浙江大学计算机学院

Java 程序设计课程报告

2022—2023 学年秋冬学期

题目	井字棋游戏平台
学号	
学生姓名	
所在专业	计算机科学与技术
所在班级	

目　　录

1 引言

本次我开发的程序是一个简易井字棋游戏平台程序。这个题目需要用到Socket 通信、多线程、Java Swing 标准库等知识，有助于加深对 Java 程序设计语言和 Java 库的使用的理解。

1．1 设计目的

按照设计要求，我所设计的井字棋游戏平台程序是一个具有一定的用户界面的、支持客户端网络通信和匹配、游戏逻辑完善的小游戏程序。具体功能如下：

(1) 通过主程序入口选择打开服务器端程序或者客户端程序。由于程序运行在一个特定端口号上，同时只能运行一个服务器端程序；客户端程序数量则不受限制。

(2) 服务器端程序需要从客户端程序接受请求数据包、对客户端发送一定的指令并管理目前所有客户端和游戏房间的运作情况。服务器端也需要对所有游戏房间以及套接字响应程序进行线程管理。

(3) 客户端程序通过 GUI 接受游戏用户的操作，并构造对应的请求数据包，与服务器端进行通信。通过服务器端与客户端的双向通信，完成不同客户端间井字棋游戏的正常运作。

1．2 设计说明

本程序采用 Java 程序设计语言，在 NetBeans IDE v8.0.2 下编辑、编译与调试。全部程序编写均由我一人完成。

本程序没有使用任何外部库和外部资源。

2 总体设计

2．1 功能模块设计

本程序需实现的主要功能有：

(1) 对于用户端，需要完成用户间的创建房间、退出房间、匹配对手、准备及开始游戏、游戏中投降等操作的正常响应；

(2) 对于网络层面，需要完成服务器端和客户端间的套接字通信，让服务器能够多线程地对每个用户的请求即时做出响应；

(3) 对于服务器端，需要让服务器能即时监视目前存在的房间情况，以确保用户端游戏过程的正常运行；

(4) 对于用户，需要让用户在开始游戏后能感受到正确的游戏逻辑，并能通过鼠标操作完成游戏并得出游戏结果。

程序的总体功能较为复杂，如下图所示：

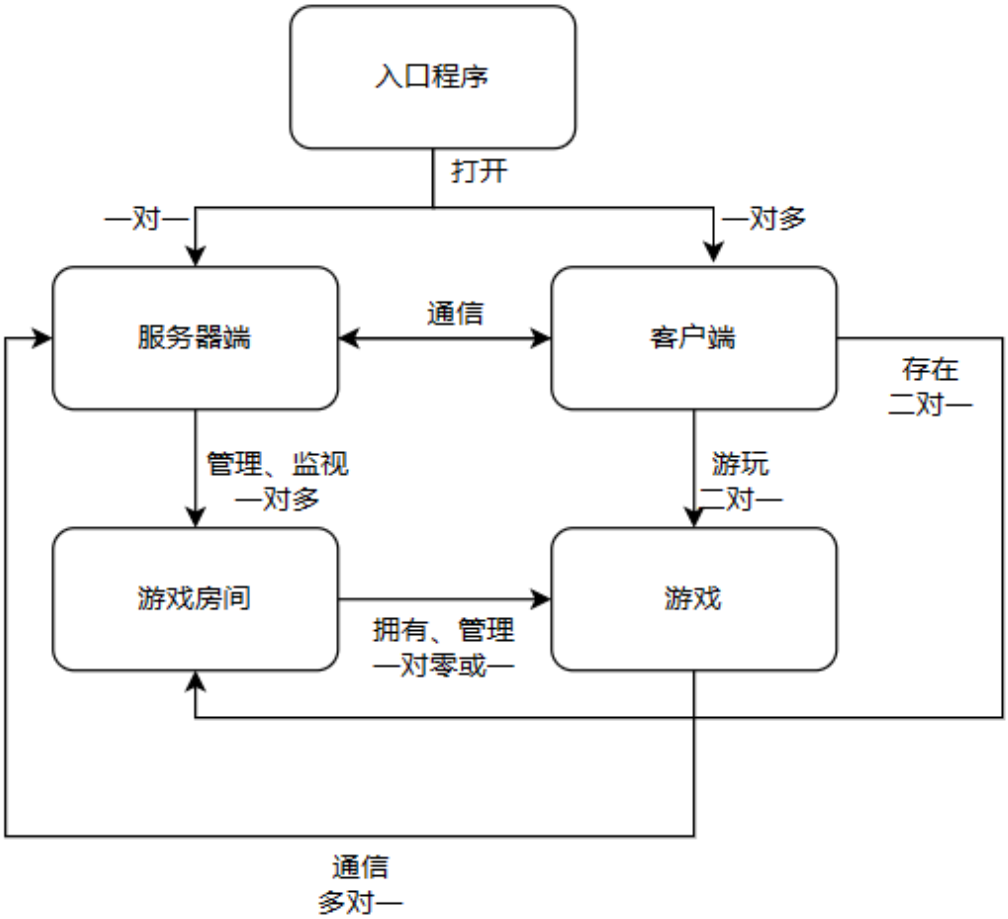

图 1 总体功能图

2. 2 流程图设计

本程序的绝大部分时间在多线程地处理进程通信和游戏逻辑，因此难以画出一个囊括全部流程的流程图。类间互联和通信情况如下：

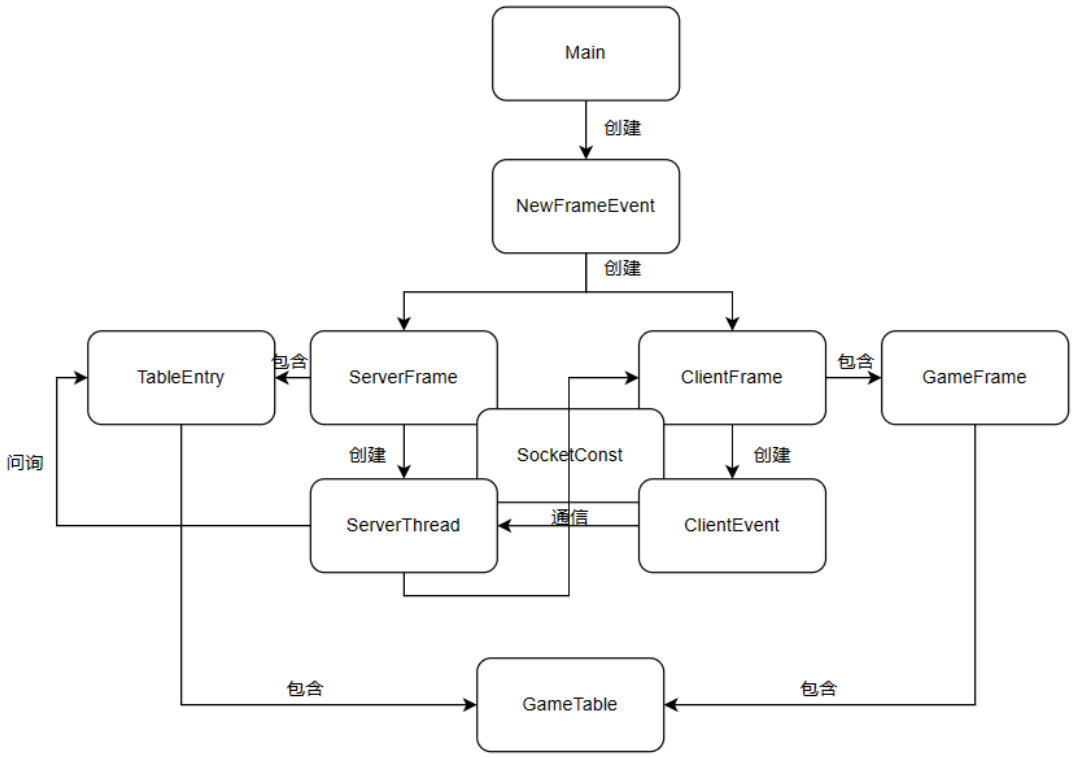

图 2 类间关联示意图

其中，每局游戏及其维护的流程比较明晰，其流程如下：

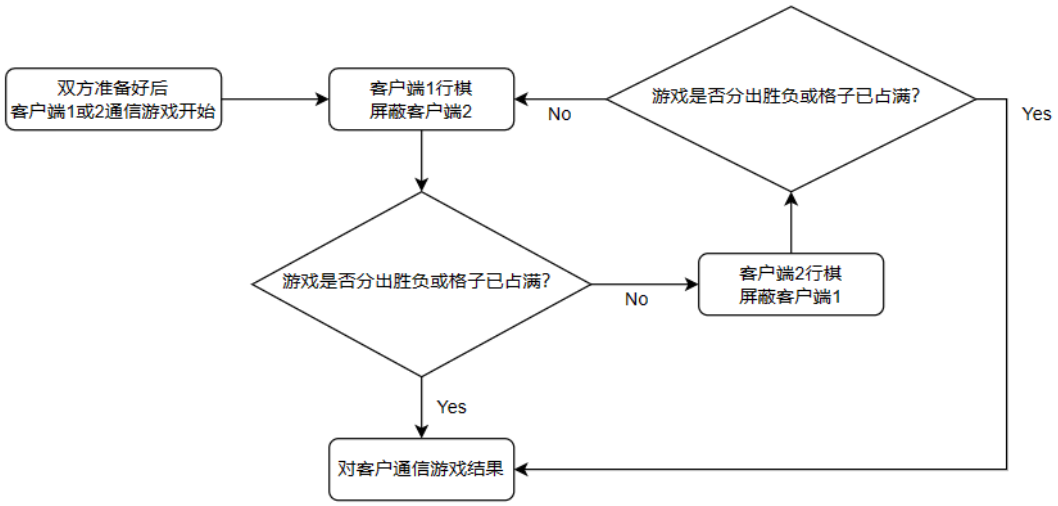

图 3 游戏流程图

3　详细设计

3．1　Main 类设计

该类的源代码存在 Main.java 文件中。声明：

```
public class Main extends JFrame {}
```

在本程序设计中，Main 类是整个程序的入口类，同时需要负责处理一个简单的启动程序 GUI；因此，我让该类由 JFrame 类派生而来。程序入口是类中的静态 main()方法，该方法实例化一个 Main 类，从而调用一次构造方法 Main()，在该方法中构造 GUI。

响应模式为，使用 Java swing 中的 JButton 类接受用户输入；鉴于启动程序需要完成的响应较为简单，只需要打开新窗口的操作，我另设计了实现 JActionListener 接口的 NewFrameEvent 类（稍后说明）进行处理。

3．2　NewFrameEvent 类设计

该类的源代码存在 NewFrameEvent.java 文件中。声明：

```
public class NewFrameEvent implements ActionListener {}
```

该类是一个设计比较简单的类，只负责处理启动程序通过按钮类对服务器端程序或用户端程序的启动，因此我让该类由 JActionListener 类派生而来。原本的设计目的是通过该类完成所有不需要参数的新窗口类的构造，但实际上本程序中只有这两个类可以通过 NewFrameEvent 类启动。

NewFrameEvent 类有以下成员方法和变量：

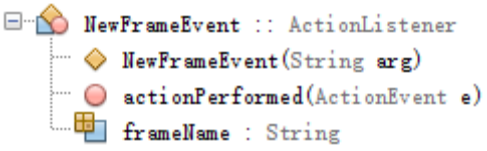

图 4 NewFrameEvent 类的成员

以下是该类中成员数据和方法的详细说明：

（1）成员变量

① private final String frameName：这个成员变量由构造函数直接设置，标明需要打开的窗口类的类名。

（2）方法

① public NewFrameEvent(String arg)：这是该类唯一的构造方法。在进入构造函数后，按照参数设置 frameName。

② public void actionPerformed(ActionEvent e)：实现接口方法。该方法打开一个新线程，并在该线程中的 run()方法内通过反射机制实例化一个 FrameName 所代表的类，从而调用该类对应的构造函数。

```
Class.forName(frameName).newInstance();
```

3．3　SocketConst 类设计

该类的源代码存在 SocketConst.java 文件中。声明：

```
public final class SocketConst {}
```

SocketConst 类是一个纯粹的工具类，只包含许多 int 类型的常量定义和一个私有的构造函数，以防止实例化。

这些常量是接下来 ServerFrame 和 ClientFrame 进行套接字通信的标准提示符，主要分为以下三类：

（1）客户端程序给服务器发送数据的标准提示符：

```
public static final int ConnectToEmpty = 1;
public static final int ConnectToRandom = 2;
public static final int Standby = 3;
public static final int ExitTable = 4;
public static final int StartGame = 5;
public static final int Surrender = 6;
public static final int Step = 7;
public static final int GetClientID = 8;
```

图 5 客户端 Socket 标准提示符

（2）服务器对客户端指令的标准提示符：

```
public static final int ChangeTable = 200;
public static final int ChangeReady = 201;
public static final int ChangeGaming = 202;
public static final int ChangeOpnt = 203;
public static final int ChangeOpntReady = 204;
```

图 6 服务器端 Socket 标准提示符（一）

（3）服务器端对客户端发送的维护客户端游戏进程的标准提示符：

```
public static final int ChangeGameStatus = 300;
public static final int Win = 301;
public static final int Lose = 302;
public static final int Draw = 303;
```

图 7 服务器端 Socket 标准提示符（二）

这些定义的常量将在之后服务器端和客户端通信的部分使用。

3．4 GameTable 类设计

该类的源代码存在 GameTable.java 文件中。声明：

```
public final class GameTable {}
```

这个类用来维护游戏进程信息。在服务器端，每个游戏房间都需要实例化一个 GameTable 对象来维护当前房间游戏进行的情况；在客户端，每个客户端也需要实例化一个 GameTable 对象以保存目前游戏的进度信息。

GameTable 类有以下成员变量和方法：

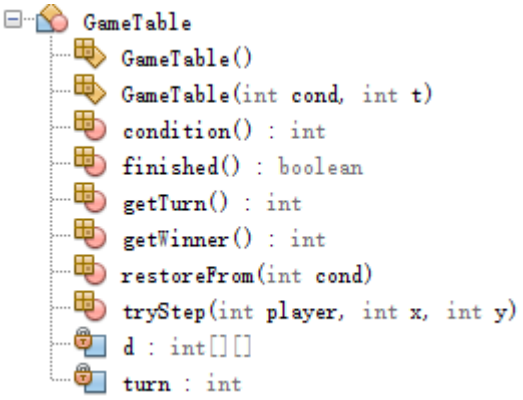

图 8 GameTable 类的成员

以下是该类中成员数据和方法的详细说明：

（1）成员变量

① private int[][] d：这个二维数组保存当前的棋盘状态。各个元素为 0 代表没有下子，1 代表玩家 1 占据此格，2 代表玩家 2 占据此格。

② private int turn：保存目前轮到哪个玩家下棋。

（2）方法

① GameTable()：该类的默认构造方法，相当于新开始一局游戏。把 d[][] 初始化为全 0，让 turn 随机为 1 或者 2（也就是随机决定先后手）。

② GameTable(int cond, int t)：这个构造方法相当于复制一局游戏。cond 是棋盘状态 d[][]的一个 hash 值，t 是对应 turn 的变量。这个方法调用 restoreFrom()从 cond 中复制棋盘状态，并设置 turn 为 t。

③ int getTurn()：单纯地返回当前的 turn 值。

④ int condition()：把当前棋盘状态 d[][]计算出一个 hash 值并返回。实际上因为设计的棋盘只有 3*3 所以这个 hash 值的构造和解码都无比简单。

⑤ void restoreFrom(int cond)：从 condition()方法产生的一个 hash 值重建棋盘状态。

⑥ void tryStep(int player, int x, int y)：表示玩家 player 尝试在(x,y)位置下棋。如果该操作合法（轮到该玩家的回合，且该位置未下棋），那么更新棋盘状态，并且让 turn 轮换给对手。

⑦ boolean finished()：检查棋盘状态，如果目前应该是游戏结束的情况（三连子或棋盘已占满）那么返回 true。否则返回 false。

⑧ int getWinner()：检查棋盘状态，如果有胜利者返回该胜利者。否则返回 0（可能是平局，或游戏尚未结束）。

这个类的代码部分都相对简单，不太需要进行详细说明。

3.5 ClientFrame 类设计

该类的源代码存在 ClientFrame.java 文件中。声明：

```
public final class ClientFrame extends JFrame {}
```

该类就是程序的客户端类，用以构造用户所使用的客户端。该类在程序中通过 NewFrameEvent 类调用进行实例化。

该类有以下成员变量和方法：

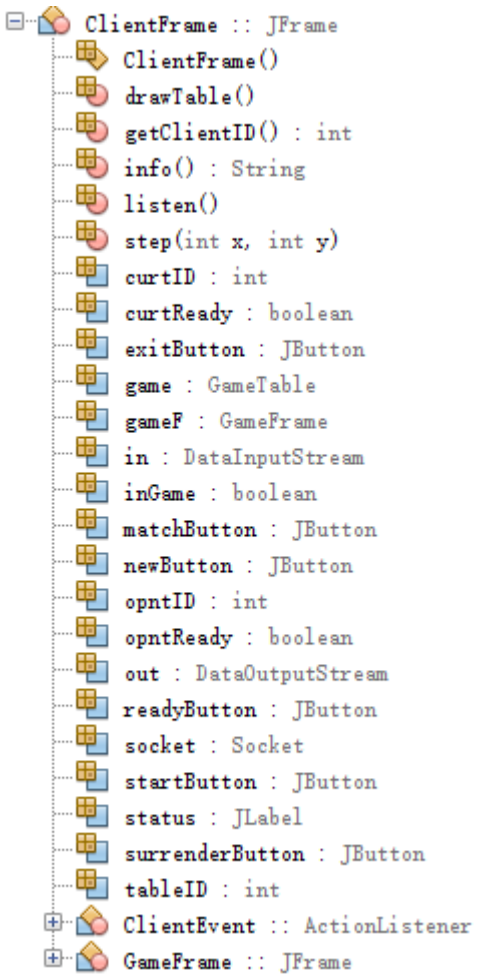

图 9 ClientFrame 类的成员

以下是该类中成员数据和方法的详细说明：

（1）成员变量

① int tableID, curtID, opntID; boolean curtReady, opntReady, inGame：这些变量保存当前玩家所在的桌号、用户和对手 ID、准备情况以及目前游戏是否已开始的信息。

② GameTable game：保存当前桌的游戏情况。

③ GameFrame gameF：当前桌的游戏窗口（另开）。

④ Socket socket; DataInputStream in; DataOutputStream out：与服务器端进行 Socket 通信的相关类。

⑤ JLabel status; JButton newButton, matchButton, readyButton, exitButton, startButton, surrenderButton：窗口组件。status 表示目前玩家所在房间的情况，其他按钮表示用户在当前情况下的各种操作。

（2）方法

① ClientFrame()：该类的唯一构造方法，由 NewFrameEvent 类调用。方法内完成 GUI 和 socket 的设置，之后调用 listen()保持对套接字通信的监听。需要注意，设置 JButton 类后对每个按钮都添加了一个 ClientEvent 类（稍后说明）的监听器。除去 GUI 部分的其他核心设置代码：

```
if(ServerFrame.hasServer == false) return;
   //如果不存在服务器，不能打开客户端

socket = new Socket("localhost", 5646);
in = new DataInputStream(socket.getInputStream());
out = new DataOutputStream(socket.getOutputStream());
//与服务器端建立 socket 连接

curtID = getClientID();
tableID = -1;
//从服务器端获取唯一的用户 id

listen();
//开始监听服务器端的通信
```

② String info()：以字符串方式返回当前客户端及其所在房间的信息。

③ int getClientID()：向输入流写入 SocketConst.GetClientID 提示符，从 socket 输出流中读取服务器返回的 clientID 并返回。

④ void step(int x, int y)：向输入流写入当前客户端想要在(x,y)位置下棋的信息。

⑤ void drawTable()：绘制当前用户窗口的界面。

⑥ void listen()：监听服务器端传回的通信数据包，进行响应操作。该方法的主要结构如下：

```
while(true){
   int event = in.readInt();
   if(event == SocketConst.ChangeTable) //忽略详细处理
   else if(event == SocketConst.ChangeReady) //忽略详细处理
   else if(event == SocketConst.ChangeGaming) //忽略详细处理
   else if(event == SocketConst.ChangeOpnt) //忽略详细处理
   else if(event == SocketConst.ChangeOpntReady) //忽略详细处理
   else if(event == SocketConst.ChangeGameStatus) //忽略详细处理
   else if(event == SocketConst.Win) //忽略详细处理
   else if(event == SocketConst.Lose) //忽略详细处理
   else if(event == SocketConst.Draw) //忽略详细处理
```

```
        drawTable(); //重新绘制 GUI
    }
```

此外还有两个该类的内部类，将继续进行设计说明。该类的运行流程图如下，并可以参照测试部分：

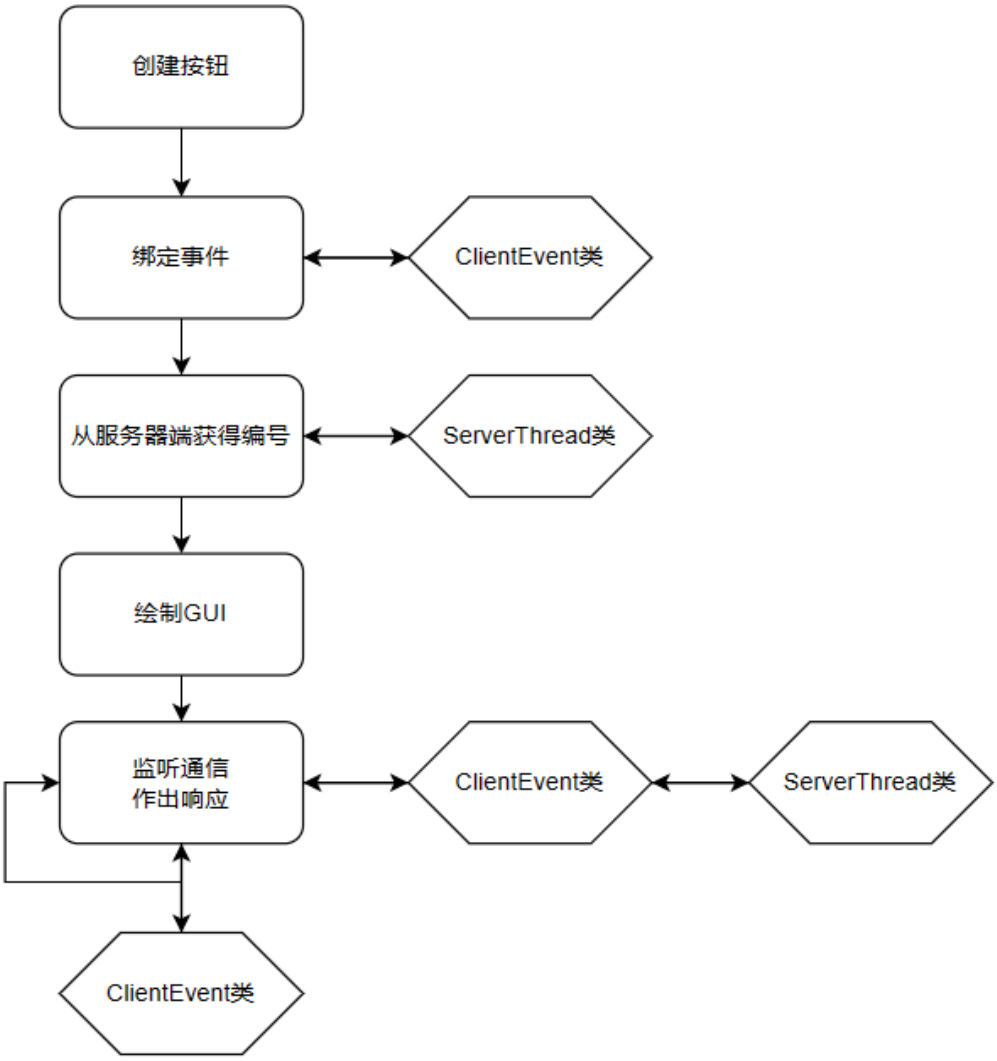

图 10 ClientFrame 类流程图

3．6　ClientFrame$ClientEvent 类设计

该类的源代码存在 ClientFrame.java 文件中。声明：

```
class ClientEvent implements ActionListener {}
```

该类是 ClientFrame 类的一个内部类，是客户端用来处理按钮信息的通用类。该类在构造时保存一个提示符常量 event，当对应按钮被触发时就向服务器端发送提示符对应操作的相关信息。

其实也就是直接向 out 流写入这个 event。也许这个类可以算是程序重构时留下来的一个垃圾类，因为这点功能应该不太需要专门写个类来做……

```
out.writeInt(event);
```

3．7 ClientFrame$GameFrame 类设计

该类的源代码存在 ClientFrame.java 文件中。声明：

```
final class GameFrame extends JFrame {}
```

该类是 ClientFrame 类的一个内部类，是与客户端唯一对应的游戏界面窗口。该类的目的就是建立一个新窗口并画出当前棋盘情况。为了保持能在原位置画出更新的棋盘，该类同时接受一个 x 和 y 坐标作为构造函数的参数。该类使用两个匿名类（分别派生自 JPanel 类和 MouseListener 类）完成棋盘绘画以及对棋手鼠标点击下棋事件的监听（当监听到时，调用外部类的 step()方法）。

其实现的主要代码：

```
getContentPane().add(new JPanel(){
public void paint(Graphics graphics){
   //省略画出棋盘的操作
}
});
addMouseListener(new MouseListener(){
//省略其他鼠标事件
public void mouseClicked(MouseEvent e){
if(0 <= e.getX() && e.getX() <= 80){
   if(0 <= e.getY() && e.getY() <= 90) step(0, 0);
   else if(90 <= e.getY() && e.getY() <= 190) step(0, 1);
   else if(190 <= e.getY() && e.getY() <= 300) step(0, 2);
}else if(80 <= e.getX() && e.getX() <= 180){
   if(0 <= e.getY() && e.getY() <= 90) step(1, 0);
   else if(90 <= e.getY() && e.getY() <= 190) step(1, 1);
   else if(190 <= e.getY() && e.getY() <= 300) step(1, 2);
}else if(180 <= e.getX() && e.getX() <= 300){
   if(0 <= e.getY() && e.getY() <= 90) step(2, 0);
   else if(90 <= e.getY() && e.getY() <= 190) step(2, 1);
   else if(190 <= e.getY() && e.getY() <= 300) step(2, 2);
}
}
});
```

3．8　ServerFrame 类设计

该类的源代码存在 ServerFrame.java 文件中。声明：

```
public final class ServerFrame extends JFrame {}
```

该类就是程序的服务器端类，用以构造服务器端，对用户端的请求进行响应。该类在程序中通过 NewFrameEvent 类调用进行实例化。

该类有以下成员变量和方法：

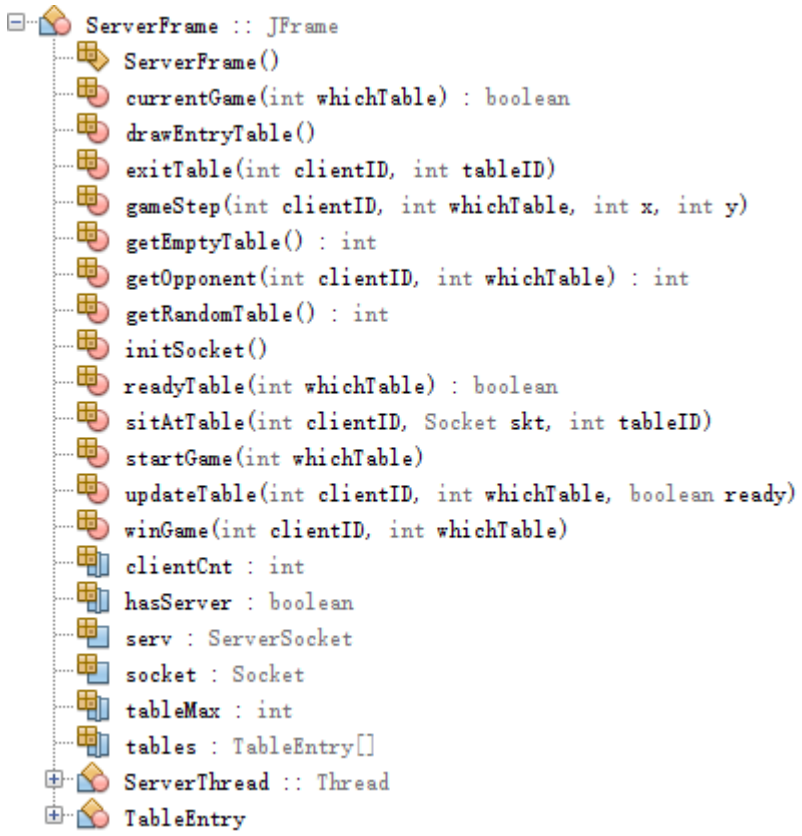

图 11 ServerFrame 类的成员

以下是该类中成员数据和方法的详细说明：

（1）成员变量

① static final int tableMax：一个程序员设定的常量，表示一个服务器支持的最大房间数量。为方便调试，代码中设置为 2。

② static boolean hasServer：一个布尔变量，表示服务器是否有在运作。如果已经存在服务器，不能打开新的服务器。

③ static int clientCnt：已连接客户端数量的计数，用以给客户端分配 ID。

④ static TableEntry[] tables：客户端所维护的每一房间的详细信息，以 TableEntry 类（稍后说明）表示。

⑤ ServerSocket serv; Socket socket：通信用 socket。

（2）方法

① ServerFrame()：该类的唯一构造方法，由 NewFrameEvent 类调用。方法内完成 GUI 的设置、初始化每个房间的信息，并调用 drawEntryTable()显示当前的房间信息。最后调用 initSocket()开始进行通信的监听。

② void initSocket()：初始化服务器端 socket（serv），循环监听新连接，对每个新连接建立一个 ServerThread 类（稍后说明）的线程并 start。

```
(new ServerThread(socket)).start();
```

③ void drawEntryTable()：对每个 tables 的元素调用 tables[i].info()生成一个 JLabel，挂载到 GUI 上，以完成对所有现存房间的监视。

④ 其他方法：基本上都是与 TableEntry 类的 tables[i]进行互动，修改房间信息；少数是从 tables[]中获取信息，并返回一个满足要求的房间号。这些方法由于会被多个 ServerThread 线程调用，因而都声明为 synchronized 的，以防止对 tables[]的同时修改。

ServerFrame 类的外部方法实际上只与 TableEntry 类通信，其他的通信操作实际上都由在 initSocket()中产生的 ServerThread 实例完成。

该类的流程图：

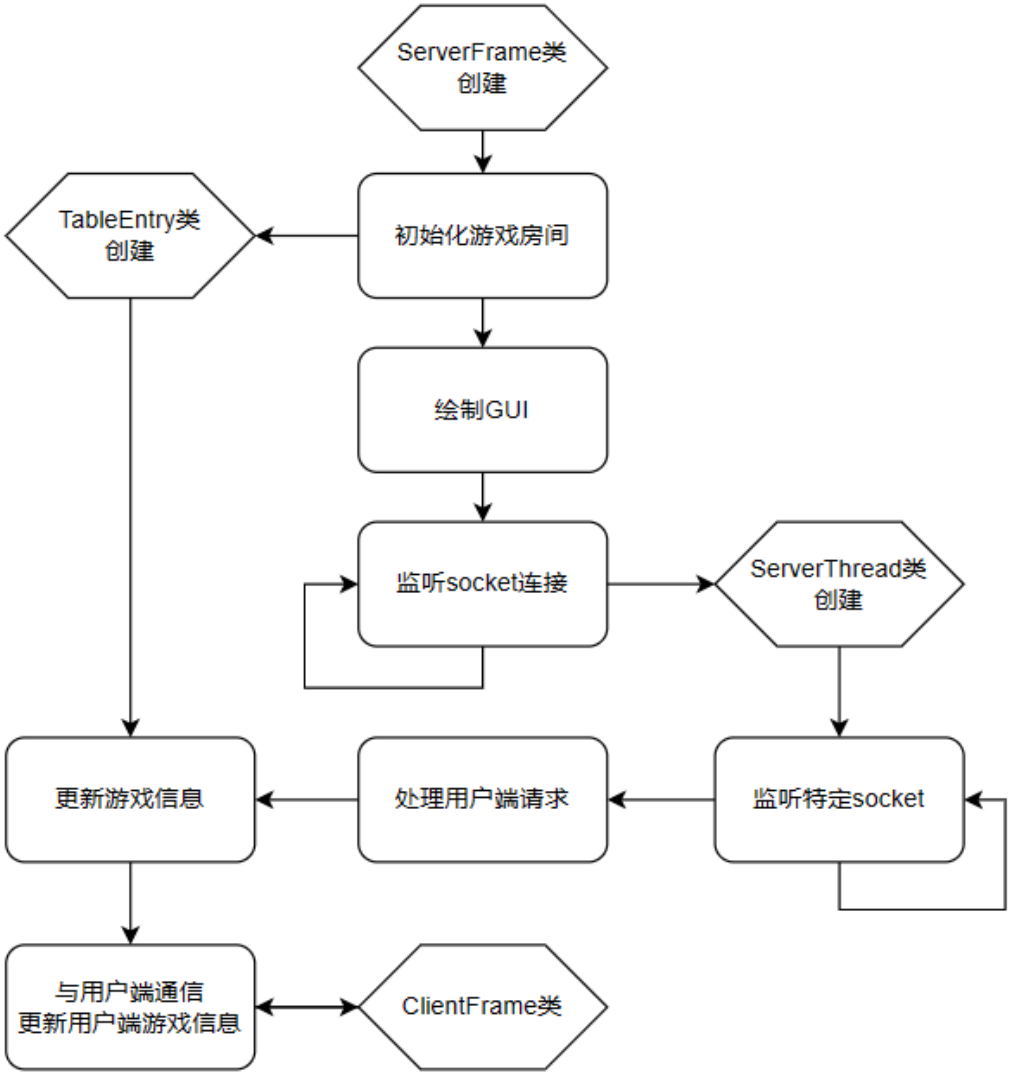

图 12 ServerFrame 类流程图

3．9　ServerFrame$ServerThread 类设计

该类的源代码存在 ServerFrame.java 文件中。声明：

```
class ServerThread extends Thread {}
```

该类是 ServerFrame 类的一个内部类，主要用于处理客户端与每个进程的通信。

该类有以下成员变量和方法：

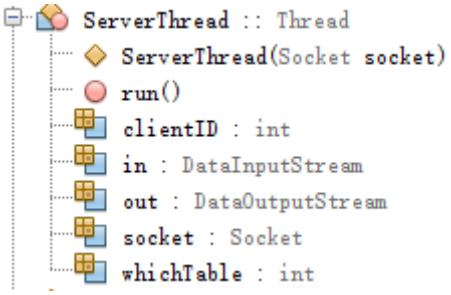

图 13 ServerThread 类的成员

whichTable 和 clientID 保存该线程所对应的客户端的信息；socket、in、out 用以与对应客户端进行通信。显然，一个 ServerThread 线程只对应一个客户端。

构造方法对这些数据进行初始化设定。监听过程在 run()中实现，其主要代码：

```
while(true){
   int opr = in.readInt();
   if(opr == SocketConst.ConnectToEmpty) //忽略详细处理
   else if(opr == SocketConst.ConnectToRandom) //忽略详细处理
   else if(opr == SocketConst.Standby) //忽略详细处理
   else if(opr == SocketConst.ExitTable) //忽略详细处理
   else if(opr == SocketConst.StartGame) //忽略详细处理
   else if(opr == SocketConst.Surrender) //忽略详细处理
   else if(opr == SocketConst.Step) //忽略详细处理
   else if(opr == SocketConst.GetClientID) //忽略详细处理
   drawEntryTable(); //更新主线程显示的房间信息
}
```

3．10　ServerFrame$TableEntry 类设计

该类的源代码存在 ServerFrame.java 文件中。声明：

```
class TableEntry {}
```

该类是 ClientFrame 类的一个内部类，用以保存每个房间的信息。同时也完成服务器端对房间内客户端的指令性操作。

该类有以下成员变量和方法：

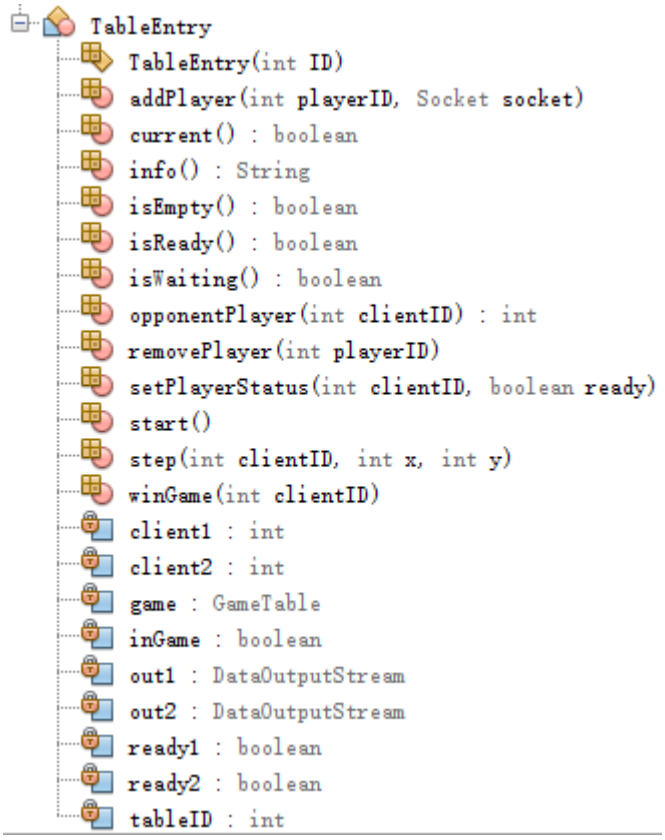

图 14 TableEntry 类的成员

其数据成员均用以维持特定房间内玩家和游戏进程的信息。

该类的许多方法通过对应房间内玩家客户端的 socket 的输出流 out1 和 out2 完成对游戏进程的控制，也就是说，用户的游戏操作数据主要是与该类进行交互。另外有些方法是判断房间满足特定要求，或返回一定的房间信息。

info()方法返回一个代表房间内状况的字符串，这个字符串返回到服务器，用以监视房间信息。

4 测试与运行

4．1 程序测试

程序主要通过 2-4 客户端与 1 服务器之间的交互进行测试。所有常见异常均已排除，或得以在程序内处理。

经大量测试，实验要求的功能实现已经完备，可以期待后续完善一些其他功能和美化 GUI。

4．2 程序运行

以下以图示方式进行运行效果展示。各图信息详见图名。

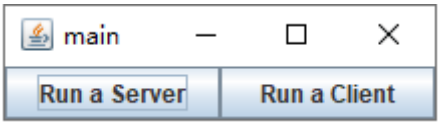

图 15 程序启动入口（Main）

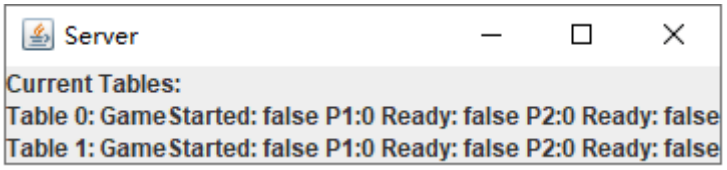

图 16 初始监视信息（Server）

图 17 客户端入口（Client1）

图 18 进入空闲房间后（Client1）（TableID: 0）

图 19 与另一客户端进行匹配后（Client1）（Opponent = 2）

图 20 与另一客户端进行匹配后（Client2）（Table = 0）（Opponent = 1）

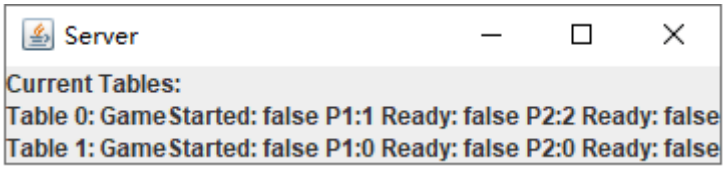

图 21 与另一客户端进行匹配后（Server）

图 22 双方点击准备后（Client1）（Ready = true）（OpntReady = true）

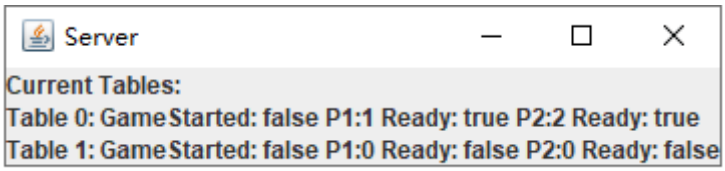

图 23 双方点击准备后（Server）

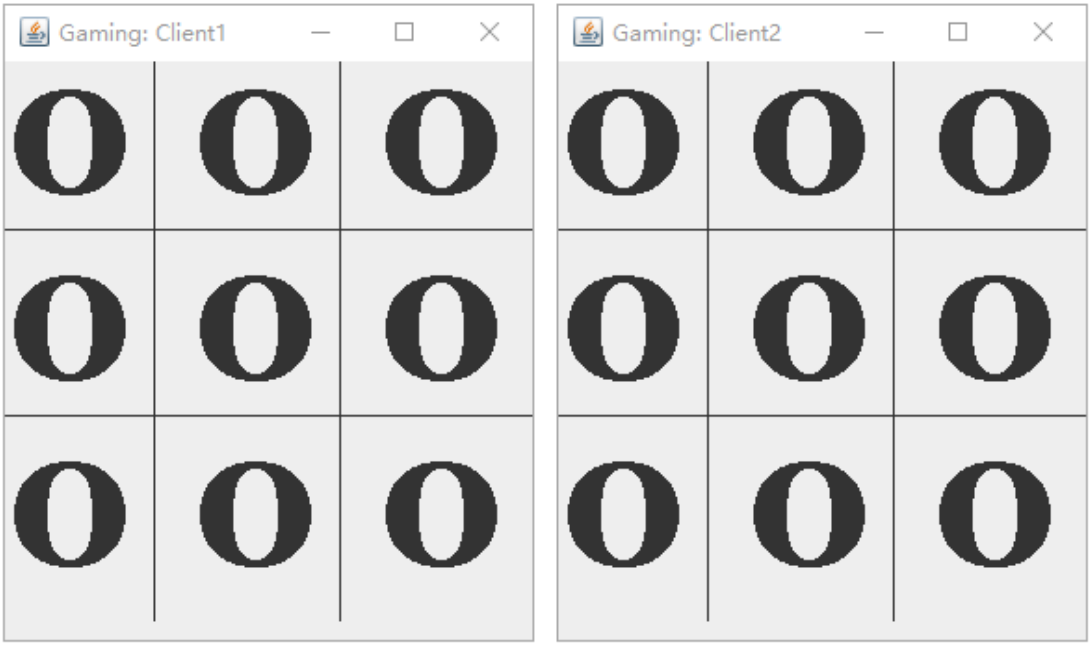

图 24 任何一方点击开始游戏后（Game）

图 25 任何一方点击开始游戏后（Client1）

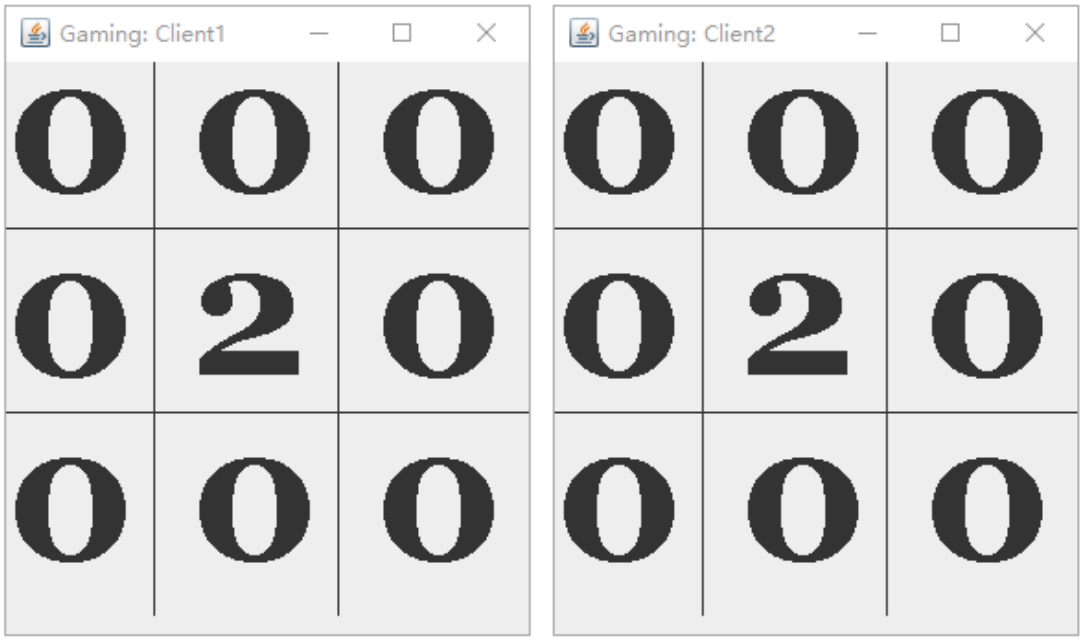

图 26 Client2 先下一步后（Game）

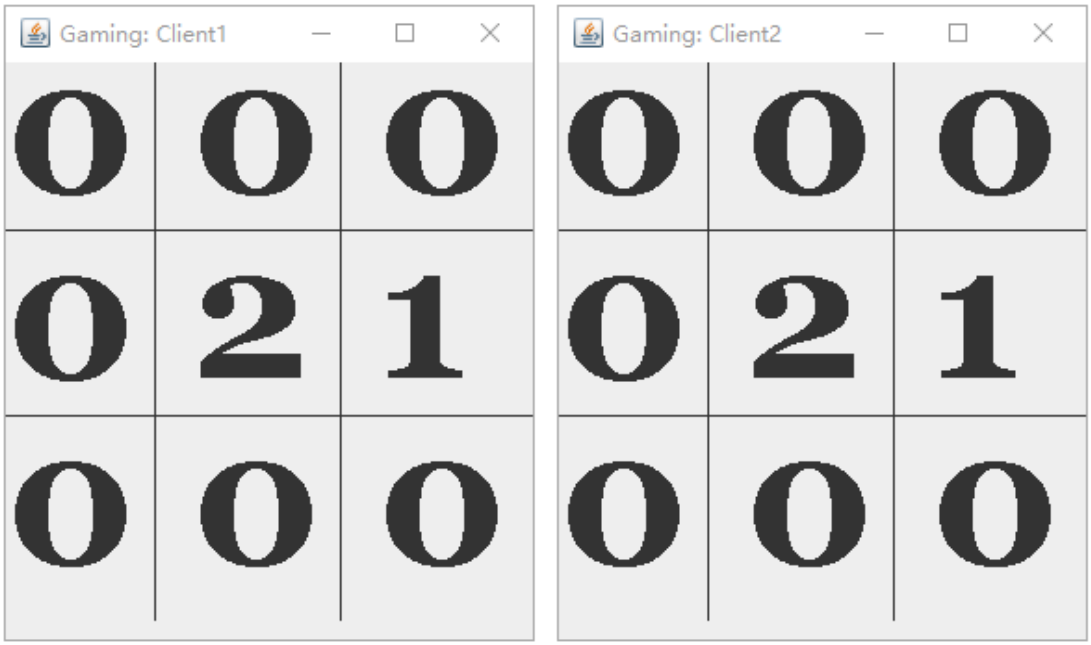

图 27 Client1 回应一步（Game）

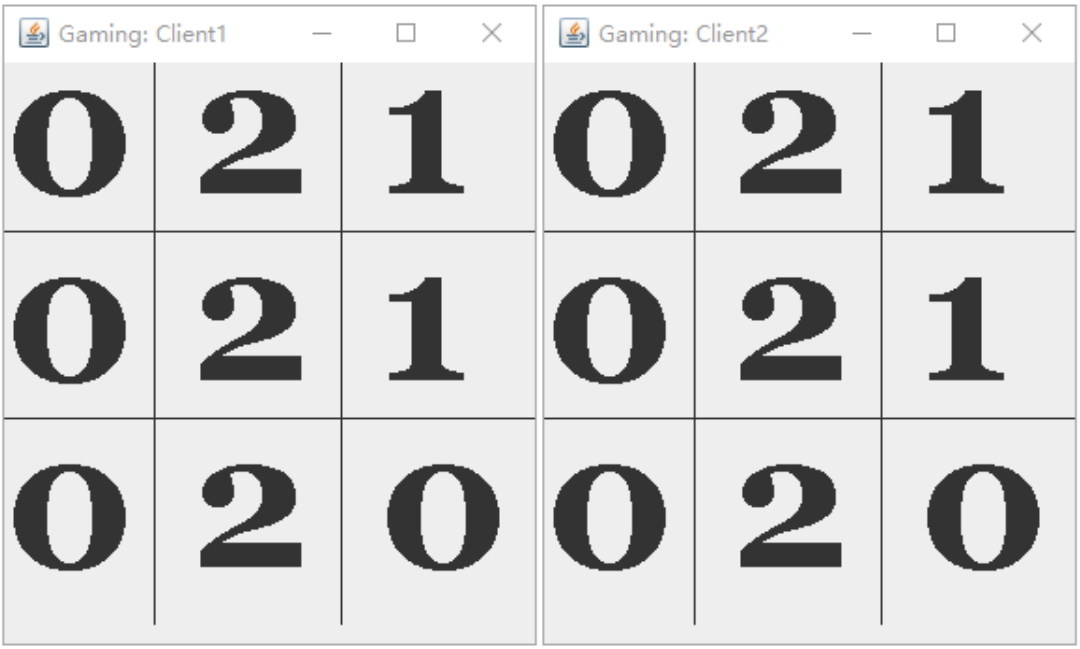

图 28 游戏分出胜负时点（Game）

图 29 游戏分出胜负时点（Client1）

图 30 游戏平局时点（Client1）

图 31 点击消息框后（Client1）

图 32 Client1 退出房间后（Client1）（Table = -1）

图 33 Client1 退出房间后（Client2）（opponent = 0）

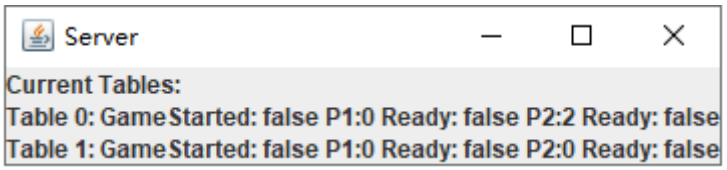

图 34 Client1 退出房间后（Server）

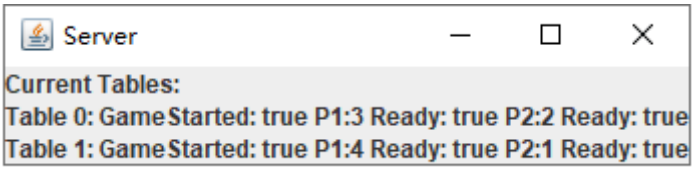

图 35 同时进行两桌游戏（Server）

5 总结

本次大作业编写使我对 Java 编程语言的普适性得到了更深的了解。大多数标准库中的类的使用都很方便，类嵌套、匿名类等机制也使编程逻辑变得非常清晰完善。

实际上我本来想做个更复杂的游戏程序，但是由于大作业在期末时段，时间似乎不太允许，而且井字棋写好之后在此基础上扩展到其他棋类游戏（围棋、五子棋、象棋、翻转棋等）都是很方便的，我觉得再写成其他游戏也不太能展示我对 Java 的理解更加深刻。应该说从 Java 这门课程中收获最大的是这种编程思维，和这种软件开发方式。

我对 Java swing 所做的 GUI 的开发是确实不太熟悉，因此虽然我大作业的代码量与我所知的往届作业差别不大，但 GUI 却相对显得十分寒酸。我的代码的大部分都用在游戏逻辑和多线程处理上了，因此对一些信息我只用了最基础的 JLabel 或 JDialog（本来确实应该更突出一下某些重要信息……）来表示，也没有对服务器端和客户端之间的通信进行一些在服务器端可见的记录。

不过游戏能流畅运行就行，对吧……至少我对我游戏逻辑的处理还是非常有信心的。客户端和服务器端的交互以及随之而来的数据处理问题也确实让我苦战了许久，但最终还是解决掉了。总之，这门课程对 Java 编程的挖掘确实非常有意义，让我从中学到了许多。

参考文献

[1] Java 通过字符串调用类的方法 getMethod() Hopeful Qiang 的博客-CSDN 博客

[2] java 构造函数中启动线程_java – 为什么不在构造函数中启动线程？如何终止？..._weixin_39555415 的博客-CSDN 博客

[3] Java 并发编程实战（3）- 互斥锁 - 李潘 - 博客园 (cnblogs.com)

[4] 关于 Java 互斥锁_WhataNerd 的博客-CSDN 博客_java 互斥锁

[5] Java Swing 如何关闭当前窗口？_Roc-xb 的博客-CSDN 博客_java swing 窗口关闭

[6] Java 并发编程实战（3）- 互斥锁 - 李潘 - 博客园 (cnblogs.com)

[7] Java 中三种常用布局方式_齐天大荒的博客-CSDN 博客

[8] Java 用 JFrame、JPanel、Graphics 绘图案例讲解_大脑补丁的博客-CSDN 博客_jframe 绘图

[9] 简单谈谈 java 中匿名内部类构造函数? - 一只涡流 - 博客园 (cnblogs.com)

[10] JOptionPane 详解_tjk123456 的博客-CSDN 博客_joptionpane