

杭州电子科技大学

《网络编程》

结课报告

课 题	斗地主
学 院	计算机学院
专 业	计算机科学与技术
班 级	17052318
姓 名	任庆（17220624）
指导教师	吴永胜
完成日期	2020 年 5 月

目录

第 1 章	绪论	3
第 2 章	系统设计	3
2.1	网络通信	3
2.1.1	传输层协议选择: TCP	3
2.1.2	通信草图.....	3
2.2	同步	4
2.2.1	游戏五大阶段.....	4
2.2.2	流程图.....	4
2.3	语法语义分析.....	4
2.3.1	范围.....	4
2.3.2	语法.....	4
2.3.3	语义.....	4
第 3 章	编码调试	5
3.1	TCP 通信	5
3.1.1	客户端.....	5
3.1.2	服务器.....	5
3.1.3	实现.....	6
3.2	同步	7
3.2.1	阶段定义.....	7
3.2.2	信号量.....	7
3.2.3	主服务器.....	7
3.2.4	服务器接收端.....	8
3.2.5	主客户端.....	8
3.2.6	客户端接收器.....	8
3.2.7	同步序列码.....	9
3.3	语法语义	9
3.3.1	卡牌定义.....	9
3.3.2	出牌类型.....	9
3.3.3	语法检查.....	10
3.3.4	比较大小.....	10
3.4	打印卡牌	10
3.4.1	原理.....	10
3.4.2	实现效果.....	11
第 4 章	测试运行	11
4.1	Waiting	11
4.2	Preparation	12
4.3	Selecting.....	12
4.4	Running.....	13
4.5	Finish	13
第 5 章	体会总结	14
参考文献	14

第1章 绪论

因为这门课是网络编程嘛，当然要完成的东西得在多台机器上完成，围绕着通信这个核心基础功能展开，因为斗地主小时候玩的比较多，对里面的规则非常的熟悉，而且我感觉这个游戏刚好是我能够做的，感觉做起来会比较有意思，所以选了这个游戏来进行开发。正巧这个学期选了一门 JAVA 语言，也为了更加深入的了解一下这门语言，所以决定选这门语言作为这个游戏的开发语言。

第2章 系统设计

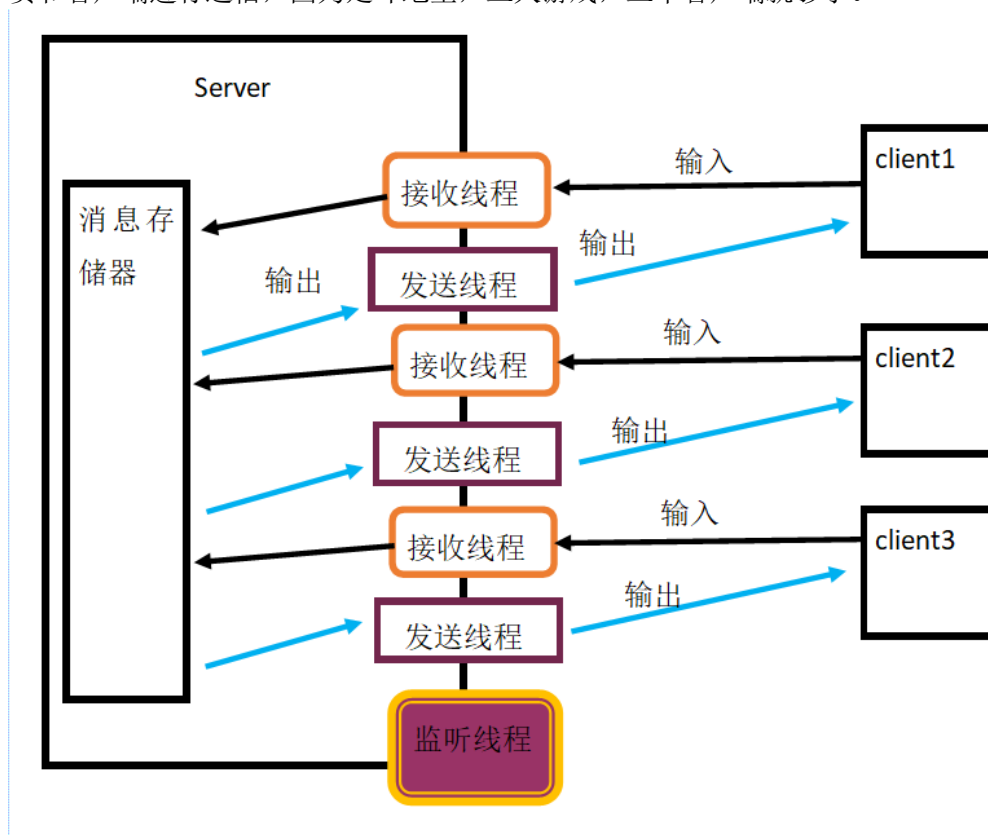
2.1 网络通信

2.1.1 传输层协议选择：TCP

原因：因为这个游戏数据传输比较严谨，有一点差错系统可能就瘫痪了，所以选择 TCP 来作为通信基础。

2.1.2 通信草图

服务器得有一个监听线程，若监听到一个客户端，就创建一个接收线程和一个发送线程负责和客户端进行通信，因为是斗地主，三人游戏，三个客户端就够了。

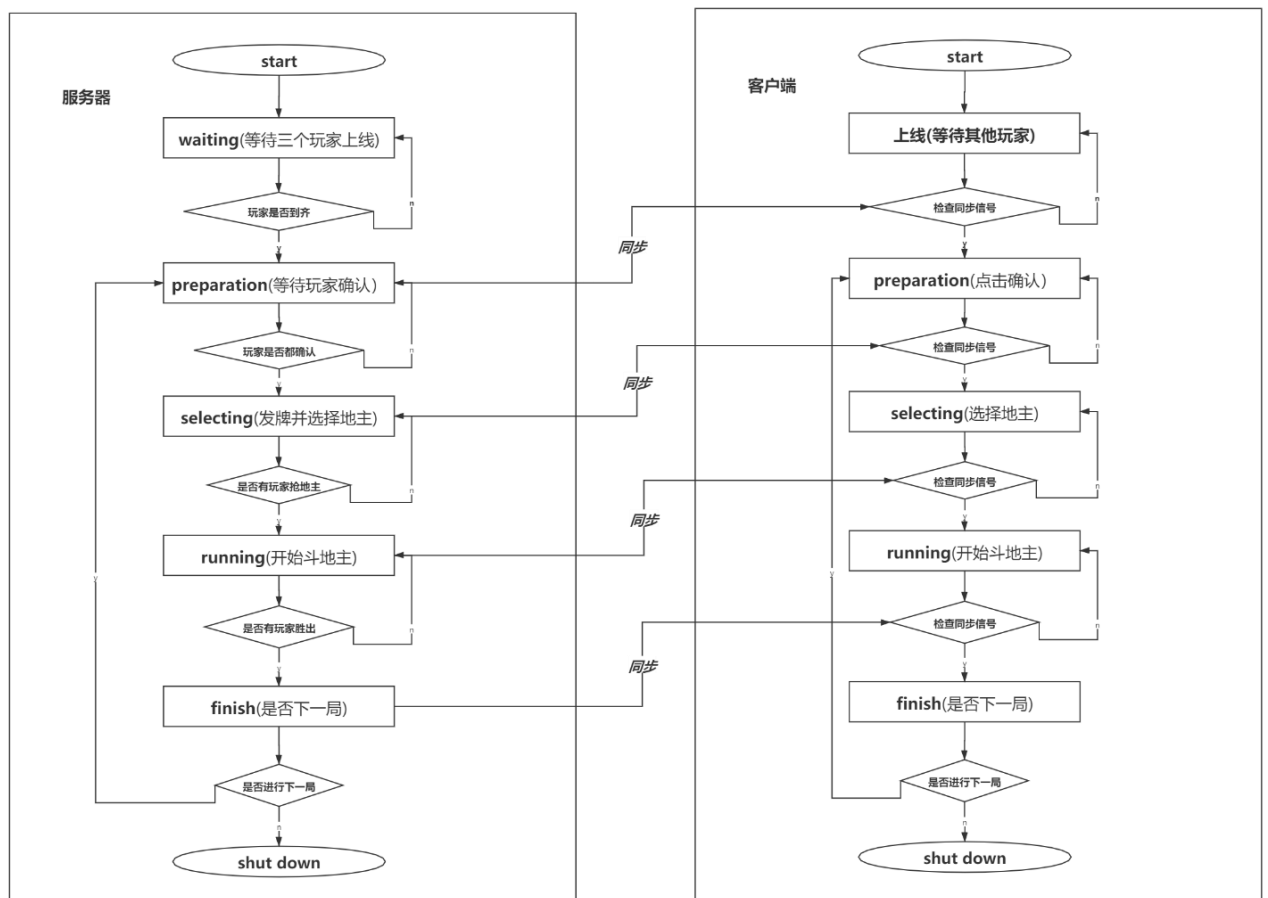


2.2 同步

2.2.1 游戏五大阶段

- **waiting**(等待三个玩家上线)
- **preparation**(等待三个玩家确认开始游戏)
- **selecting**(发牌并选择地主)
- **running**(开始斗地主)
- **finish**(是否进入下一局)

2.2.2 流程图



2.3 语法规则分析

2.3.1 范围

打出的牌是否有规定的其他符号，是否是包括在手牌里。

2.3.2 语法

斗地主也有自己的一套语法规则，不能乱出，如顺子，飞机，连队。

2.3.3 语义

语法正确还不够，还要有大小的比较，如单牌的大小为大王>小王>2>A>K>Q>J>10>9>8>7>6>5>4>3。

第3章 编码调试

3.1 TCP 通信

3.1.1 客户端

- 输入 IP 和端口，并创建套接字。

```
try {
    ip = args[0];
    port = Integer.parseInt(args[1]);
} catch (Exception e) {
    ip = "localhost";
    port = 6666;
}

socket = new Socket(InetAddress.getByName(ip), port);
```

- 建一个写的流负责发送数据，输入 exit 则退出程序。

```
String text;
while (!(text = sc.nextLine()).equals("exit")) {
    switch (gameStatus) {
        case WAITING: waiting(text); break;
        case PREPARATION: preparation(text); break;
    }
}
```

- 另创建一个分支线程负责接收数据，若收到数据则打印输出。

```
private static void sendMessage(String text) throws IOException {
    writer.write(text);
    writer.newLine();
    writer.flush();
}
```

3.1.2 服务器

- 首先在主界面创建服务器套接字，并开辟一个监听线程，负责监听客户端的连接。

```
while (true) { // 阻塞监听，若成功则创建套接字
    Socket accept = socket.accept();
    socketList.add(accept);
    String name = null;
```

- 监听线程核心如下。

```
for (int i = 0; i < 3; i++) {
    if (!Server.sendBufferStatus.get(sendBuffer[i])){
        name="玩家"+i;
        new SendThread(accept,sendBuffer[i]).start();
        Server.sendBufferStatus.put(sendBuffer[i],true);
        break;
    }
}
```

- 接收客户端消息线程码如下。

```
Server.reserveBuffer.append(name).append(": ").append(s);
synchronized (Server.reserveBuffer){
    Server.reserveBuffer.notify();
}
```

- 当加入或结束后发一条系统提示。

```
synchronized (stringBuffer) {
    stringBuffer.append(msg);
    stringBuffer.notify();
}
```

- 接收的消息送给主线程，主线程遍历每个套接字，分别发送。

```
//发送广播消息
private static void sendBoardMessage(String msg){
    try {
        Thread.sleep(100); //睡眠0.2s让服务器有时间发
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    for (StringBuffer stringBuffer : sendBuffer) {
        synchronized (stringBuffer) {
            stringBuffer.append(msg);
            stringBuffer.notify();
        }
    }
}
```

3.1.3 实现

```
欢迎您登录游戏！，输入< exit >退出
系统提示： 玩家1加入群聊      ( 当前人数：2)
系统提示： 玩家2加入群聊      ( 当前人数：3)
系统提示： 所有玩家以进入，输入< yes >确认开始游戏
```

3.2 同步

3.2.1 阶段定义

- 我用枚举类型定义了游戏的五大阶段。

```
public enum GameStatus {  
    WAITING, /* 等待玩家进入游戏 */  
    PREPARATION, /* 等待玩家确认开始游戏 */  
    SELECTING, /* 发牌阶段并选择地主 */  
    RUNNING, /* 游戏运行阶段 */  
    FINISH /* 游戏结束 */  
}
```

3.2.2 信号量

- 在操作系统中信号量较 P/V，也叫 wait/signal，JAVA 中是用 wait()/notify(),应该说的一个意思在发送线程中，发完一条消息 wait 就阻塞。

```
while (true){  
    synchronized (message) {  
        writer.write(String.valueOf(message));  
        writer.newLine();  
        writer.flush();  
        message.delete(0,message.length());  
        message.wait();//阻塞  
    }  
}
```

- 若要发消息，就用 notify () 唤醒发送线程。

```
for (StringBuffer stringBuffer : sendBuffer) {  
    synchronized (stringBuffer) {  
        stringBuffer.append(msg);  
        stringBuffer.notify();  
    }  
}
```

- wait(),和 notify()都要用 synchronize () 包住，表示线程安全的，也就是同步的意思。

3.2.3 主服务器

- 服务器主要分为五片代码，也就是服务器的五个阶段，把这五个方法执行完，那一轮游戏也就结束了

```
private static void Scheduler() throws Exception {
    waiting();
    preparation();
    selecting();
    running();
    finish();
}
```

3.2.4 服务器接收端

- 服务器接收器根据当前游戏阶段，把接收的数据进行不同的处理，然后再交给主服务器。

```
while ((s=reader.readLine())!=null){
    switch (Server.gameStatus){ //当前游戏的状态
        case WAITING: waiting(s);break ;
        case PREPARATION: preparation(s);break ;
        case SELECTING: selecting(s); break ;
        case RUNNING: running(s);break ;
        case FINISH: finish(s);break ;
    }
}
```

3.2.5 主客户端

- 客户端始终接收用户的输入，根据同的阶段，判断用户输入是否合法，在判断是否发给服务器。

```
while (!(text=sc.nextLine()).equals("exit")){
    switch (gameStatus){
        case WAITING:waiting(text);break;
        case PREPARATION:preparation(text);break;
        case SELECTING:selecting(text);break;
        case RUNNING:running(text);break;
        case FINISH:finish(text);break;
    }
}
```

3.2.6 客户端接收器

- 客户端接收器先判断服务器发来的是否为同步序列码，若是则更新客户端游戏状态。


```

if (s.length()==Client.SYN_STATUS.length()+1){ //同步序列码
    if (s.substring(0,s.length()-1).equals(Client.SYN_STATUS))
    {
        Client.gameStatus=GameStatus.values()[s.charAt(s.length-1)];
        break;
    }
}
switch (Client.gameStatus){

```

- 若不是则根据当前状态对发来的数据进行处理。

```

switch (Client.gameStatus){
    case SELECTING:selecting(s);break;
    case RUNNING:running(s);break;
    default: System.out.println(s);break;
}

```

3.2.7 同步序列码

- 服务器和客户端都有一些同步序列码，客户端来判断服务器发来的的是什么类型的数据。

```

public static final String SYN_STATUS = "dfd4fa1cs1f6a5s4e8w";
public static final String SYN_TURN = "syn_turn";
public static final String SYN_SYSTURN = "syn_systurn";
public static final String SYN_CARD = "card";
public static final String SYN_LANDER = "lander";
public static final String SYN_lastTurn = "last_turn";
public static final String SYN_HISTORY ="历史记录:";
public static final String SYN_LEFTCARDNUM ="left";

```

3.3 语法规义

- 我定义了一个规则工具类 **Rule**，主要是对传入的卡牌进行语法检查和比较。

3.3.1 卡牌定义

- 我是用一个数组来实现卡牌的定义。

```

public static final String rule="3456789XJQKA2-+";//规则

```

- 为了方便，用 X 来代表 10，- 代表小王，+ 代表大王。

3.3.2 出牌类型

- 我定义了一个枚举类型来说明出的牌的类型。

```
public enum Type{
    Error,        //错误
    Single,       //单张
    Pair,         //一对
    TripleByNull, //三不带
    TripleByOne,  //三带一
}
```

3.3.3 语法检查

- 根据出牌的数量来决定用什么方法进行检查，若有错误，则返回 Error，否则返回对应的类型。

```
switch (card.length()){
    case 0:return Type.Error;
    case 1:return Type.Single;
    case 2:return twoCardsInspect(card);
    case 3:return threeCardsInspect(card);
    case 4:return fourCardsInspect(cards);
    case 5:return fiveCardsInspect(card);
    default:return defaultInspection(card);
}
```

3.3.4 比较大小

- 若没有语法错误，则具对应的类型，对两组牌进行比较，返回 true 或 false。

```
switch (type){
    case Single:return singleCompare(src,des);
    case Pair:return pairCompare(src,des);
    case TripleByNull:case TripleByOne:case TripleByPair:
        return tripleCompare(src,des);
    case Bomb:return bombCompare(src,des); |
    case Continuous: //找到相应类型的比较规则
    case PairContinuous:return continuousCompare(src,des);
    case AirplaneByNull:case AirplaneByOne:case AirplaneByPair:
        return airplaneCompare(src,des);
    case Error:return false;
}
```

3.4 打印卡牌

3.4.1 原理

- 客户端接收服务器发来的卡牌，得先对其进行大小排序。

```
private static void sortCard(ArrayList<String> card) {
    List tmp= (List) card.clone();
    card.clear();
    for (int i = rule.length-1; i >= 0; i--) {
        String key=rule[i];
        while (tmp.contains(key)){
            card.add(key);
            tmp.remove(key);
        }
    }
}
```

- 为了好看，得按一定的规则打印到屏幕上。

```
for (int i = 0; i < size; i++) {
    System.out.print(" ____");
}
System.out.println();
for (String item:cards){
    System.out.print("| "+item+" ");
}
System.out.println("| \n");
```

3.4.2 实现效果

```

| - | 2 | 2 | K | K | Q | J | J | X | X | X | 8 | 7 | 6 | 5 | 4 | 3 |
| + | 2 | 2 | A | K | X | 8 | 8 | 8 | 7 | 5 | 5 | 5 | 4 | 3 | 3 | 3 |
| A | A | K | Q | Q | Q | J | J | 9 | 9 | 7 | 7 | 6 | 6 | 6 | 4 | 4 |
| A | 9 | 9 |
```

第4章 测试运行

4.1 Waiting

- 服务器先开起来

```

C:\Program Files\Java\jdk-13.0.2\bin\java
该服务器地址为<127.0.0.1>,端口为<6666>
```

- 客户端依次加入，人没到齐时可以随意聊天

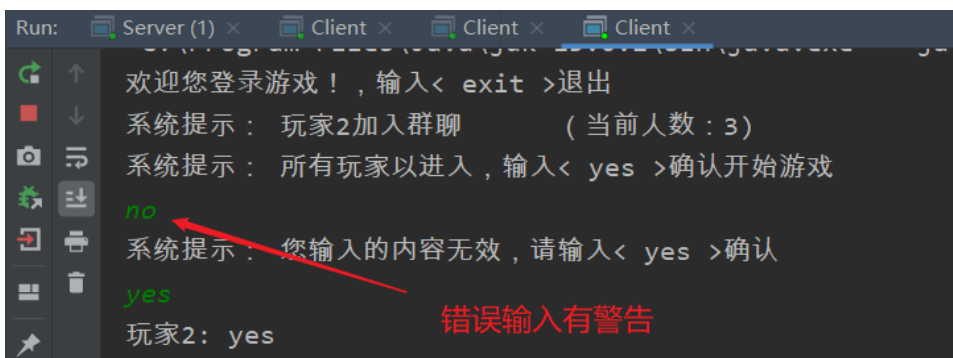
```

欢迎您登录游戏！，输入< exit >退出
系统提示： 玩家0加入群聊      (当前人数：1)
系统提示： 玩家1加入群聊      (当前人数：2)
玩家1: hello
hi
玩家0: hi
|

```

4.2 Preparation

- 当三个玩家都加入时就进入到准备阶段，玩家只能输入 yes 确认。



```

Run:  Server (1) x Client x Client x Client x
欢迎您登录游戏！，输入< exit >退出
系统提示： 玩家2加入群聊      (当前人数：3)
系统提示： 所有玩家以进入，输入< yes >确认开始游戏
no
系统提示： 您输入的内容无效，请输入< yes >确认
yes
玩家2: yes

```

4.3 Selecting

- 当玩家都输入 yes 确认以后就进入到 selecting 阶段，这个阶段发手牌并选地主，随机一个序号输入 yes 或 no 选择是否抢地主，若 10 秒没有反应，则自动换下一位玩家

```

您的卡牌为：      手牌剩余数量 | 地主0号：0张      | 玩家1号：0张      | 您2号：0张      |
| 2 | 2 | 2 | A | K | K | K | Q | Q | J | X | X | X | 7 | 5 | 5 | 3 |
系统提示： 您是否要抢地主,输入 < yes > 或 < no >, 10S倒计时
系统提示： 玩家0正在选择，请耐心等待...
系统提示： 玩家1正在选择，请耐心等待...

```

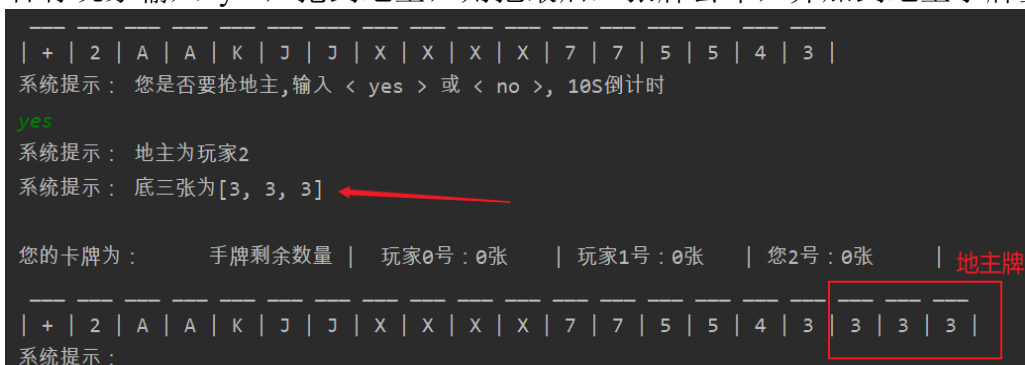
- 若无玩家选择地主，则重新洗牌在选

```

| 2 | 2 | 2 | A | K | K | K | Q | Q | J | X | X | X | 7 | 5 | 5 | 3 |
系统提示： 您是否要抢地主,输入 < yes > 或 < no >, 10S倒计时
系统提示： 玩家0正在选择，请耐心等待...
系统提示： 玩家1正在选择，请耐心等待...
系统提示： 没有玩家选择地主，重新洗牌！
您的卡牌为：      手牌剩余数量 | 地主0号：0张      | 玩家1号：0张      | 您2号：0张      |
| 2 | 2 | J | X | X | 9 | 9 | 8 | 8 | 8 | 7 | 6 | 6 | 6 | 6 | 5 | 5 |

```

- 若有玩家输入 yes，抢到地主，则把最后三张牌公布，并加到地主手牌里。



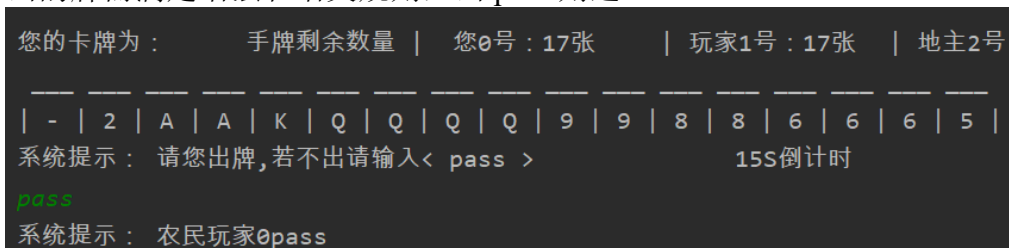
- 此时游戏进入到 running 阶段。

4.4 Running

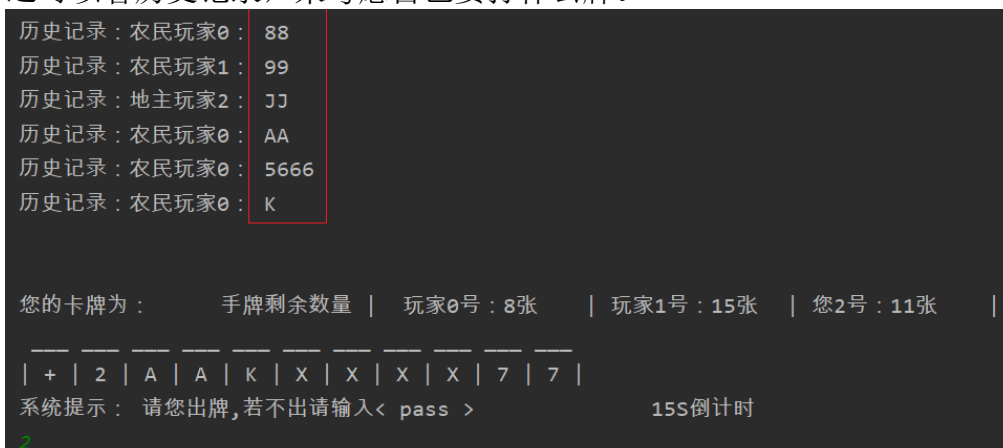
- 这个阶段，只有轮到的玩家才允许出牌，若超出时间未出牌，则默认打一张最小的。



- 出的牌需满足语法和语义规则，出 pass 则过。



- 还可以看历史记录，来考虑自己要打什么牌。



- 若有人打完手牌则进入 finish 阶段。

4.5 Finish

- 展示结果：并算本局倍率，玩家可以自由发言。

```
历史记录：农民玩家0： 99
历史记录：地主玩家2： AA
历史记录：农民玩家0： QQQQ
                                炸弹倍率 x2

您的卡牌为：      手牌剩余数量 | 您0号：1张      | 玩家1号：15张 | 地主2号：8张 |
——
| 2 |
系统提示： 请您出牌...          15S倒计时
2
农民玩家0： 2
```

```
系统提示： 恭喜农民获得胜利 !!!
系统提示： 本场倍率： 2,愿赌服输，自觉转账 !!!
玩家2： 地主快转账
-.-
玩家1： -.-
```

第5章 体会总结

这个项目差不多花了我一周的时间，代码加起来有一千行左右，也是我第一次写这么长的代码，写这个小项目用到了非常多的基础知识，如数据结构，操作系统，网络编程，编译原理等，也明白了基础的重要性，我数据结构没怎么学好，算法导论写没学过，导致有些效率不是那么高，今年也要考研了，数据结构得好好的重新学一遍，考研过后还想去学习一下算法导论，算法学好了，那么在开发项目中就能做到游刃有余了。开发完这个项目收获还是挺大的，不仅理论基础方面更加扎实了，JAVA 这门语言我也算是正是入门了，以后再遇到什么问题，也可以对症下药，也不会连问题出在哪都不知道了。网络编程这门课结束，以后也没有专业选修课了，只能靠自己了，总之，加油吧！

参考文献

- 丁振凡、章剑：java 语言使用教程（第 2 版），北京邮电大学出版社。
- 黄晓东：Java 课程设计案例精编，中国水利水电出版社,2002.4。
- 李诚、肖占彪：java 2 简明教程，清华大学出版社，2004.8。
- DavidFlanagan: effecttive enterprise java，中国电力出版社，2005.6。
- 胡书敏：java 设计模式-自动化与性能，清华大学出版社，2004.7。
- 谢希仁：计算机网络（第 7 版），电子工业出版社 2017.1
- Elliotte Rusty Harold（哈诺德 R.E.）：Java 网络编程（第四版）
- 宋敬彬、孙海滨：《Linux 网络编程》，大学出版社，2010