浙江大学计算机学院

Java 程序设计课程报告

2019-2020 学年 秋冬 学期

题目	网络程序之猜数字游戏
学号	3170105860
学生姓名	彭子帆
所在专业	软件工程
所在班级	软工 1701

目　　录

1. 引言

本次开发的是猜数字两人对战网络游戏，可以对 Java 语言中的各项功能有更好的理解和使用，通过具体的程序来加深对 Java 语言的掌握，提高自己的编程水平，为以后的工作打下一定的基础。

1.1 设计目的

猜数字游戏是一个很简单但又有趣的游戏。本文使用 Java 语言编写一个具有图形界面、运用 socket 编程、运用 jdbc 的程序具体功能如下：

(1) 初始化服务端，如数据库中不存在表则创建表
(2) 点击开始，查看规则，退出
(3) 匹配对手，匹配到对手客户端生成图形化界面
(4) 随机生成三个随机数，二者进行猜测，猜对者获得分数，最终三局两胜
(5) 其中每次的数字记录都会加入到数据库表中

1.2 设计说明

本程序采用 Java 程序设计语言，在 IntelliJ IDEA 平台下编辑、编译与调试。并通过建立 Maven 工程进行工程包管理。具体程序由我个人开发而成。使用了数据库，socket，多线程，swing 框架设计 gui，工作时间轴如表 1 所示:

时间	完成工作
12.30	学习 swing、数据库以及 socket 编程
1.11	客户端编写
1.13	服务器编写
1.15	文档撰写

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) 用户打开客户端

(2) 查看规则

(3) 退出

(4) 开始对战

(5) 数据库储存历史数字信息

(6) 根据游戏规则进行游戏

程序的总体功能如图 1 所示：

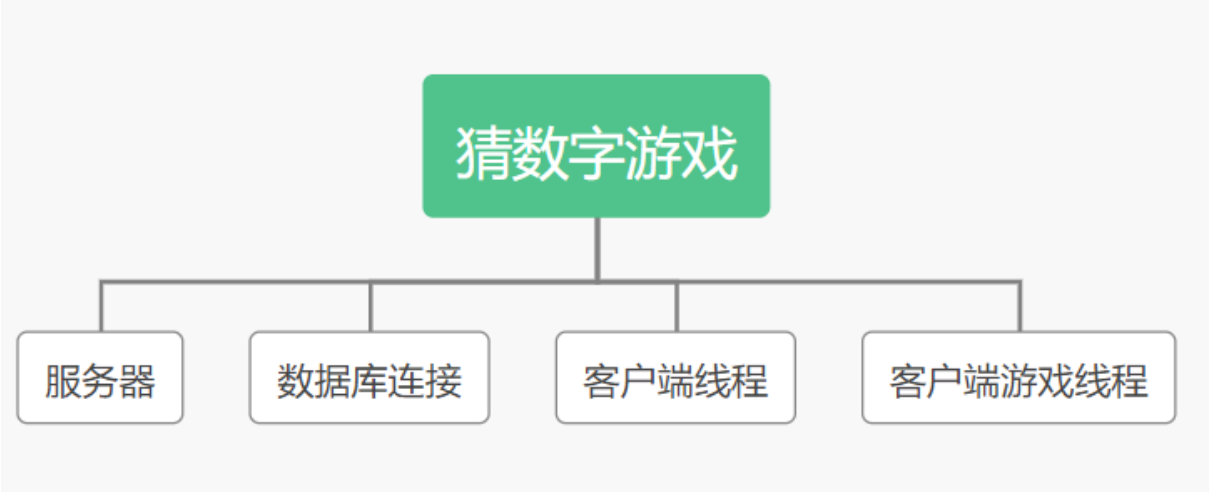


图 1 总体功能图

2.2 流程图设计

程序总体流程如图 2 所示：

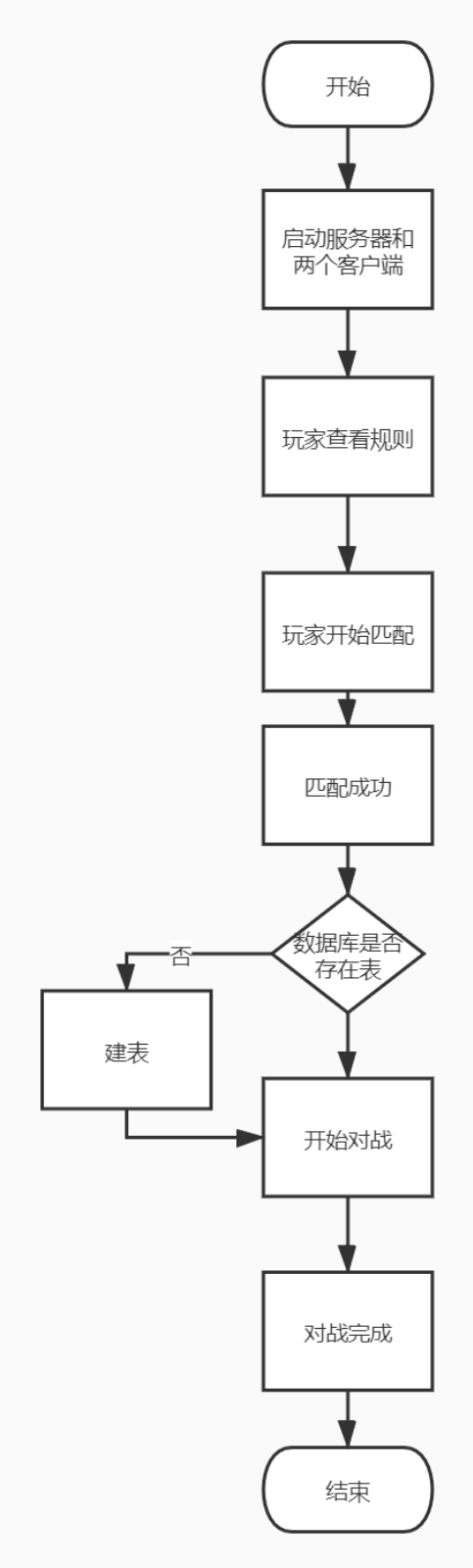
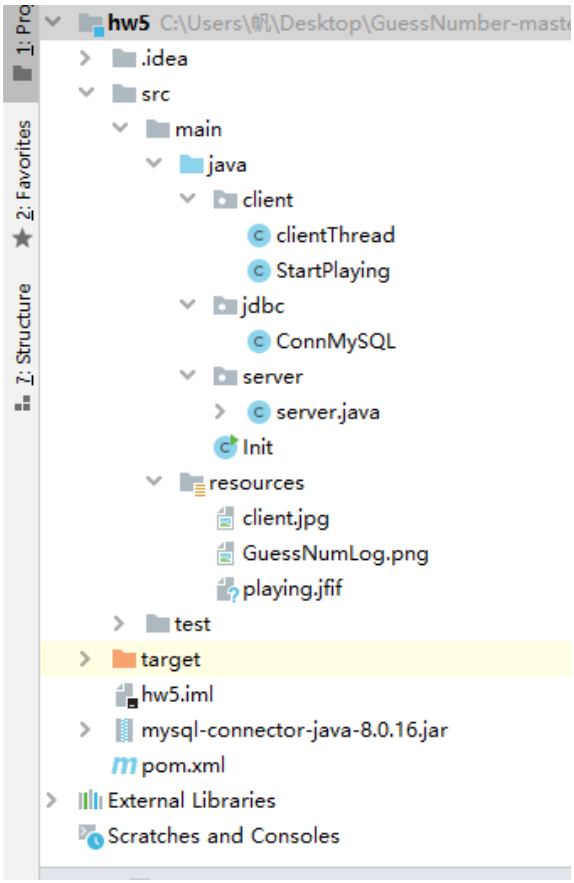


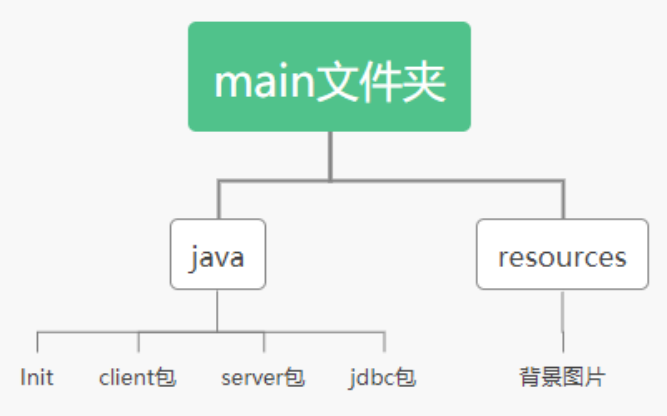
图 2 总体流程图

3. 详细设计

设计过程中，主要分为客户端、服务器和数据库管理，文件结构如下：



其为一个在 IDEA 下建立的 maven 工程，主类中拥有 java 文件和 resources，resources 放入背景图片们，类中分为三个包，客户端，数据库，服务器，另外最外层有一个初始化运行的类，用来实例化客户端、数据库和服务器，启动。下面是文件、类的结构：

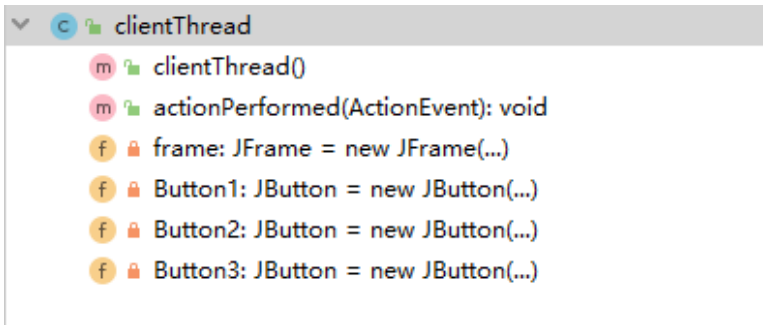


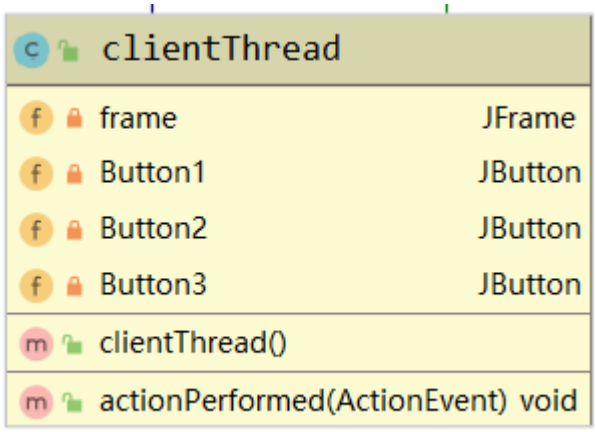
3.1 客户端线程的设计

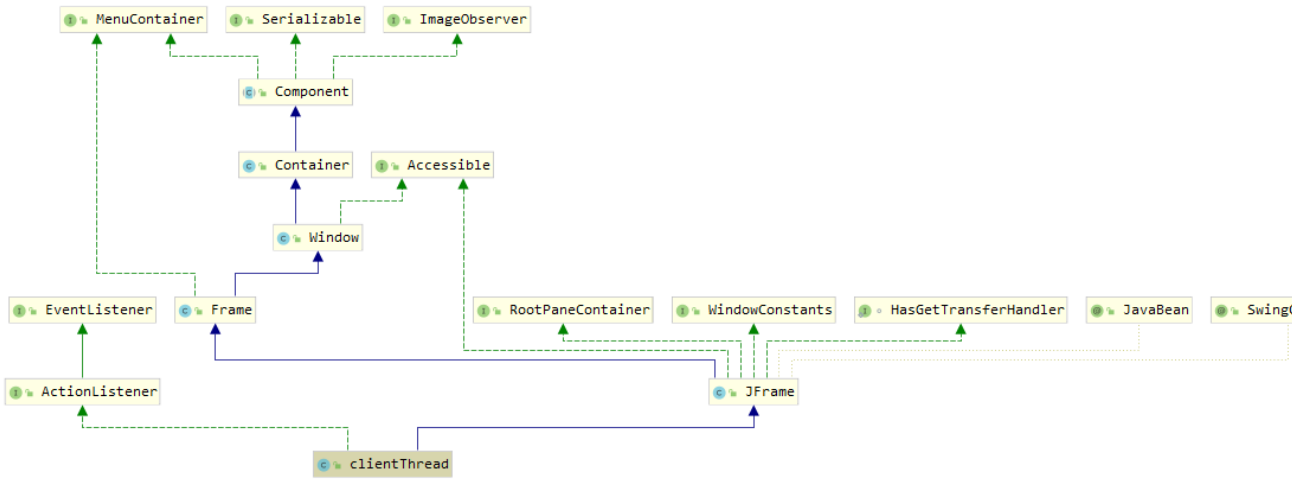
开启客户端后，客户端回首先连接地址 127.0.0.1，端口连接为 5860。连接成功后会跳出客户端界面。

客户端包下有两个类：clientThread, StartPlaying

其中 clientThread 类和函数、变量以及关系分别如下：







客户端类实现了客户端线程，生成一个客户端。首先调用 JFrame 类，放入所需组件，并监听事件，如果用户能够点击三个 button 进行选择开始、规则、退出。

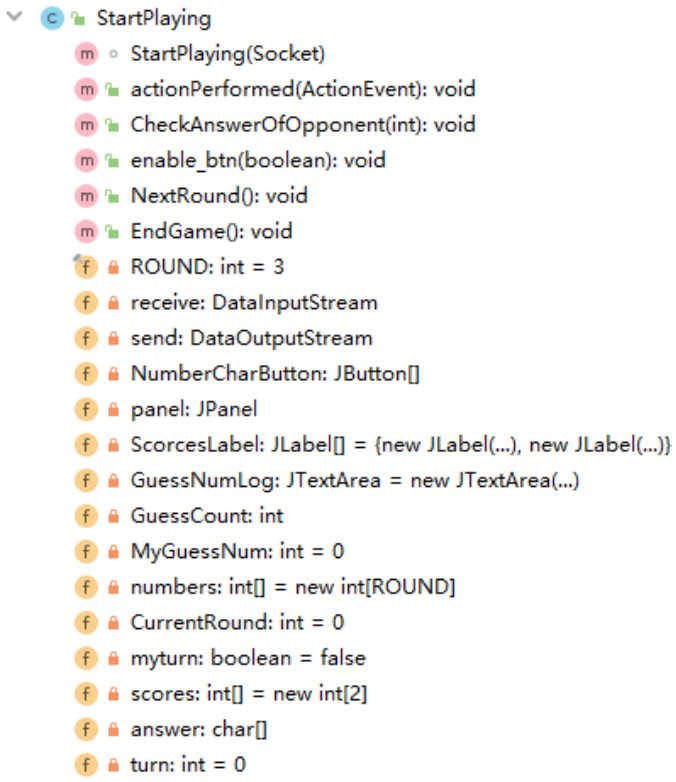
客户端会进行监听，当玩家开始匹配后则会按钮变为不可点击，等待另一位玩家匹配。如果两个玩家都连入其中，则进入游戏界面开始游戏。

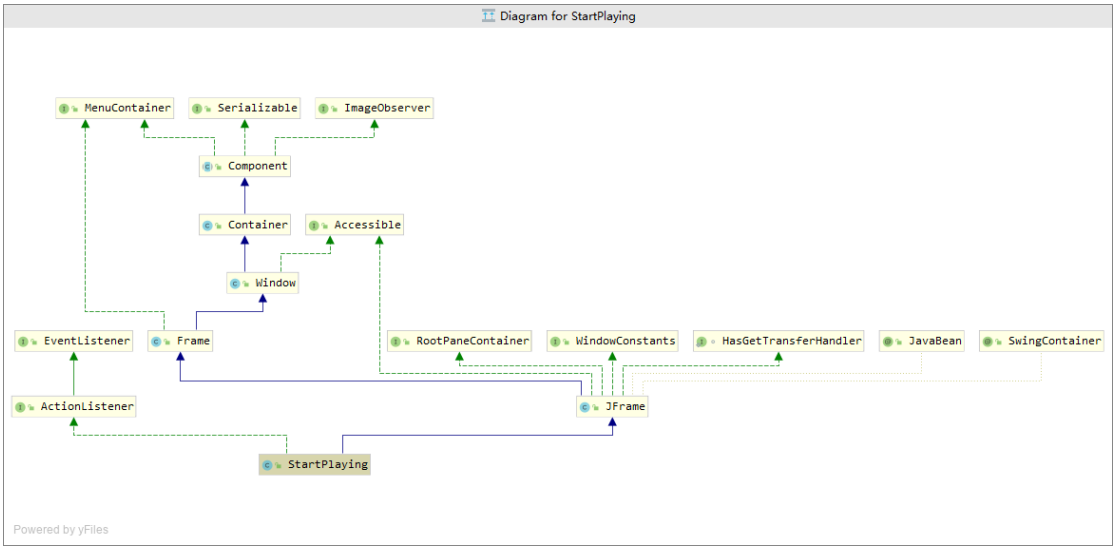
关键代码如下：

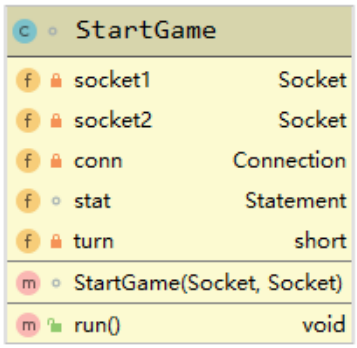
```
1.  Socket connection = new Socket("127.0.0.1", 5860);
2.  new Thread(()-> {
3.      try {
4.          DataInputStream signal = new DataInputStream(connection.getInputStre
    am());
5.          int match = signal.readInt();
6.          if (match == 0) {
7.              new StartPlaying(connection);
8.          }
9.      } catch (IOException ex) {
10.         ex.printStackTrace();
11.     }
12. }).start();
```

游戏客户端存在于 StartPlaying 类中，因此首先进行初始化各个组件的位置等等，收到服务器的初始三个数字后便可以正式开始，玩家一先开始猜。

StartPlaying 类和函数、变量以及关系分别如下：







以下是 UML 图中有关的主要数据和方法的详细说明：

(1) 成员变量

① ROUND 定义轮数。

② receive 和 send 获取数据输入流

③ myturn 是否是我的回合

④ CurrentRound 当前第几轮

⑤ player 和 scores 分别记录玩家和玩家得分

(2) 方法

①
```
/**
 * 用来判断你是否猜对并且加入自己的猜测到记录中
 * 若猜对则获得分数并开始下一局
 * @param event 事件、这里就是鼠标点击
 */
public void actionPerformed(ActionEvent event)
```

②
```
/**
 *用来判断对手是否猜测的正确，并写入记录
 * 如果正确结束并加分
 * @param GuessNum 对手猜的数字
 */
public void CheckAnswerOfOpponent(int GuessNum)
```
③
```
/**
 * 用来将所有按钮 enable 或者 disable,取决于参数
 * @param my_turn 为true 时 enable，为false 则 disable
 */
public void enable_btn(boolean my_turn)
```
④
```
/**
 *用来判断对手是否猜测的正确，并写入记录
 * 如果正确结束并加分
 * @param GuessNum 对手猜的数字
 */
public void CheckAnswerOfOpponent(int GuessNum)
```
关键代码如下：

```
1.  //判断谁是第一个
2.  int GuessNum = receive.readInt();
3.  //如果收取结果不正确或者为-1 则轮到我了修改 myturn=1
4.  //由于服务端初始化时确定第一个游戏时发送-1，为 0 时不是第一个游戏
5.  if(GuessNum==0){
6.      myturn = false;
7.  }
8.  else if (GuessNum==-1) {
9.      myturn = true;
10.     enable_btn(myturn);
11. }
12. //正式开始游戏，获取服务器传来的另一个游戏线程的消息
13. while (true){
14.     //获取刚刚另一个线程猜的结果通过服务端发来的结果
15.     GuessNum = receive.readInt();
16.     //如果不相等则轮到我进行猜测
17.     if (GuessNum!=numbers[CurrentRound]) {
18.         myturn = true;//将我的回合改成 true
19.         enable_btn(myturn);
20.     }
21.     CheckAnswerOfOpponent(GuessNum);
22. }
```

3.2 服务器的设计

Server 包中的类和类中的变量以及类图如下：

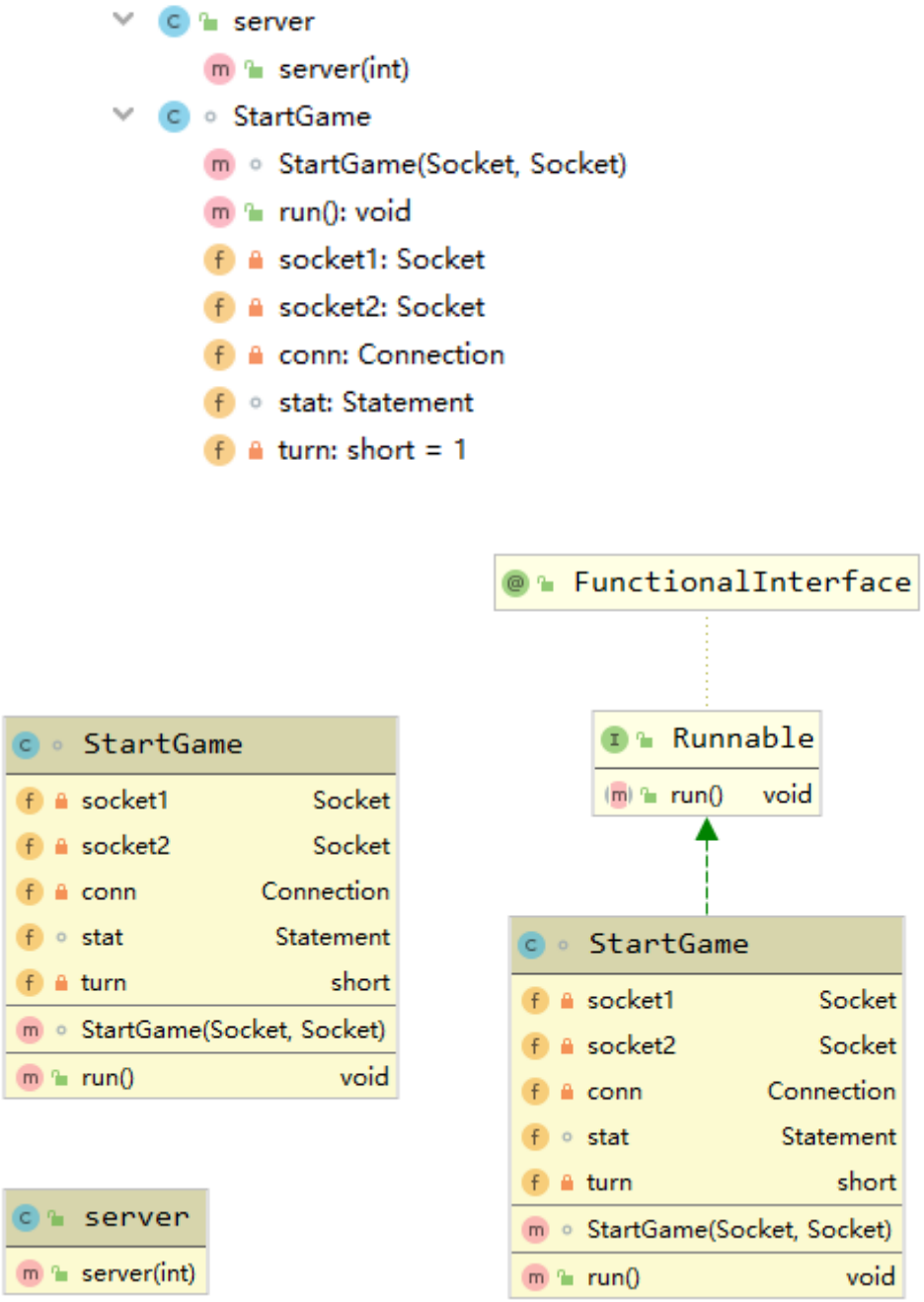


图 4 Server **类的 UML 图**

关键代码如下：

```
1.  /**
2.    * 用来启动服务端，启动后提示已启动，等待两个客户端线程来连接。连接完成后正式开始游戏
3.    */
4.   public server(int PORT) {
5.       new Thread(() -> {
6.           Socket socket1 = null;
7.           Socket socket2 = null;
8.           try {
9.               ServerSocket serverSocket = new ServerSocket(PORT);
10.              System.out.println("服务器已启动！\n");
11.              while (true) {
12.                  socket1 = serverSocket.accept();
13.                  System.out.println("玩家 1 正在匹配！");
14.                  socket2 = serverSocket.accept();
15.                  System.out.println("玩家 2 正在匹配！\n 开始对局！");
16.                  new Thread(new StartGame(socket1, socket2)).start();
17.              }
18.          } catch (IOException | SQLException e) {
19.              // TODO Auto-generated catch block
20.              e.printStackTrace();
21.          } finally {
22.              try {
23.                  socket1.close();
24.                  socket2.close();
25.              } catch (IOException e) {
26.                  // TODO Auto-generated catch block
27.                  e.printStackTrace();
28.              }
29.          }
30.      }).start();
31.  }
```

以下是 UML 图中有关数据和方法的详细说明：

(1) 成员变量

```
private Socket socket1;
private Socket socket2;
private Connection conn;
Statement stat;
private short turn = 1;//当前哪个线程在行动
```

(2) 方法

①
```
/**
 * 构造函数根据两个游戏客户端的socket
 * @param socket1 第一个客户端的socket
 * @param socket2 第二个客户端的socket
 * @throws SQLException
 */
StartGame(Socket socket1, Socket socket2) throws
SQLException
```

②
```
/**
 * 重写接口内的方法
 */
@Override
public void run()
```

3.3 数据库管理的设计

本次客户端管理属性以及方法如下：

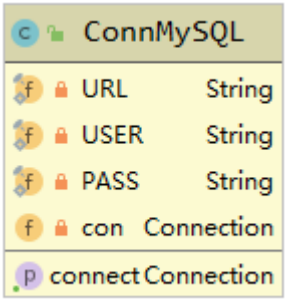


图 5 ConnMySQL 类的 UML 图

其中变量 URL 为连接数据所需 url，user 为用户名，pass 为密码，连接后的连接消息。方法为连接，返回连接值。

成员变量：

```
//需要更改对应的mysql 的用户名、密码以及创建响应数据库名为hw5
private static final  String
URL="jdbc:mysql://localhost:3306/hw5?useUnicode=true&character
Encoding=utf8&serverTimezone=GMT%2B8&useSSL=false";//链接的
mysql
private static final String USER = "root";
private static final String PASS = "123456";
```

关键代码截图如下：

```
1.  private Connection con = null;
2.  public Connection getConnect() {
3.      try {
4.          //1.加载驱动
5.          Class.forName("com.mysql.cj.jdbc.Driver");
6.          System.out.println("数据库驱动加载成功！");
7.          //2.创建连接
8.          con = DriverManager.getConnection(URL, USER, PASS);
9.          System.out.println("连接成功！");
10.     } catch (ClassNotFoundException e) {
11.         // TODO Auto-generated catch block
12.         e.printStackTrace();
13.     } catch (SQLException e) {
14.         // TODO Auto-generated catch block
15.         e.printStackTrace();
16.     }
17.     return con;
18. }
```

4. 测试与运行

4.1 程序测试

在程序代码基本完成后，经过不断的调试与修改，最后测试本次所设计的客户端、服务器以及数据库管理都能够正常运行，总的来说本次设计在功能上已经基本达到要求，其他细节方面有待以后完善。

4.2 程序运行

运行 Init 类后即会运行服务器以及两个线程：

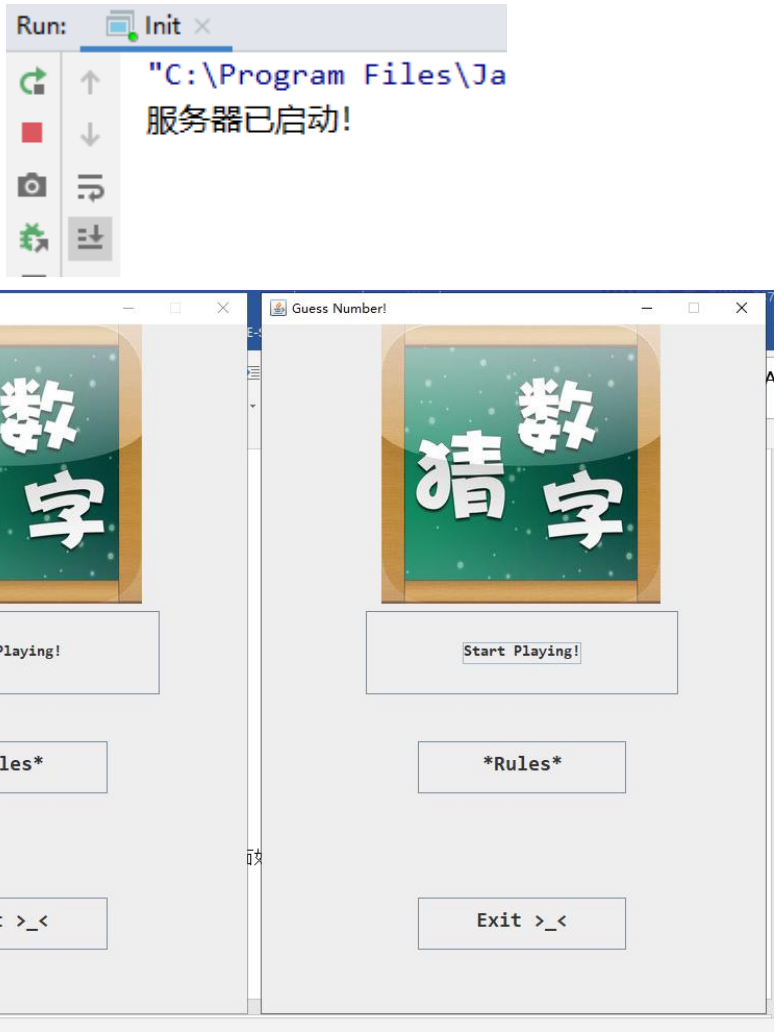
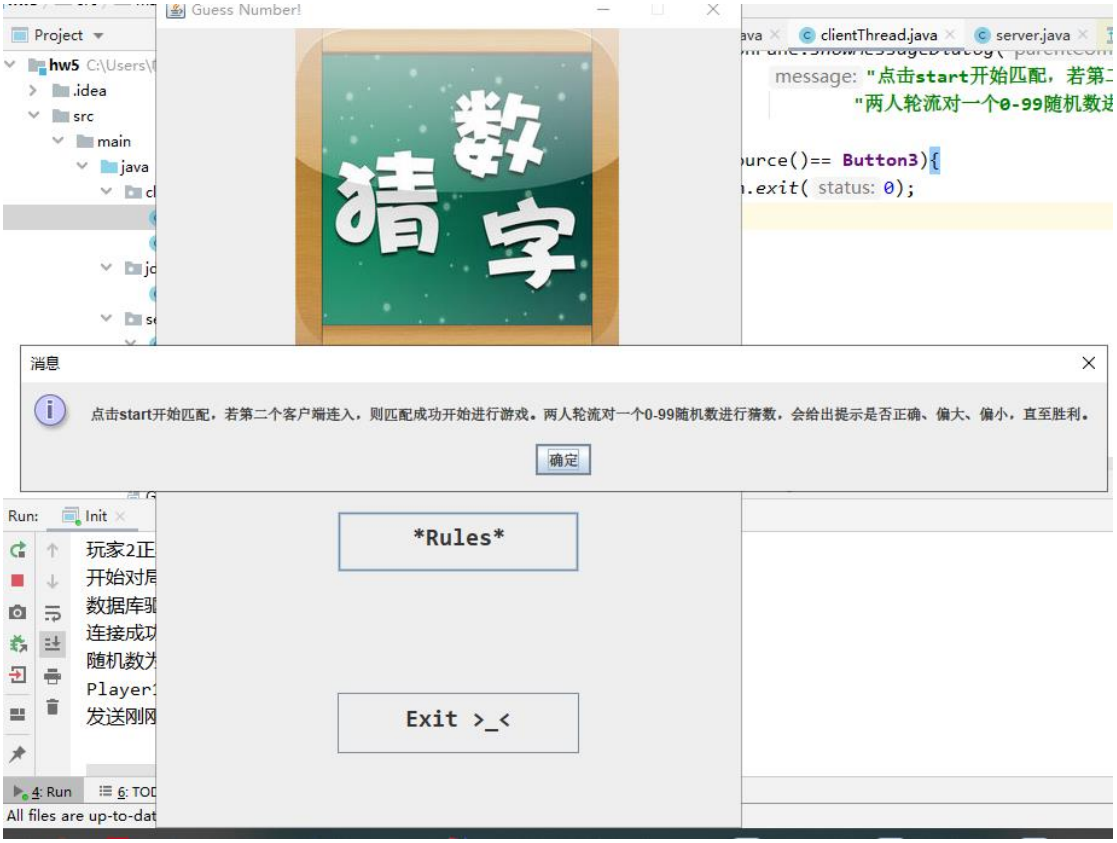
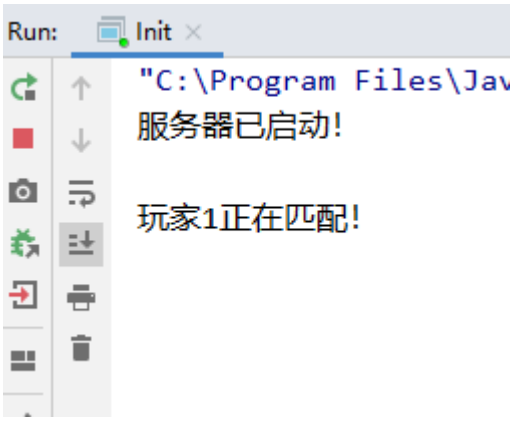


图 8

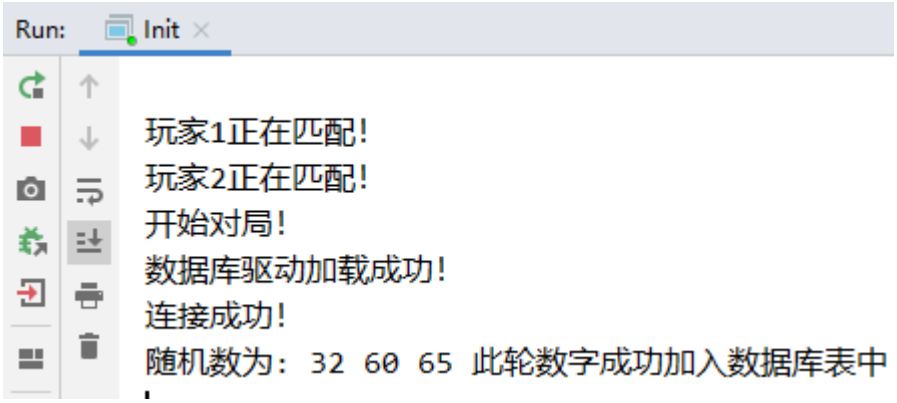
当点击规则后：



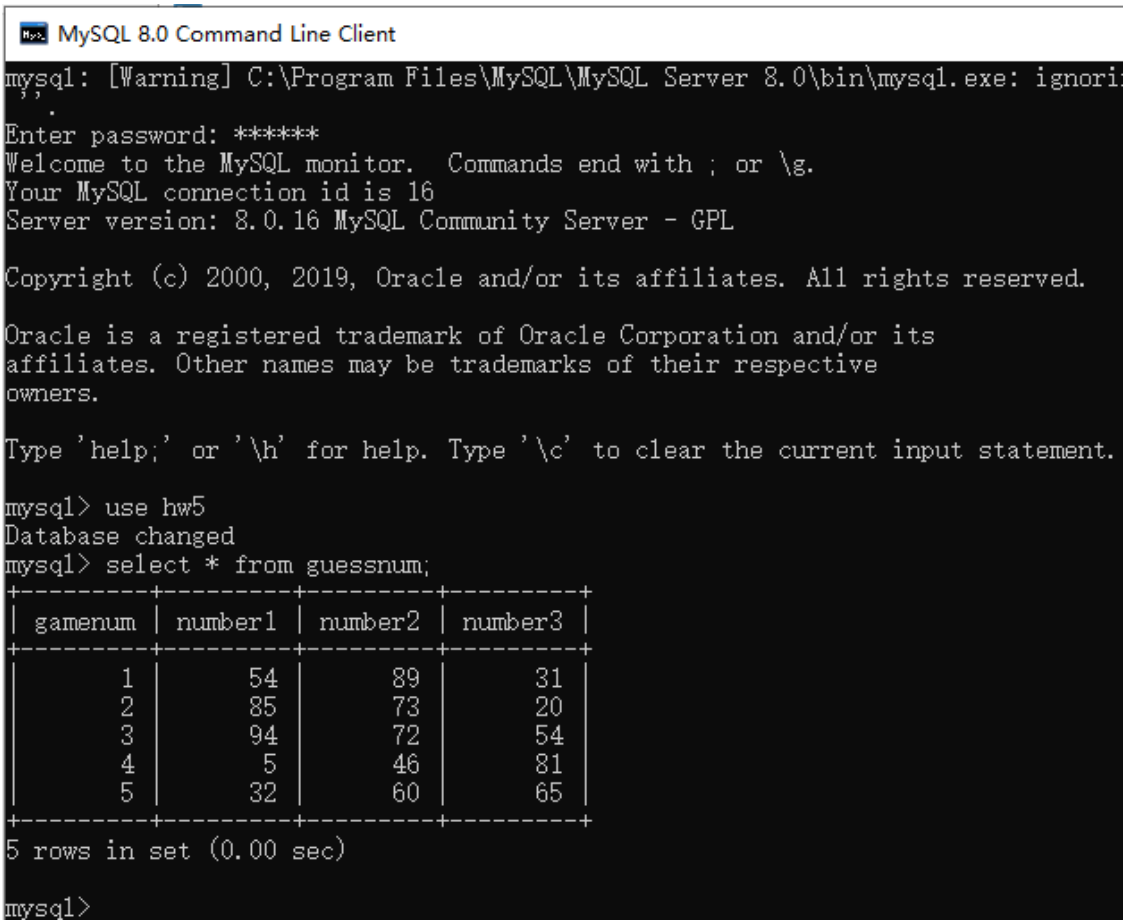
其中一个进行匹配时的服务器和客户端：



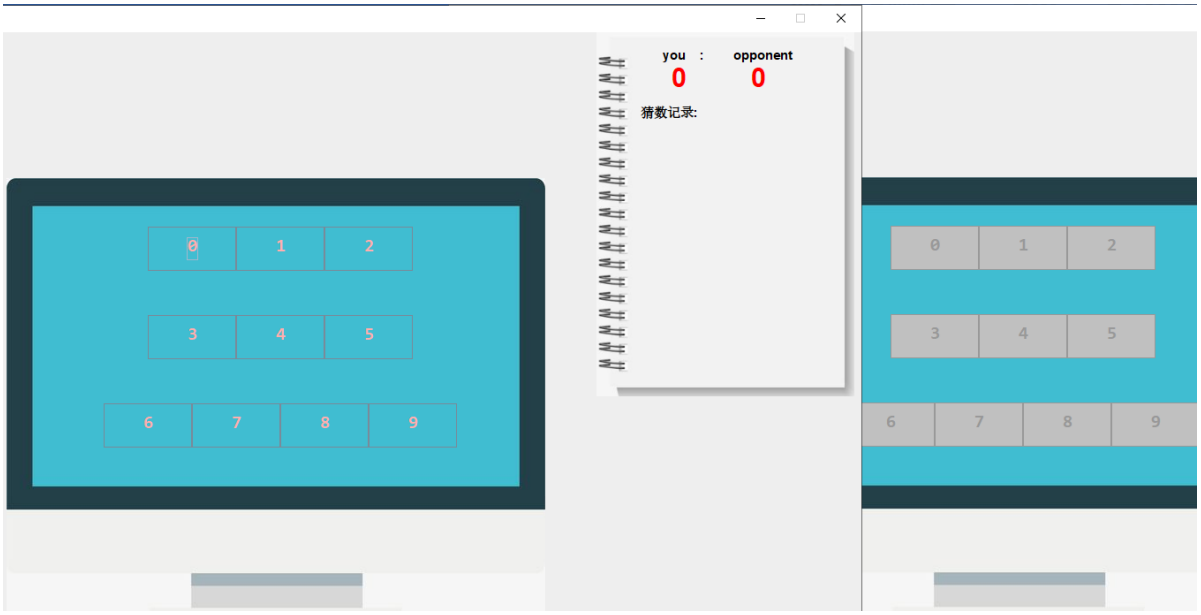
两个都点击之后弹出：（因为已经创建过表所以不需要在创建表）



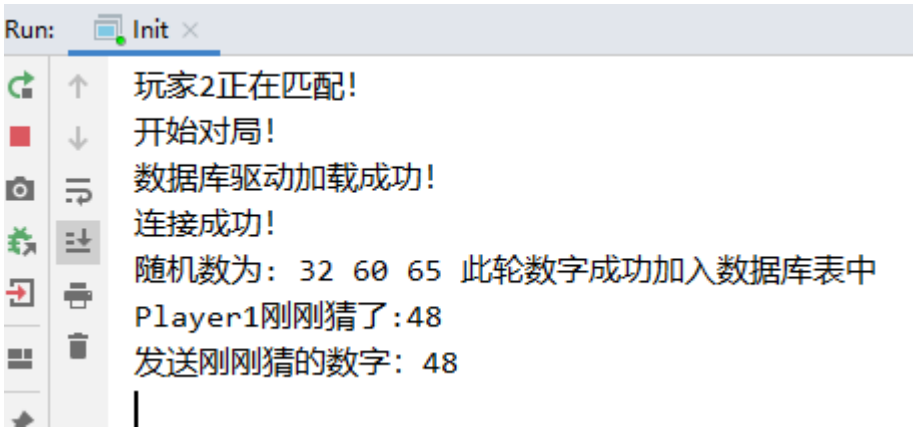
数据库中添加的结果如下：



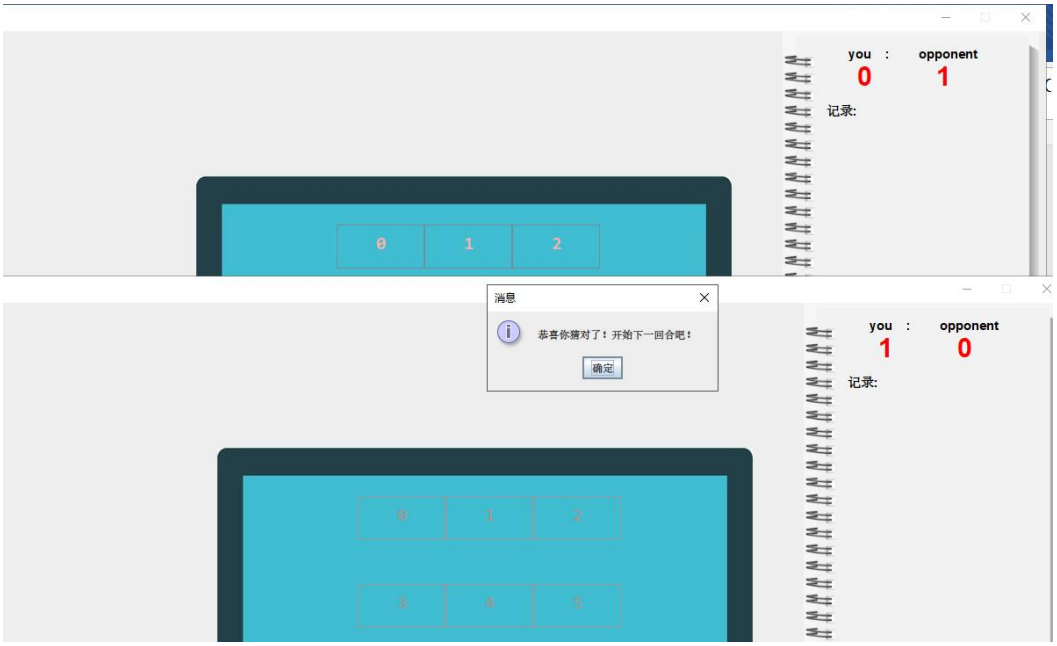
两个玩家分别可以进行猜数字以及等待对方：



当一个玩家点击两次生成一个两位数之后发送到服务器，然后另一个玩家开始：



当对手答对的时候则加分并显示下一回合：



5. 总结

通过本次的作业和学习，我对于 Java 的 gui、socket 以及 jdbc 编程有了一定的认识，也学会了 maven 包管理工具和多线程处理机制，本学期我们也学习了操作系统和计算机网络，但这两门课都是用最基本的 C 语言进行编程的，所以根据操作系统的多线程和比起操作系统的多线程编程，Java 实现多线程的方式简单很多，并且 socket 编程中 Java 也更加的方便以及更好地进行消息传递、转发。有关于图形界面、同步和互斥，Java 封装过的类中都已经做好了这些操作，因此只需要我来进行简单地操作即可。

这个小游戏是第二次引用外部 jar 包，然后对各个类进行分包管理，也是我第二次使用 maven 包管理工具，并且第一次进行 SOCKET 编程以及在 JAVA 上进行多线程编程必然在编程过程中也遇到很多问题，有些问题比较简单但就不知道错在哪里，有些大问题如何解决，后来经过自己的不断调试，这个小游戏的客户端以及服务器最终完成。

通过该课程设计，全面系统的理解了程序构造的一般原理和基本实现方法。把死板的课本知识变得生动有趣，激发了学习的积极性。把学过的 Java 的知识强化，能够把课堂上学的知识通过自己设计的程序表示出来，加深了对理论知识的理解。现在通过自己动手做实验，从实践上认识了 Java 外部包是如何引入的，package 是如何管理的，GUI 是如何进行设计放入的，多线程是如何运行的，服务器如何接受和转发消息的。课程设计中程序比较复杂，在调试时应该反复耐心地调试方能收获颇丰。