

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

# 斗地主

## 一、开发环境

编程语言：JAVA 语言

编程环境：IntelliJ IDEA

## 二、设计思路

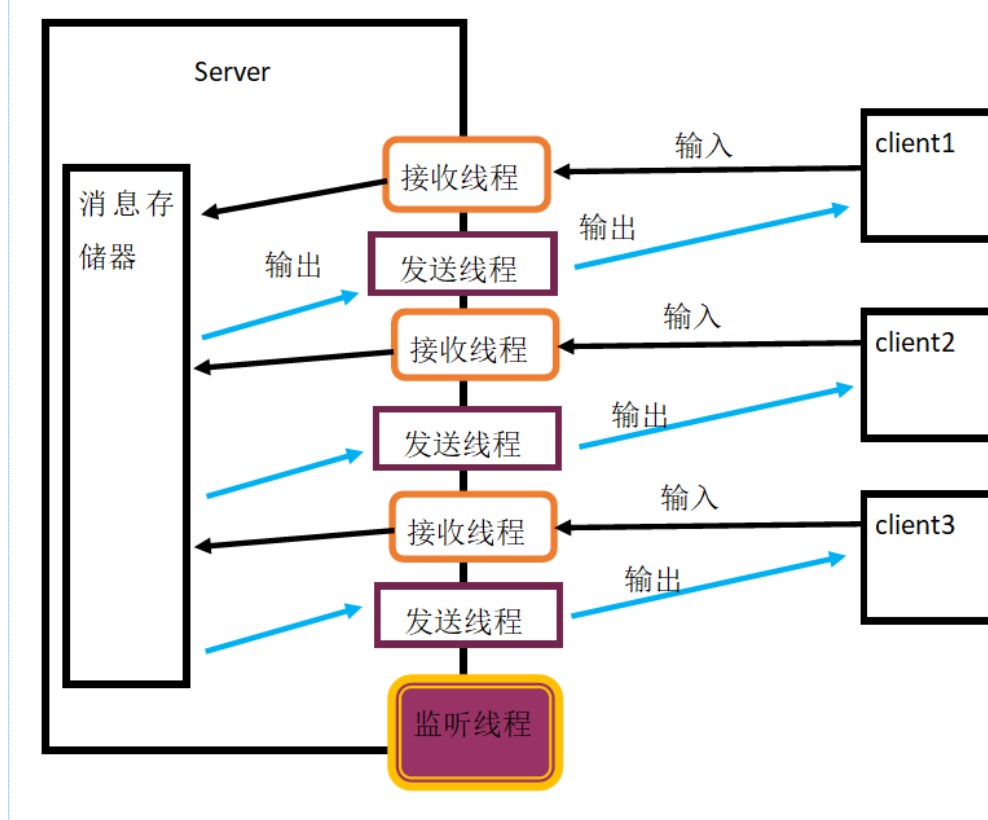
### 1. 网络通信

#### 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

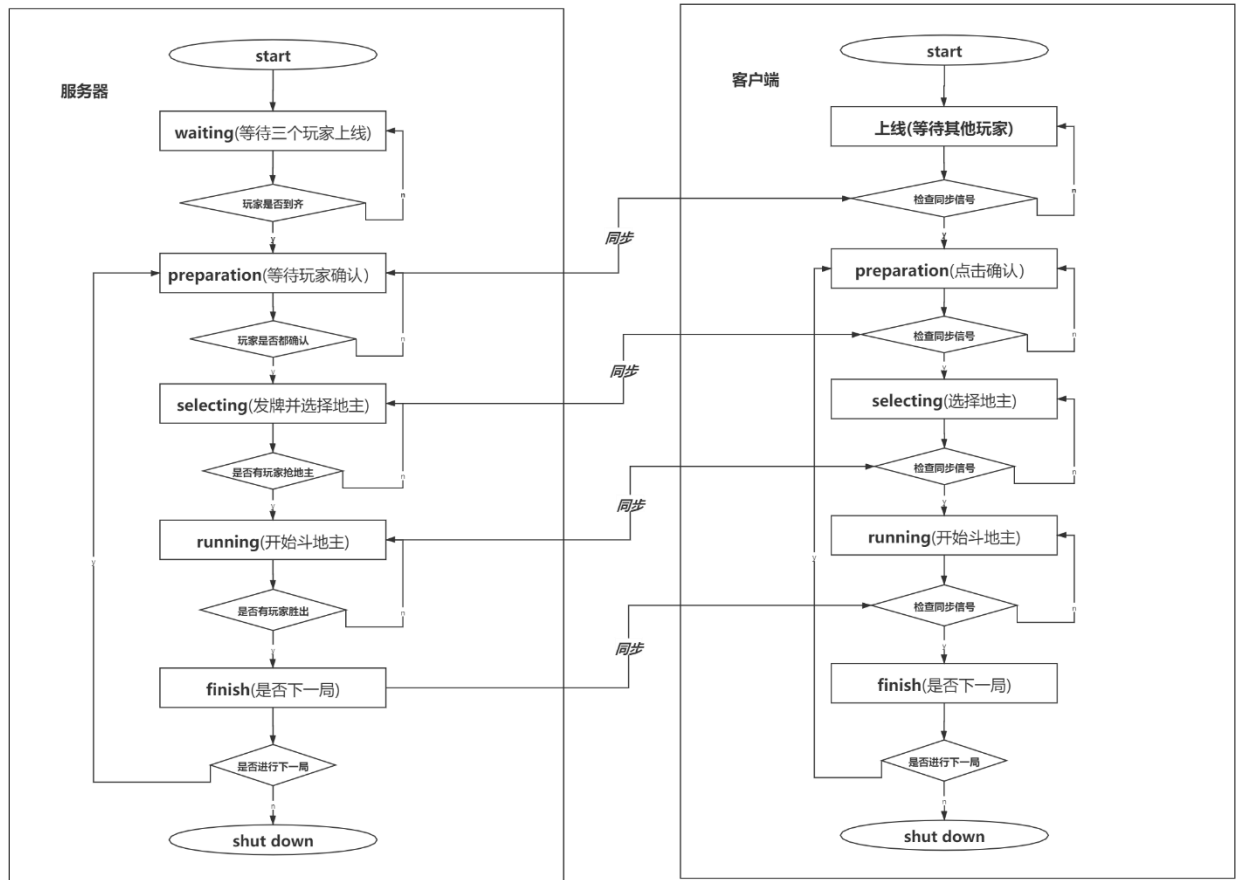
#### 2) 通信草图

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. 同步

同步草图



## 3. 语法规义分析

### 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) 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

### 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

## 三、 代码实现

### 1. TCP 通信

#### 1) 客户端

- 输入 IP 和端口，并创建套接字

```
try {    /* 输入的服务器IP和端口 */
    IP=args[0];port=Integer.parseInt(args[1]);
}catch (Exception e){
    IP="localhost";port=12306;
}
Socket 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。

```
while (!(text=sc.nextLine()).equals("exit")){
    writer.write(text);
    writer.newLine();
    writer.flush();
}
```

- 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

```
while (flag){
    while ((s = reader.readLine())!=null)
        System.out.println(s);
}
```

#### 2) 服务器

- 1、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。

```
ServerSocket serverSocket=new ServerSocket(port);/* 开启监听客户端连接的线程
ListenThread listenThread=new ListenThread(serverSocket,socketList,stringB
listenThread.start();
```

监听线程核心如下

```

while (true){
    Socket accept = socket.accept(); /* 阻塞监听，若成功则创建套接字 */
    socketList.add(accept);          /* 为每一个客户端创建一个接收消息的线程 */
    Thread thread = new ReserveThread(accept, stringBuffer, socketList, writerMap);
    thread.start();
}

```

2、接收客户端消息线程代码如下

监听消息

```

while (true){
    while ((s=reader.readLine())!=null){
        if (s.equals("exit")) break breakout;
        stringBuffer.append(name+": "+s);
    }
}

```

当加入或结束后发一条系统提示

```

Chat_TCP_Server.SYSTEM_INFO+name+"入群聊      (当前人数: "+socketList.size()+")")
(Chat_TCP_Server.SYSTEM_INFO+name+"已下线    (当前人数: "+socketList.size()+")");

```

3、接收的消息送给主线程，主线程遍历每个套接字，分别发送。

```

while (true){
    if (!stringBuffer.toString().equals("")){
        for(Socket socket:socketList){
            writer= writerMap.get(socket); /* 遍历每个套接字，发给每个客户端 */
            writer.write(stringBuffer.toString());
            writer.newLine();
            writer.flush();
        }
        stringBuffer.delete(0,stringBuffer.length()); /* 发完后清空信息 */
    }
}

```

3) 实现

```

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

```

## 2. 同步

### 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 */  
}
```

### 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(),应该说的是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) 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();  
}
```

#### 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 ;  
    }  
}
```

#### 5) 主客户端

客户端始终接收用户的输入，根据不同的阶段，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;  
    }  
}
```

#### 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)-48];
        break;
    }
}

```

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;
}

```

## 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是什么类型的数据

```

private static final String SYN_STATUS = "dfd4fa1cs1f6a5s4e8w";//设置状态同步序列码
private static final String SYN_TURN = "syn_turn";//设置顺序同步序列码
private static final String SYN_SYSTURN = "syn_systurn";//设置系统出牌顺序同步序列码
private static final String SYN_CARD = "card";//设置系统出牌顺序同步序列码
private static final String SYN_LANDER = "lander";//设置系统出牌顺序同步序列码
private static final String SYN_LastTurn = "last_turn";//设置最后一次出牌者序号同步序列码
private static final String SYN_HISTORY = "历史记录 : "; //记录出牌历史
private static final String SYN_LEFTCARDNUM = "left"; //玩家剩余手牌同步信号

```

## 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

### 1) 卡牌定义

我是用一个数组来实现卡牌的定义

```

String rule="3456789XJQKA2-+";//规则

```

为了方便，用 X 来代表 10，- 代表小王，+ 代表大王。

### 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
}

```



### 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

```
/**
 * 检查卡片是否正确
 * @param card 待检查卡片
 * @return 卡片类型
 */
public static Type isCardCorrect(String card) {
    String[] cards = card.split(regex: "");
    for (String s : cards)
        if (!rule.contains(s)) return Type.Error; // 有非法字符
    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);
    }
}
```

### 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`

```
/* 按对应的类型比较两组牌的大小
 * @param src 待比较卡牌
 * @param des 被比较卡牌
 * @param type 比较的类型
 * @return true为大, false为比不过
 */
public static boolean compare(String src, String des, Type type) {
    if (src.length() == des.length()) { // 长度必须相等
        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); // 找到相应类型的比较规则 */
            case Continuous:
            case PairContinuous: return continuousCompare(src, des);
            case AirPlaneByNull: case AirPlaneByOne: case AirPlaneByPair:
                return airplaneCompare(src, des);
            case Error: return false;
        }
    }
    else if (type == Type.Bomb) { // 长度不等检查是否为炸弹
        return bombCompare(src, des);
    }
}
```

## 4. 打印卡牌

### 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

```
/* 按规则对card进行排序 */
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

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

### 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 |
____
| A | A | K | Q | Q | Q | J | J | 9 | 9 | 7 | 7 | 6 | 6 | 6 | 4 | 4 |
____
| A | 9 | 9 |
```

## 四、 结果演示

### 1. Waiting

服务器先开起来

```

  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

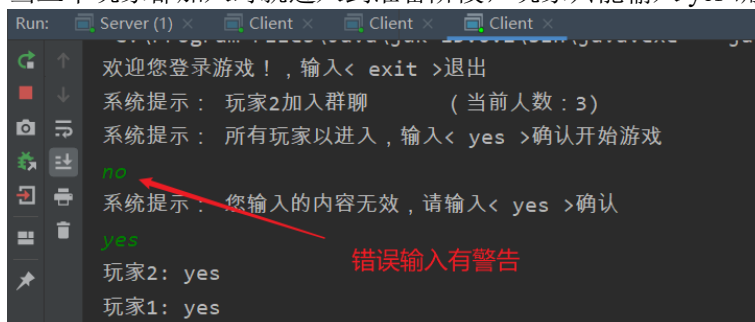
```

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
|

```

## 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
玩家1: yes

```

错误输入有警告

## 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！
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 |
系统提示： 玩家1正在选择，请耐心等待...
系统提示： 您是否要抢地主,输入 < yes > 或 < no >, 10S倒计时

```

若有玩家输入 yes，抢地主，则把最后三张牌公布，并加到地主手牌里

```

| + | 2 | A | A | K | J | J | X | X | X | X | 7 | 7 | 5 | 5 | 4 | 3 |
系统提示： 您是否要抢地主,输入 < yes > 或 < no >, 10S倒计时
yes
系统提示： 地主为玩家2
系统提示： 底三张为[3, 3, 3]
您的卡牌为：      手牌剩余数量 |  玩家0号：0张      |  玩家1号：0张      |  您2号：0张      |  地主牌
| + | 2 | A | A | K | J | J | X | X | X | X | 7 | 7 | 5 | 5 | 4 | 3 | 3 | 3 | 3 |
系统提示：

```

此时游戏进入到 running 阶段

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

```

您的卡牌为：      手牌剩余数量 |  玩家0号：17张      |  玩家1号：17张      |  您2号：20张      |
| + | 2 | A | A | K | J | J | X | X | X | X | 7 | 7 | 5 | 5 | 4 | 3 | 3 | 3 | 3 |
系统提示： 请您出牌...      15S倒计时
地主玩家2： 3      超出时间未打出，则默认出一张      计数器

```

出的牌需满足语法和语义规则，出 pass 则过

```

您的卡牌为：      手牌剩余数量 |  您0号：17张      |  玩家1号：17张      |  地主2号：15张      |
| - | 2 | A | A | K | Q | Q | Q | Q | 9 | 9 | 8 | 8 | 6 | 6 | 6 | 5 |
系统提示： 请您出牌,若不出请输入< pass >      15S倒计时
pass
系统提示： 农民玩家0pass

```

还可以看历史记录，来考虑自己要打什么牌

```

历史记录：地主玩家2： 3334
历史记录：地主玩家2： 55
历史记录：农民玩家0： 88
历史记录：农民玩家1： 99
历史记录：地主玩家2： JJ
历史记录：农民玩家0： AA
历史记录：农民玩家0： 5666
历史记录：农民玩家0： K

您的卡牌为：      手牌剩余数量 |  玩家0号：8张      |  玩家1号：15张      |  您2号：11张      |
| + | 2 | A | A | K | X | X | X | X | 7 | 7 |
系统提示： 请您出牌,若不出请输入< pass >      15S倒计时
2

```

若有人打完手牌则进入 finish 阶段

## 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： -.-
```

## 五、 心得体会

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！