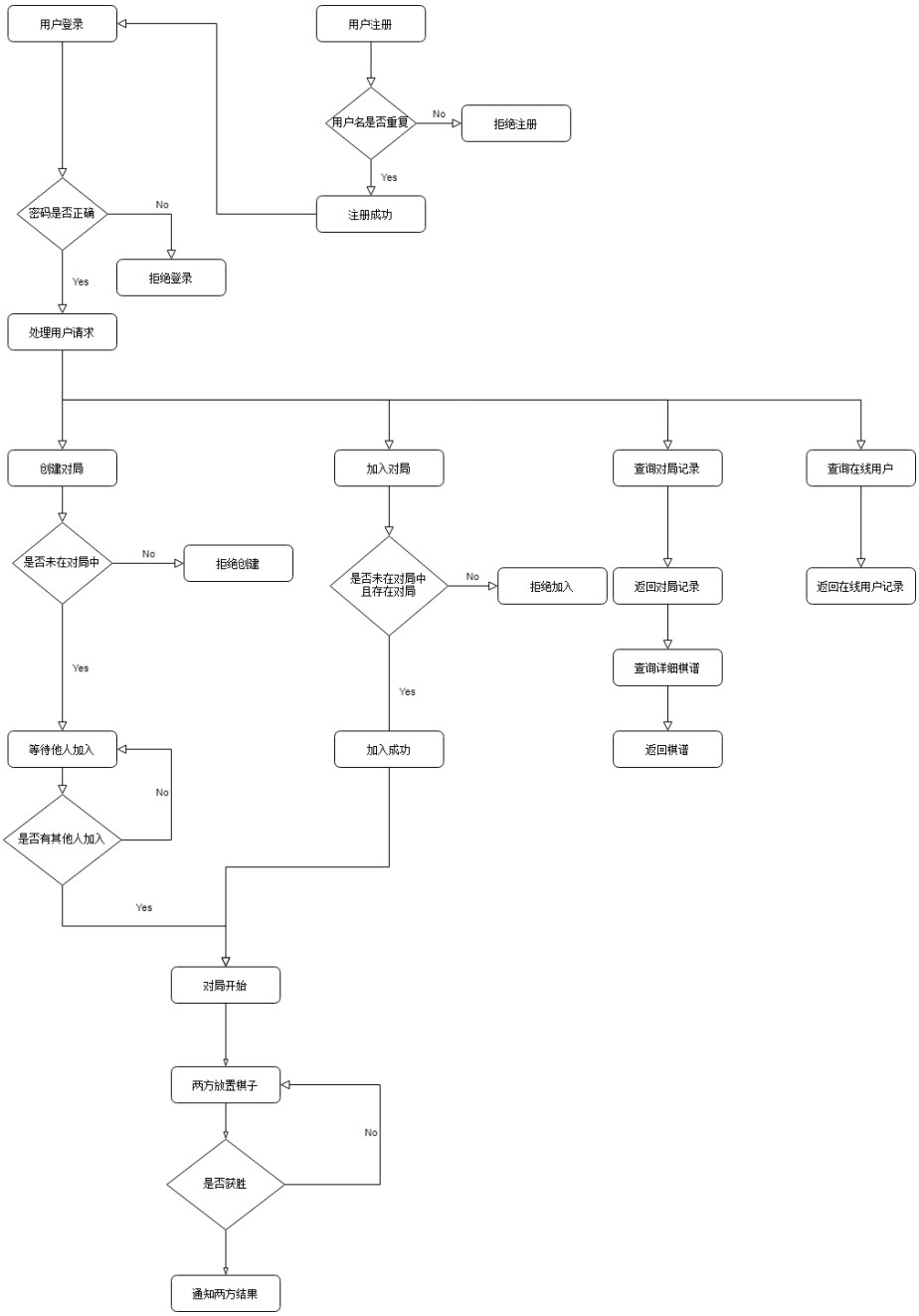
用户登录
用户注册
用户名是否重复
No
拒绝注册
Yes
密码是否正确
No
注册成功
拒绝登录
Yes
处理用户请求
创建对局
加入对局
查询对局记录
查询在线用户
是否未在对局中
No
拒绝创建
是否未在对局中
且存在对局
No
拒绝加入
返回对局记录
返回在线用户记录
查询详细棋谱
Yes
Yes
等待他人加入
加入成功
返回棋谱
No
是否有其他人加入
Yes
对局开始
两方放置棋子
No
是否获胜
通知两方结果

图 4: 服务端流程设计

3 详细设计

3.1 客户端设计

3.1.1 登录界面与功能的设计

登录界面主要采用了用户名输入框、密码输入框、登录按钮和注册按钮这 4 个控件。当点击登录按钮时，LoginHandler 会读取输入的用户名和密码，然后上传给服务端。在上传之前，还会检查是否已经登录，若已经登陆，就会弹出提示框提醒用户。如果服务端回应密码输入错误，也会弹出提示框提醒用户登录成功后，跳转到主界面。用户还可以通过注册按钮调转到注册界面进行注册。

登录控件代码框架如下：

```
public class LoginHandler {
    //用户名输入框
    @FXML
    private TextField usernameField;
    //密码输入框
    @FXML
    private PasswordField passwordField;
    //登录按钮绑定方法
    private void handleLogin() {}
    //注册按钮绑定方法
    private void handleRegister() {}
    //弹出提示框方法
    private void showAlert(Alert.AlertType alertType,
    String title, String message) {}
}
```

Listing 1: 登录界面控件

图 5: 登录界面

3.1.2 注册界面与功能的设计

注册界面录主要采用了用户名输入框、密码输入框注册和按钮这 3 个控件。当点击注册按钮时，RegisterHandler 会读取输入的用户名和密码，然后上传给服务端。如果服务器回应用户名重复，会弹出提示框提醒用户重新输入用户名，如果登录成功，会跳转回登录页面
注册控件代码框架如下：

```
public class RegisterHandler {
    @FXML
    private TextField usernameField;
    @FXML
    private PasswordField passwordField;
    //绑定注册按钮方法
    @FXML
    private void handleRegister() {}
    //弹出提示框方法
    private void showAlert(Alert.AlertType alertType,
    String title, String message) {}
}
```

Listing 2: 注册界面控件

图 6: 注册界面

3.1.3 主界面与功能的设计

主界面主要采用了 5 个按钮控件来提供不同的功能。

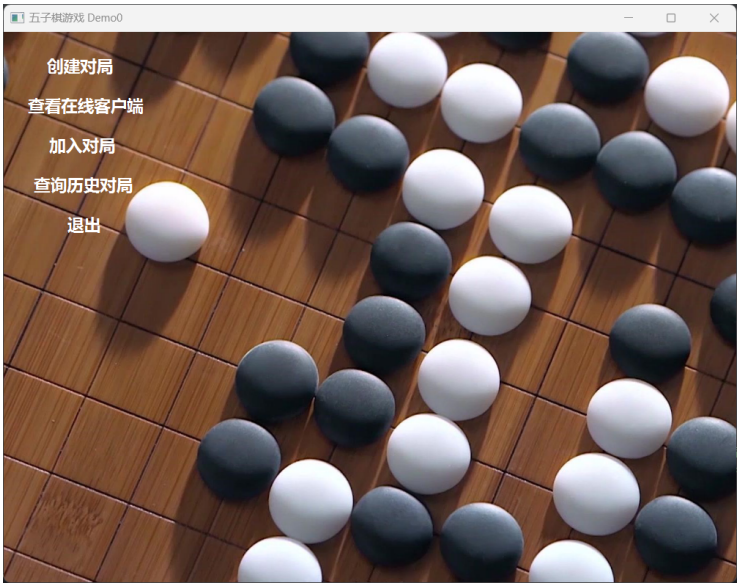

图 7: 主界面

3.1.3.1 创建对局按钮 当点击了创建对局按钮时，客户端向服务端发送创建对局请求，获取服务端许可后，创建对局页面。关于对局页面，将在五子棋棋盘界面与功能的设计中介绍。

3.1.3.2 查看在线客户端 当点击了查看在线客户端页面后，客户端向服务端发送查看在线客户端请求，获取服务端结果并展示。

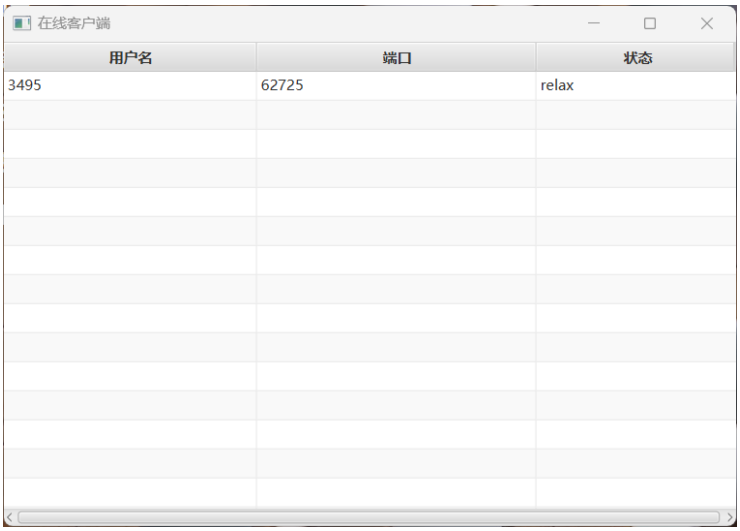

图 8: 在线客户端展示表

3.1.3.3 加入对局 当点击了加入对局按钮，客户端向服务端发送加入对局请求，获取服务端加入许可后，创建对局页面，若服务端通知当前没有可加入的对局，客户端弹出相关提示提醒用户。

3.1.3.4 查询历史对局按钮 当点击了查询历史对局按钮，客户端向服务端发送查看历史对局请求，获取服务端结果并展示。

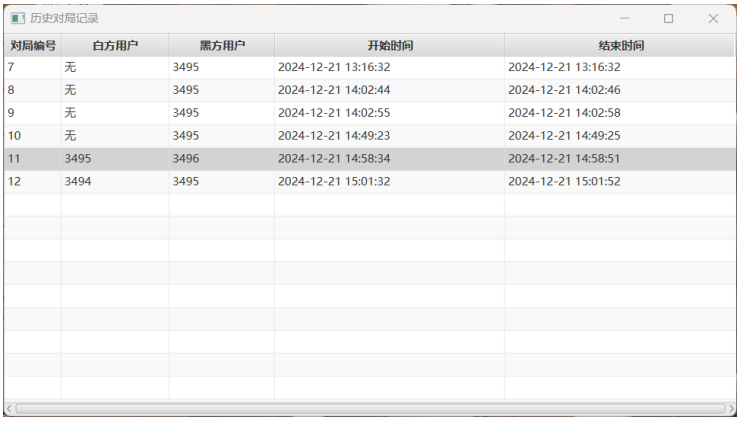

图 9: 历史对局展示表

在此基础上，用户双击对应的行，客户端将向服务端发送查询具体棋谱请求，获取服务端结果并展示

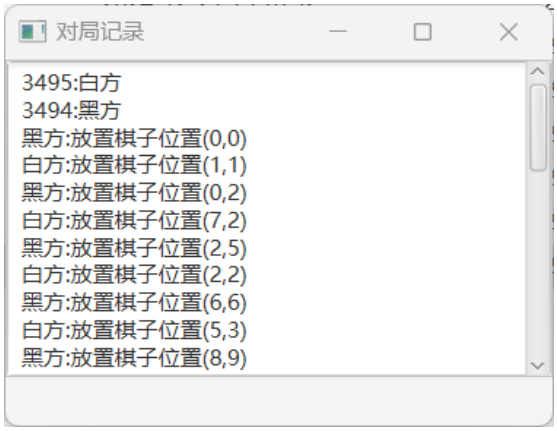

图 10: 棋谱展示

3.1.3.5 退出按钮 当用户点击退出按钮时，客户端程序断开与服务端点连接，并结束程序。

主界面控件的代码框架如下：

```
public class MainHandler {
    // 客户端与服务端通信的 Socket
    Socket socket;
    // 客户端通过 out 向服务端发送信息
    PrintWriter out;
    // 绑定查询在线用户按钮方法
    // 其他辅助成员变量略去
    @FXML
    protected void queryClientList(ActionEvent event) {}
    // 绑定查询用户历史对局按钮方法
    @FXML
    protected void queryAllHistory() {}
    // 绑定用户退出按钮方法
    @FXML
    public void quit(ActionEvent event) {}
    // 绑定用户创建对局按钮方法
    @FXML
    protected void createGame(ActionEvent event) {}
    // 绑定用户加入对局按钮方法
    @FXML
    protected void joinGame(ActionEvent event) {}
    //其他辅助成员函数略去
}
```

Listing 3: 注册界面控件

3.1.4 五子棋棋盘界面与功能的设计

五子棋棋盘类继承了 JavaFx 的 Pane 控件类，绑定了鼠标点击事件。点击了创建对局之后，若有人加入；或是加入了别人创建的对局，服务端就会向客户端发送对局开始的回应，随后客户端创建五子棋棋盘界面，并且用弹窗提示用户持黑子或是白子。考虑到用户会多次创建对局，因而在主界面的控件类中包含了一个五子棋棋盘类的成员对象，方便在对局终止之后对

其进行重置。

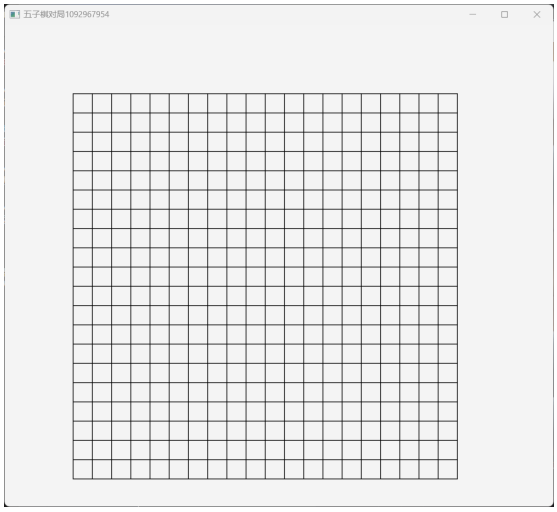

图 11: 五子棋棋盘界面

开始对局之后，用户在棋盘上点击，客户端会获取用户点击的棋盘坐标，发送给服务端，服务端验证坐标的合法性后记录下坐标，判断是否对局结束，之后再通知客户端绘制对应坐标点的棋子。对局结束后，客户端收到服务端的结果通过弹窗提示用户，然后关闭棋盘。如果一方在对局中途退出，服务端会通知另一方的客户端退出对局，之后两方棋盘窗口均关闭

3.1.5 客户端接收信息线程的设计

考虑到发送与接收的异步性，客户端还需要独立运行一个接收线程来专门负责接收服务器发来的信息。这个线程不涉及图形窗口，它的代码框架如下：

```
public class ReceiveMessageHandler implements Runnable {
    // 与服务端通信的 socket
    Socket socket;
    // 负责接收服务器信息
    BufferedReader in;
    // 和主界面线程共享的窗口
    Stage stage;
    // 创建该线程的主线程
    MainHandler mainHandler;
    // 其他负责成员对象略去
    // 接收服务端响应创建对局页面
    public void createGame(String response) {}
    // 接收服务端响应绘制棋子
    public void placeChess(String response) {}
    // 接收服务端响应退出对局
    public void quitGame() {}
    // 接收服务端响应终止对局
    public void overGame() {}
    // 接收服务端响应获取并展示在线客户端
    public void clientList(String response) {}
    // 接收服务器响应设置登录状态
    public void login() {}
    // 接收服务器响应开始对局
    public void startGame(String response) {}
    // 接收服务器响应获取并展示该用户所以对局记录
    public void showAllHistory(String response) {}
    // 接收服务器响应获取并展示具体棋谱
    public void showRecord(String response) {}
    // 其他辅助成员函数略去
}
```

Listing 4: 注册界面控件

3.2 服务端设计

3.2.1 与客户端交互模块的设计

服务端的根本逻辑是接收客户端请求并响应。为了满足不同客户端的并发请求，服务端采用一个客户端对应一个服务端线程 (ClientHandler) 的方式来处理。这样的设计使得服务端能够同时处理多个用户多个对局。考虑并发场景的临界资源访问问题，在一些共享的成员变量 (如当前在线用户表、当前可加入对局表和正在进行对局表) 操作时服务端使用 synchronized 关键字来保证数据一致性。以下是 ClientHandler 的代码框架：

```java
public class ClientHandler implements Runnable {
    // 登录用户信息，包含用户名、id、端口等
    private final ClientInfo clientInfo;
    // 在线用户表
    private final Map<String, ClientInfo> clients;
    // 等待开始的对局表
    private final Map<Integer, Game> waitingGames;
    // 正在进行的对局表
    private final Map<Integer, Game> runningGames;
    // 在线用户的用户名表（用于快速查询用户是否登录）
    private final Set<String> loggedUsers;
    // 发送在线客户端表到客户端
    public void sendClientsList(PrintWriter out) {}
    // 发生对局记录到客户端
    public void queryAllHistory(PrintWriter out) {}
    // 通知客户端是否可以创建对局
    public void createGame(PrintWriter out) {}
    // 通知客户端是否可以加入对局
    public void joinGame(PrintWriter out) {}
    // 通知客户端是否可以在特定位置放置棋子
    public void placeChess(PrintWriter out, String message) {}
    // 通知客户端退出对局（对局中止）
    public void quitGame(PrintWriter out) {}
    // 通知客户端结束对局（对局结束）
    public void overGame() {}
    // 通知客户端是否登录成功
    public void login(PrintWriter out, String inputLine) {}
    // 通知客户端是否注册成功
    public void register(PrintWriter out, String inputLine) {}
    // 发生对局棋谱到客户端
    public void queryRecord(PrintWriter out, String inputLine) {}
}
```

Listing 5: 注册界面控件

除此之外，考虑到用户信息和对局数据的持久性，服务端还采用 MySQL 数据库来储存对局数据和用户信息，使得用户登录功能、用户查询历史对局功能和用户查询对局棋谱功能得以实现。服务端设计了 DataBaseManager 类来与数据库进行交互。以下是 DataBaseManager 的代码框架：

```
public class DatabaseManager {
    private static final String DB_URL = " 略去";
    private static final String DB_USER = " 略去";
    private static final String DB_PASSWORD = " 略去";
    // 通过 con 对数据库进行操作
    private static Connection con;
    // 查询是否存在该用户，如果有就返回其 id，否则返回-1
    public static Integer login(String username, String password) {}
    // 往数据库插入该用户的信息，如果成功返回 true
    public static Boolean register(String username, String password) {}
    // 往数据库中插入对局的记录
    public static void gameRecord(String black_username, String white_username,
        String content, Timestamp start_time, Timestamp end_time) {}
    // 查询某个用户参与的对局记录并返回记录（不包含棋谱）的列表
    public static List<Record> queryAllRecords(String user_name) {}
    // 查询某个 id 的记录并返回记录的棋谱
    public static String queryRecord(int id) {}
}
```

Listing 6: 注册界面控件

3.2.2 数据库的设计

3.2.2.1 用户表设计 用户表由自增的 id、表示用户名的字符串 username 和表示密码的字符串 password 构成，以下是建表语句：

```
CREATE TABLE `user` (
    `username` varchar(50) NOT NULL,
    `password` varchar(45) CHARACTER SET latin1 NOT NULL,
    `id` int(11) NOT NULL AUTO_INCREMENT,
    PRIMARY KEY (`id`,`username`),
    UNIQUE KEY `username_UNIQUE` (`username`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8mb4
```

Listing 7: 用户表设计

3.2.2.2 对局记录表设计 对局记录表由自增的 id、表示黑方用户名的字符串 black_ username、表示白方用户名的字符串 white_username、表示棋谱内容的字符串 content、表示开始时间的 start_time 和表示结束时间的 end_time 组成，以下是建表语句：

```
CREATE TABLE `record` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `black_username` varchar(50) DEFAULT NULL,
    `white_username` varchar(50) DEFAULT NULL,
    `content` text NOT NULL,
    `start_time` datetime NOT NULL,
    `end_time` datetime NOT NULL,
    PRIMARY KEY (`id`)
    ) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8mb4
```

Listing 8: 用户表设计

4 测试与运行

4.1 程序测试

测试场景	测试步骤	测试样例	测试结果
用户登录	1. 打开客户端 2. 输入正确用户名和密码 3. 点击登录	用户名:3495 密码:123456	通过: 客户端跳转到主界面

	1. 打开客户端 2. 输入错误用户名和密码 3. 点击登录	用户名:3333 密码:123456	通过: 客户端弹窗显示密码错误
用户注册	1. 打开客户端 2. 输入服务端没有的用户名 3. 输入密码 4. 点击注册	用户名:3496 密码:123456	通过: 客户端弹窗显示注册成功, 之后跳转到登录页面
	1. 打开客户端 2. 输入服务端存在的用户名 3. 输入密码 4. 点击注册	用户名:3495 密码:123456789	通过: 客户端弹窗显示用户名重复
	1. 注册成功 2. 尝试用注册时的账号密码登录	用户名:3496 密码:123456	通过: 客户端跳转到主界面
	1. 注册失败 2. 尝试用注册时的账号密码登陆	用户名:3495 密码:123456789	通过: 客户端弹窗显示密码错误
用户创建对局	1. 登录后进入主界面 2. 点击创建对局按钮	用户名:3495 密码:123456	通过: 客户端弹出对局界面
	1. 登录后进入主界面 2. 点击创建对局按钮 3. 再次点击创建对局按钮	用户名:3495 密码:123456	通过: 客户端第一次弹出对局界面 第二次弹窗提示已经在对局中
	1. 客户端 A 登录后进入主界面 2. 点击创建对局按钮 3. 客户端 B 登录后进入主界面 4. 点击加入对局按钮 客户端 A 再次点击创建对局按钮	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过: A 客户端第一次弹出对局界面 第二次提示对局开始 第三次弹窗提示已经在对局中 B 客户端弹出对局界面和对局开始提示
	1. 客户端 A 登录后进入主界面 2. 点击创建对局按钮 3. 客户端 B 登录后进入主界面 4. 点击加入对局按钮 客户端 B 点击创建对局按钮	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过: A 客户端第一次弹出对局界面 第二次提示对局开始 B 客户端第一次弹出对局界面和对局开始提示 第二次弹窗提示已经在对局中
用户加入	1. 登录后进入主界面 2. 点击创建对局按钮	用户名:3495 密码:123456	通过: 客户端弹窗提示没有可加入的对局
	1. 客户端 A 登录后进入主界面 2. 点击创建对局按钮 3. 客户端 B 登录后进入主界面 4. 点击加入对局按钮	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过: A 客户端第一次弹出对局界面 第二次提示对局开始 B 客户端弹出对局界面和对局开始提示
用户端进行对局 服务端进行记录	1. 客户端 A 创建对局 2. 客户端 B 加入对局 3.A、B 轮流下棋直到一方获胜	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 两方客户端弹窗提示结果 关闭弹窗后 对局界面关闭 服务端数据库记录的棋谱包含 A、B 下棋过程以及结果
	1. 客户端 A 创建对局 2. 客户端 B 加入对局 3. 轮到 A 的回合 4.A 点击已经下过的位置 5.A 点击没下过的位置	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 客户端 A 第一次点击后没有反应 第二次点击后出现棋子 服务端未收到 A 的第一次点击记录

	1. 客户端 A 创建对局 2. 客户端 B 加入对局 3. 轮到 B 的回合 4.A 点击没下过的位置	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 客户端 A 弹窗提示当前尚未轮到 服务端未收到 A 的点击记录
	1. 客户端 A 创建对局 2. 客户端 B 加入对局 3.A、B 轮流下棋 4. 在没结束时 A 退出	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 客户端 A 和客户端 B 同时退出对局界面 服务端数据库中记录的棋谱包含 A、B 的下棋过程以及最后对局中止
	1. 客户端 A 创建对局 2. 客户端 A 关闭对局界面	A 用户名:3495 密码:123456	通过 客户端 A 退出对局界面 服务端数据库中记录的棋谱只包含 A 的用户名以及对局中止
用户查询在线客户端	1. 客户端 A 登录 2.A 查询在线客户端 3. 客户端 B 登录 4.A、B 查询在线客户端	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 客户端 A 第一次显示只有 A 的登录记录 客户端 A 第二次显示 A、B 登录记录 客户端 B 显示 A、B 的登录记录
	1. 客户端 A 登录 2. 客户端 B 登录 3.A 查询在线客户端 4.B 退出 5.A 查询在线客户端	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 客户端 A 第一次显示 A、B 的登录记录 客户端 A 第二次显示 A 登录记录
用户查询历史对局记录	1. 客户端 A 登录 2. 客户端 A 查询历史对局记录 3. 客户端 B 登录 4. 客户端 A、B 进行对局至结束 5. 客户端 A 查询历史对局记录	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 客户端 A 第一次查询到之前的历史对局记录 客户端 A 第二次查询到历史对局记录包含刚发生的对局
	1. 客户端 A 登录 2. 客户端 A 查询历史对局记录 4. 客户端 A 创建对局后退出 5. 客户端 A 查询历史对局记录	A 用户名:3495 密码:123456	通过 客户端 A 第一次查询到之前的历史对局记录 客户端 A 第二次查询到历史对局记录包含刚创建的对局
	1. 客户端 A 登录 2. 客户端 A 查询历史对局记录 3. 客户端 B 登录 4. 客户端 A、B 进行对局，中途退出 5. 客户端 A 查询历史对局记录	A 用户名:3495 密码:123456 B 用户名:3495 密码:123456	通过 客户端 A 第一次查询到之前的历史对局记录 客户端 A 第二次查询到历史对局记录包含刚退出的对局
用户查询记录棋谱	1. 客户端 A 登录 2. 客户端 A 查询历史对局记录 3. 客户端 A 双击一条记录	A 用户名:3495 密码:123456	通过 客户端 A 弹窗显示该记录的棋谱
	1. 客户端 A 登录 2. 客户端 A 查询历史对局记录 3. 客户端 A 记录外的其他区域	A 用户名:3495 密码:123456	通过 客户端 A 双击没有反应

表 1: 程序测试表

4.2 程序运行

运行程序，进入用户登录页面

图 12: 登录界面

点击注册，进入用户注册页面

图 13: 注册界面

输入用户名和密码，完成注册

图 14: 完成注册

回到登录页面，进行登录

图 15: 完成登录

进入主界面

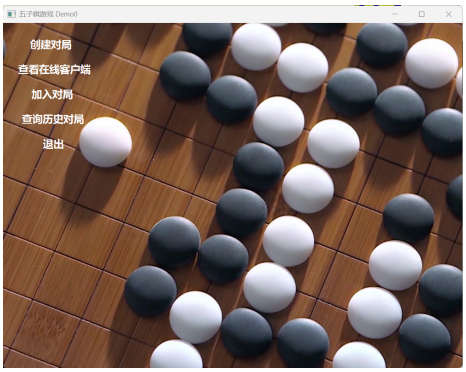

图 16: 主界面

查看在线客户端

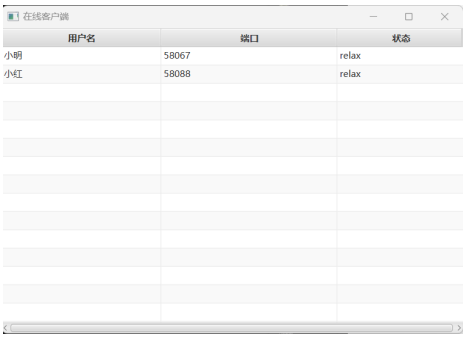

图 17: 在线客户端

点击创建对局

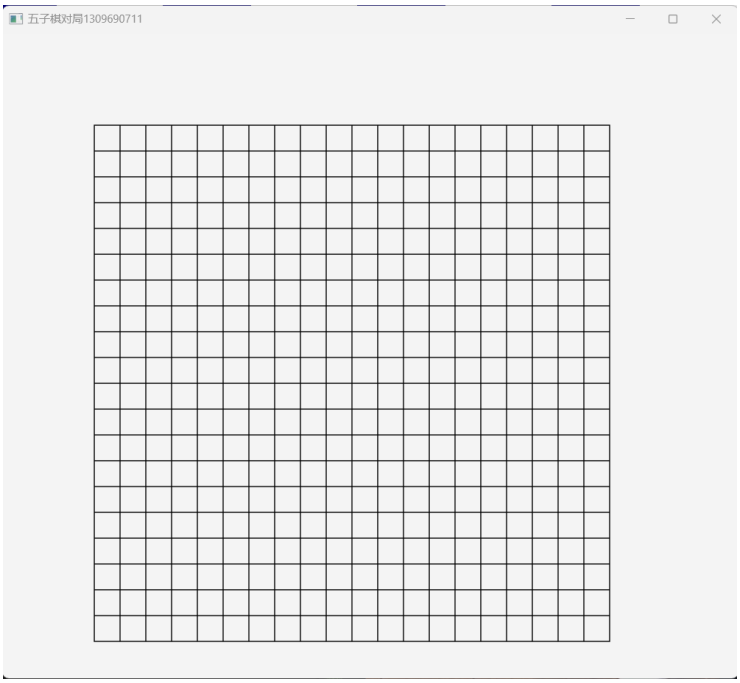

图 18: 创建对局

这时另一个客户端加入对局，对局开始

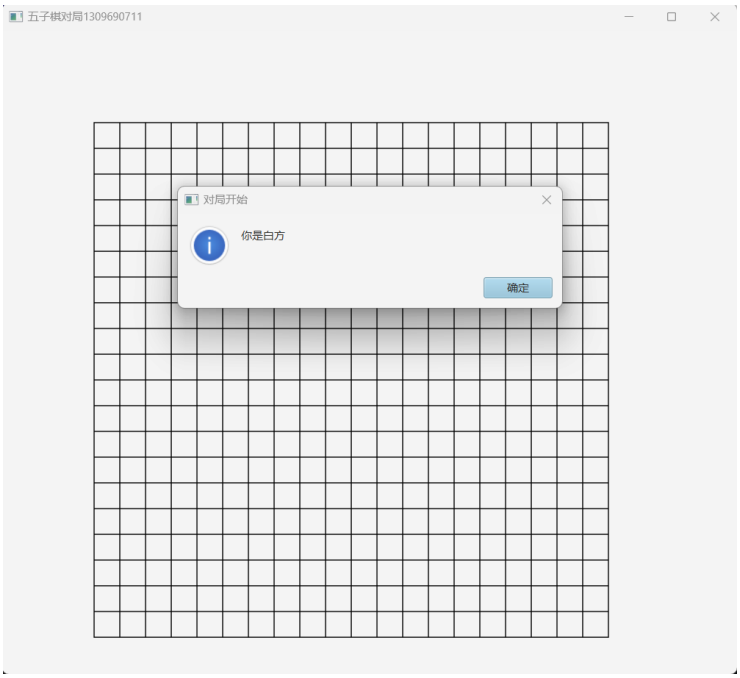

图 19: 对局开始

两边开始对局

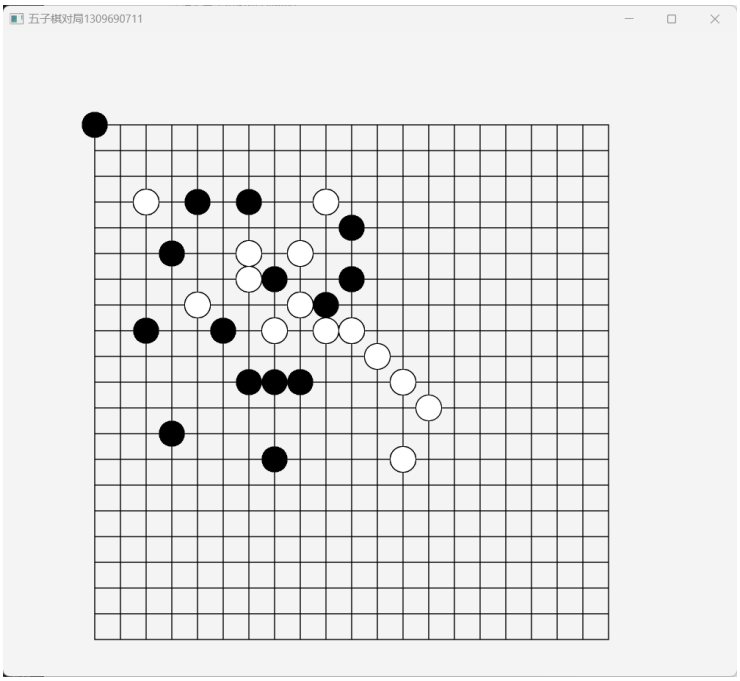

图 20: 对局中

白方赢得了对局

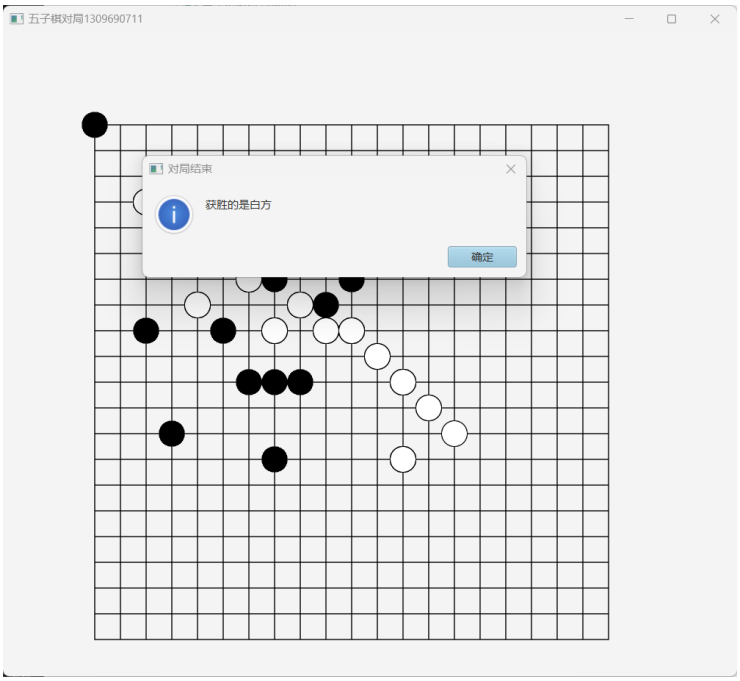

图 21: 对局结束

查看历史对局

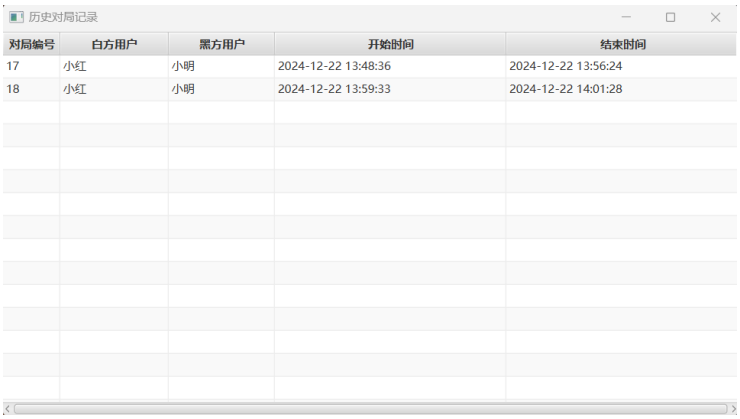

图 22: 历史对局

查看刚才对局的棋谱

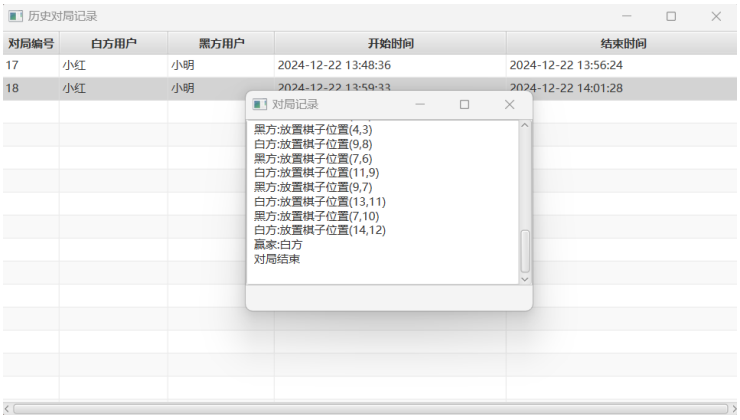

图 23: 对局棋谱

最后回到主界面退出客户端。

5 总结

在本次项目中，我使用 Java 语言，依靠网络编程、多线程编程、数据库操作和 GUI 设计方面的知识，初步搭建起了一个五子棋在线对局的应用程序。

经过这次编程实践，我不仅巩固了在 Java 程序课上学习到的相关知识，也通过互联网学到了在课程之外的一些前沿知识。譬如，在 GUI 的设计上，我通过 JavaFx、FXML 加上 scene builder 的设计模式，实现可视化、低代码地搭建 GUI 界面，简化了设计过程。同时也引入了 css 的样式设计，使得我可以向写前端网页一样装饰我的 GUI 控件。

此外，和其他诸如网络聊天室之类的应用不同，本项目作为一个网络游戏程序，还需要保证输赢的可靠性和不可篡改性，保证用户的恶意行为会被服务端所阻拦。因此，本项目在设计时也考虑到了可能存在的客户端恶意行为，将所有的对局判断、数据记录都放在服务端上执行和储存，这样，即使有用户通过恶意篡改客户端，发送非法的操作时，服务端也能进行阻拦并保证对局的正确性。这也是我在编写代码时花费时间最多的模块。

然而，由于本项目的编写时间有限，且本项目只是由个人完成的一次作业而已，还存在许多的复杂情况并没有被很好的考虑，还存在很多有意思的功能没有被实现。譬如，本项目中用户加入对局时只能随机加入，并不能由

自己选择。本项目的棋谱为文本形式，没有使用可视化的棋盘来展现等。鉴于临近期末，有限的时间和精力都不太支持我继续深入开发项目，而只能遗憾止步于此。

但总而言之，通过本次项目实践，我加深了对 Java 面向程序设计的理解，代码管理和编程水平亦有提高，这就足够了。