

**NANCHANG UNIVERSITY**

**Java语言程序设计大作业报告**



题 目：单双人猜数字游戏

专 业： 软件工程

班 级： 卓越（东软）152班

学 号： 8000115057

学生姓名： 周寒聿西

完成人数： 1人

起讫日期：

任课教师： 周翔 职称：

部分管主任：

完成时间： 2018/1/18

**目 录**

一、需求分析 - 1 -

1.1 项目介绍 - 1 -

1.2 功能需求 - 1 -

二、系统分析 - 2 -

2.1 功能模块 - 2 -

2.2 类的设计 - 2 -

2.3 本程序需解决的关键技术问题 - 3 -

三、程序实现 - 5 -

3.1 代码目录结构 - 5 -

3.2 程序实现 - 5 -

四、系统测试 - 16 -

4.1系统功能分解 - 16 -

4.2测试内容 - 16 -

五、个人小结 - 27 -

六、参考文献 - 28 -

# 需求分析

## 项目介绍

本项目是一个猜数字小游戏。

游戏每局生成4个互不相同的数字组成一个字符串，并计时，游戏者输入所猜的数字，要求输入的字符串也是由4个互不相同的数字组成，程序检查猜测情况并给游戏者提供结果反馈，游戏者根据反馈进行下一次猜测，若猜中，则计时结束，游戏结束。程序将会记录下猜中数字所花费的最短的用时。

程序统计输入的数字有n个是与生成字符串相同且位置匹配的，记为nA，有m个是与生成字符串相同但位置不匹配的，记为mB，当统计结果为4A0B时，游戏者猜中所生成的字符串。

程序分为单人模式和双人模式，单人模式由一个人独立猜测所生成的数字，双人模式由两人合作猜测，每人将会看到自己和对方的猜测结果反馈，根据反馈进行下一次猜测。

## 功能需求

1. 游戏逻辑

随机生成4个互不相同的数字组成一个字符串。

检测游戏者输入的字符串，处理非法输入。

比较生成字符串与游戏者输入的字符串，生成结果反馈nAmB。

输入字符串与生成字符串完全一致时，猜中，反馈为4A0B

1. 单人模式

开始游戏时开始计时，中途可停止游戏，游戏者每猜测一次反馈其猜测结果，猜中停止计时。

1. 双人模式

1）匹配：

选择发起匹配或接受匹配，发起匹配者发出匹配信息，接受匹配者接受到匹配信息后选择同意与否，发出匹配回应信息。发出匹配者收到同意匹配信息后，匹配成功，收到不同意信息，匹配失败。

2）开始游戏：

匹配成功后，发起匹配者发送游戏开始时间，开始计时，开始游戏；接受匹配者收到开始时间后开始计时，开始游戏。

3）游戏中：

双方发送接收猜测信息，显示在信息区域中。

4）结束游戏：

有一方退出该局时，双方均游戏结束。

有一方猜中数字，停止计时，判断是否破纪录后发送信息，更新记录，弹出对话框提示游戏结果，游戏结束；另一方接收到猜中的信息，弹出对话框提示游戏结果，游戏结束。

1. 辅助功能

可以读取数据库信息查看游戏记录，读取文件信息查看游戏规则和程序关于信息。游戏记录中记录着单、双人模式的最短猜中用时。

1. 界面

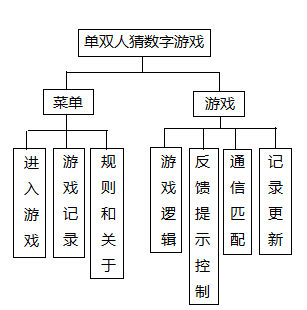
共分为开始界面，单人游戏界面，双人游戏界面，开始界面可查看游戏规则或关于界面

# 系统分析

## 功能模块

整个项目分为菜单界面模块和游戏模块。

菜单界面模块包括进入游戏界面、查看游戏记录、查看规则和关于等文本。游戏模块包括游戏逻辑控制、反馈提示控制、通信匹配控制和数据库更新四部分， 其中单人模式不包含通信匹配控制部分。



## 类的设计

本项目中的实体类有数字串类(MyString)、游戏记录类(Record)。其中MyString完成生成数字串和判断数字串是否被猜中的功能。Record有RecordDaoImp实现RecordDao接口完成游戏记录的增查改功能。

界面类有开始主界面(BeginFrame)、单人模式界面(SingleFrame)、双人模式界面(DoubleFrame)、游戏记录界面(RecordFrame)、文档信息显示界面(MessageFrame)。MessageFrame用于读取游戏规则和关于信息文件并显示，BeginFrame、singleFrame、DoubleFrame分别有自己的界面控制类BeginFrameControl、SingleFrameControl、DoubleFrameControl完成界面按钮等的响应控制。

控制类有游戏逻辑控制类(GameLogic)，数据库连接控制类(DB)、计时线程(TimeThread)、通信服务器控制类(GameServer)、通信客户端控制类(GameClient)。其中通信控制类都含有两个线程，分别控制读写。

## 本程序需解决的关键技术问题

1. 多线程

多线程是一个比较难理解和学习的概念，尤其是当线程对象比较多的时候。

在本项目的双人模式中，双方都有读写线程和计时线程，相当于有6个线程同时在运行，要理清这6个线程之间的关系，理清各个线程的任务，理清各个线程的生命周期，让6个线程互不影响，还要防止主线程受影响而阻塞，还要考虑到各个操作的顺序能不能使它们正常执行完。这些对我来说还是比较复杂的，尤其是读写线程还涉及到网络Socket通信的部分，相当于两个难点混合起来，更加有难度，比如当服务器调用socket.accpt()时，是阻塞的，这时界面上的按钮是无响应的，要退出accpt()函数，要另开一个线程调用socket.close()。

1. 时间同步

由于猜测数字串需要计时，所以当两人合作猜测时需要尽可能的将双方时间同步。

为了实现此功能，发起匹配者在接受到匹配同意信息匹配成功后，将发出游戏开始时间信息，接受匹配者收到开始时间信息，从收到的信息中的开始时刻开始计时。这样双方的时间差就是信息收发操作和开启计时线程的时延，数量级为毫秒，可以接受。

1. 网络通信Socket编程

本项目中所有的通信均用Socket完成，为了能完成匹配、时间同步和通信的任务，需要设计通信的流程和简单的协议。

参与游戏的双发需要发起匹配或接受匹配，若接受匹配，则该方为服务器方，若发起匹配，则该方为客户端方，故需要先有一方开启接受匹配，否则另一方将一直发起匹配失败。

**（1）通信流程：**

1）匹配

服务器开启，准备接受链接，客户端开启，发起链接，服务器接受链接，判断是否同意匹配，发送匹配信息。若发送拒绝信息，则服务器关闭。若客户端方接收到拒绝匹配信息，则客户端也关闭，匹配失败。若发送同意信息，客户端接收到同意信息，则匹配成功。

2）时间同步

匹配成功后，服务器等待接收游戏开始时间。客户端发送游戏开始时间后，开始计时，游戏开始，开启读进程；服务器接收到游戏开始时间后，开始计时，游戏开始，开启读进程。

3）通信

服务器客户端开启读进程后，双方随时可以收发信息通信。

4）结束通信

若我方猜中，计时结束，判断是否破纪录。若未破记录，发送猜中结果反馈信息，；若破纪录，则更新数据库后发送破纪录信息。发送完信息后，关闭socket关闭进程结束通信。

若对方猜中，则我方将接收到猜中结果反馈信息或破记录信息，接受完信息后关闭socket关闭进程结束通信。

**（2）协议：**

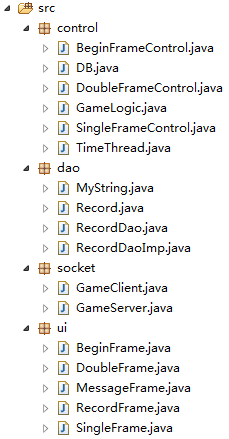
所传送的信息的第一个字段标识信息的种类，C代表普通控制信息，R代表破纪录游戏信息记录。

第一个字段为C，若第二个字段为“agree”或“reject”，则该类信息为匹配信息，分别代表服务器匹配同意信息和拒绝信息，该类信息只有这两个字段；若第二个字段为“time”，则该类信息为游戏开始时间信息，后面跟着的字段为该局游戏开始时间；若第二个字段为“0”或“1”，则该类为猜测结果反馈信息，分别代表数字串猜测信息猜中与否，后面跟着的字段分别为该次猜测所输入的数字串、该次猜测发生的时刻以及该次猜测的结果反馈信息。

第一个字段为R，则该类信息为破纪录信息，后面跟着的字段分别为该局游戏要猜的数字串，以及该局游戏猜中的时刻。

# 程序实现

## 代码目录结构



## 程序实现

以下为部分核心代码

MyString.java

//生成新字符串

**public** String initString()

{

**int** count =0;

**while**(count<length)

{

**int** num = (**int**)(Math.*random*()\*100)/10;

String s = String.*valueOf*(num);

**if**(str.indexOf(s)<0)

{

//System.out.println(""+num + ":" + s);

//System.out.println(""+s);

str += s;

count++;

}

}

**return** str;

}

//判断猜测情况是否猜对

**public** **boolean** compareString(String guess)

{

counta = 0;

countb = 0;

**for**(**int** i=0; i<length; i++)

{

**char** c = str.charAt(i);

**if**(c == guess.charAt(i))

counta++;

**if**(guess.indexOf(c)>=0)

countb++;

}

countb -= counta;

**if**(counta == 4 && countb == 0){

**return** **true**;

}**else**{

**return** **false**;

}

}

GameLogic.java

//检查输入

**public** **int** checkString(String guess){

count ++;

**int** flag = 0;

String pattern = "^[0-9]+$";

Pattern r = Pattern.*compile*(pattern);

Matcher m = r.matcher(guess);

**if**(guess.length()!=str.length()){

flag = -1;

System.***out***.println("长度不符合");

}**else** **if**(!m.matches()){

flag = -2;

System.***out***.println("输入非数字");

}**else**{

**for**(**int** i = 0; i<guess.length()-1; i++){

**for**(**int** j = i+1; j< guess.length(); j++)

**if**(guess.charAt(j)==guess.charAt(i)){

flag = -3;

System.***out***.println("输入有重复数字");

**break**;

}

**if**(flag == -3)**break**;

}

}

**if**(flag<0)

System.***out***.println("输入不合法，请重新输入！");

**return** flag;

}

TimeThread.java

**public** **void** run() {

beginTime = System.*currentTimeMillis*();

timer = 0;

**while** (state) {

**if**(timer == 3600)

timer = 0;

time = timer/60+ ":" +timer%60;

com.setText("" + time);

**try** {

*sleep*(1000);

timer++;

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

System.***out***.println("time stop!");

}

GameServer.java

//构造函数

**public** GameServer(DoubleFrame df){

d = df;

live = **true**;

matchEnable = **true**;

matchState = **false**;

guessEnable = **false**;

receive = **false**;

thread = **new** Thread(**this**);

thread.start();

rt = **new** Thread(**this**);

rt.start();

System.***out***.println("服务器启动");

}

//接受链接

**public** **synchronized** **boolean** match(){

**try** {

**try** {

server= **new** ServerSocket(6666);

System.***out***.println("服务器端口启动");

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.***out***.println(" 等待客户呼叫");

**if**(matchEnable){

d.getLabelMessage().setText("正在接受匹配...");;

socket = server.accept();

matchEnable = **false**;

matchState = **true**;

in = **new** DataInputStream(socket.getInputStream());

out = **new** DataOutputStream(socket.getOutputStream());

}

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** matchState;

}

//游戏开始

**public** **synchronized** String gameBegin(){

String s = "";

**try** {

s = in.readUTF();

System.***out***.println("服务器端开始时间："+s);

receive = **true**;

StringTokenizer st = **new** StringTokenizer(s,",");

**if**(st.nextToken().equals("C")){

**if**(!st.nextToken().equals("time")){

s = "";

}**else**{

String str = st.nextToken();

**long** time = System.*currentTimeMillis*() - Long.*parseLong*(st.nextToken());

game = **new** GameLogic();

**this**.str = game.GenerateString(str);

**if**(t == **null** || !t.isAlive()){

t = **new** TimeThread(time);

System.***out***.println("服务器端新游戏");

}

t.setState(**true**);

t.setCom(d.getLabelTimer());

t.start();

d.getPanelMessage().setVisible(**false**);

d.getPanelGame().setVisible(**true**);

d.getLabelGameMsg().setText("请输入所猜测的数字");

d.getTextField().setText(**null**);

d.getPanelGame().paintImmediately(d.getPanelGame().getBounds());

}

}

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** s;

}

//猜测一次

**public** **synchronized** String guessOnce(){

**boolean** check = **false**;

String result = game.guessOnce(d.getTextField().getText());

msg = "";

**if**(result.equals("")){

d.getLabelGameMsg().setText("输入不合法");

d.getTextField().setText(**null**);

}**else**{

**if**(!result.equals("4A0B")){//没猜中

msg = "C,0,"+d.getTextField().getText()+","

+d.getLabelTimer().getText()+",\r\n猜测结果："+result;

d.getLabelGameMsg().setText("猜测结果："+result);

d.getTextArea().append("我方：\r\n"+d.getLabelTimer().getText()+" "

+d.getTextField().getText()+",\r\n猜测结果："+result+"\r\n");

d.getTextArea().paintImmediately(d.getTextArea().getBounds());

d.getTextField().setText(**null**);

}**else**{//猜中

receive = **false**;

time = d.getLabelTimer().getText();

msg = "C,1,"+d.getTextField().getText()+","+time

+",猜中了！总用时"+time;

d.getTextArea().append("我方：\r\n"+d.getLabelTimer().getText()+" "+d.getTextField().getText()

+"猜中了,总用时"+time+"\r\n");

d.getTextArea().paintImmediately(d.getTextArea().getBounds());

gameStop();

check = checkRecord(time);

**if**(check){//破纪录

msg = "R,"+game.getStr()+","+time;

**boolean** update = updateRecord(time);

**if**(update)d.getLabelMessage().setText("记录已更新！\r\n请发起匹配或接受匹配");

**else** d.getLabelMessage().setText("记录更新失败！\r\n请发起匹配或接受匹配");

}

}

**try** {//发送信息

System.***out***.println("服务器发送：我方：\r\n"+msg);

out.writeUTF(msg);

}**catch** (SocketException e){

gameStop();

d.getLabelMessage().setText("与客户端断开连接");

serverClose();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

gameStop();

d.getLabelMessage().setText("与客户端断开连接");

serverClose();

}

**if**(!receive){

**if**(check){

JOptionPane.*showMessageDialog*(d, "猜中了，打破纪录，用时"+time,"打破纪录", JOptionPane.***INFORMATION\_MESSAGE***);

}**else**{

JOptionPane.*showMessageDialog*(d, "猜中了，用时"+time,"猜中了", JOptionPane.***INFORMATION\_MESSAGE***);

}

serverClose();

}

}

guessEnable = **false**;

**return** msg;

}

//处理接收到的信息

**public** **void** dealMsg(){

StringTokenizer st = **new** StringTokenizer(msg,",");

String snext = st.nextToken();

**if**(snext!=**null** && snext.equals("C")){

String win = st.nextToken();

String num = st.nextToken();

String tim = st.nextToken();

String tip = st.nextToken();

String str =tim+" "+ num+tip;

d.getTextArea().append("对方：\r\n" +str+"\r\n");

**if**(win!=**null** && win.equals("1")){

receive = **false**;

gameStop();

JOptionPane.*showMessageDialog*(d, "对方猜中了,总用时"+tim,

"猜中了", JOptionPane.***INFORMATION\_MESSAGE***);

serverClose();

}

}**else** **if**(snext!=**null** && snext.equals("R")){

receive = **false**;

gameStop();

JOptionPane.*showMessageDialog*(d, "对方猜中了，数字为"+st.nextToken()

+"\r\n打破纪录，用时"+st.nextToken(),"打破纪录", JOptionPane.***INFORMATION\_MESSAGE***);

serverClose();

}

}

@Override

**public** **void** run() {

// **TODO** Auto-generated method stub

**while**(live){

**try** {

Thread.*sleep*(100);

} **catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**if**(Thread.*currentThread*() == thread){

**if**(matchEnable){

**if**(match()){

**int** ret;

ret = JOptionPane.*showConfirmDialog*(d, "有匹配请求，是否同意？", "匹配请求"

,JOptionPane.***YES\_NO\_OPTION***);

**if**(ret == JOptionPane.***NO\_OPTION***){

disMatch();

gameStop();

serverClose();

}

**else**{

sendMatch();

d.getLabelMessage().setText("游戏初始化，请稍等");

gameBegin();

}

}

}

}

**if**(Thread.*currentThread*() == rt){

**if**(receive){

System.***out***.println("服务器端读");

**try** {

msg = in.readUTF();

System.***out***.println("服务器端接收：对方：\r\n" + msg);

dealMsg();

}**catch** (SocketException e){

//e.printStackTrace();

gameStop();

d.getLabelMessage().setText("与客户端断开连接");

serverClose();

}**catch** (IOException e) {

// **TODO** Auto-generated catch block

//e.printStackTrace();

gameStop();

d.getLabelMessage().setText("与客户端断开连接");

serverClose();

}

}

}

}

}

GameClient.java

//构造函数

**public** GameClient(DoubleFrame df){

d = df;

live = **true**;

matchEnable = **true**;

matchState = **false**;

receive = **false**;

guessEnable = **false**;

thread = **new** Thread(**this**);

thread.start();

rt = **new** Thread(**this**);

rt.start();

System.***out***.println("客户端启动");

}

//发起连接

**public** **synchronized** **boolean** match(){

matchState = **false**;

**try** {

d.getLabelMessage().setText("正在匹配...");

socket=**new** Socket("127.0.0.1",6666);

in = **new** DataInputStream(socket.getInputStream());

out = **new** DataOutputStream(socket.getOutputStream());

d.getLabelMessage().setText("正在等待对方同意...");

String msg = in.readUTF();

System.***out***.println("客户端接受同意信息："+msg);

StringTokenizer st = **new** StringTokenizer(msg,",");

String snext = st.nextToken();

**if**(snext!=**null** && snext.equals("C")){

snext = st.nextToken();

**if**(snext!=**null** && snext.equals("agree")){

System.***out***.println("客户端接收到同意信息");

matchState = **true**;

matchEnable = **false**;

}**else**{

d.getLabelMessage().setText("已被拒绝，请重新匹配。");

matchState = **false**;

matchEnable = **true**;

}

}

}**catch**(ConnectException e){

e.printStackTrace();

System.***out***.println("匹配失败！");

d.getLabelMessage().setText("匹配失败！");

}**catch** (UnknownHostException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

d.getLabelMessage().setText("匹配失败！");

}**catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

d.getLabelMessage().setText("匹配失败！");

}

**return** matchState;

}

//开始游戏

**public** **synchronized** **void** gameBegin(){

**try** {

game = **new** GameLogic();

str = game.GenerateString();

msg = "C,time,"+game.getStr()+","+System.*currentTimeMillis*();

out.writeUTF(msg);

System.***out***.println("客户端开始时间："+msg);

receive = **true**;

**if**(t == **null** || !t.isAlive()){

t = **new** TimeThread(0);

System.***out***.println("客户端新游戏");

}

t.setState(**true**);

t.setCom(d.getLabelTimer());

t.start();

d.getPanelMessage().setVisible(**false**);

d.getPanelGame().setVisible(**true**);

d.getLabelGameMsg().setText("请输入所猜测的数字");

d.getTextField().setText(**null**);

d.getPanelGame().paintImmediately(d.getPanelGame().getBounds());

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

@Override

**public** **void** run() {

// **TODO** Auto-generated method stub

**while**(live){

**try** {

Thread.*sleep*(100);

} **catch** (InterruptedException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**if**(Thread.*currentThread*() == thread){

**if**(matchEnable){

**if**(match()){

gameBegin();

}**else**{

clientClose();

}

}

}

**if**(Thread.*currentThread*() == rt){

**if**(receive){System.***out***.println("客户端读");

**try** {

msg = in.readUTF();

System.***out***.println("客户端接收：对方：\r\n" + msg);

dealMsg();

}**catch** (SocketException e){

gameStop();

d.getLabelMessage().setText("与服务器断开连接");

clientClose();

}

**catch** (IOException e) {

gameStop();

d.getLabelMessage().setText("与服务器断开连接");

clientClose();

}

}

}

}

}

DoubleFrameControl.java

//退出该局

d.getBtnExit().addActionListener(**new** ActionListener(){

@Override

**public** **void** actionPerformed(ActionEvent e) {

// **TODO** Auto-generated method stub

**new** Thread(){//防止此时服务器正在接受链接阻塞主线程

**public** **void** run(){

**if**((gc!=**null**&&gc.isLive())){

gc.socketClose();

}

**else** **if**((gs!=**null**&&gs.isLive())){

gs.socketClose();

}

d.getLabelMessage().setText("请发起匹配或接受匹配");

d.getPanelMessage().setVisible(**true**);

d.getPanelGame().setVisible(**false**);

}

}.start();

}

});

# 系统测试

## 4.1系统功能分解

菜单模块：进入游戏界面，查看游戏记录，查看规则等文本文档内容，退出游戏

游戏模块：

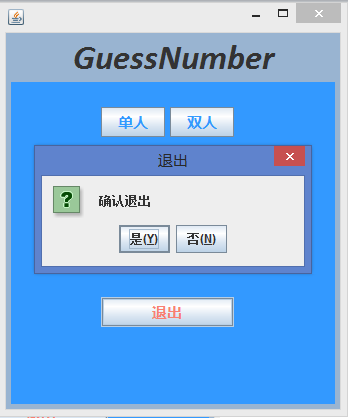
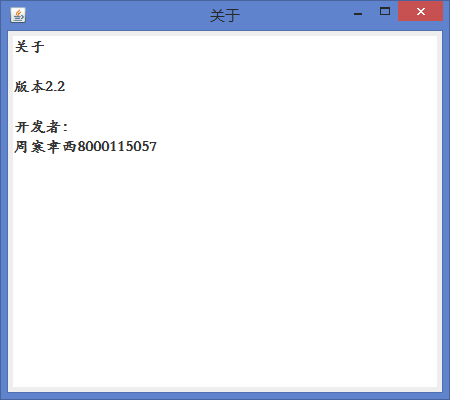
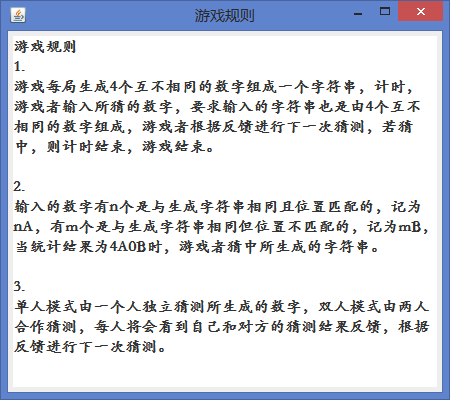
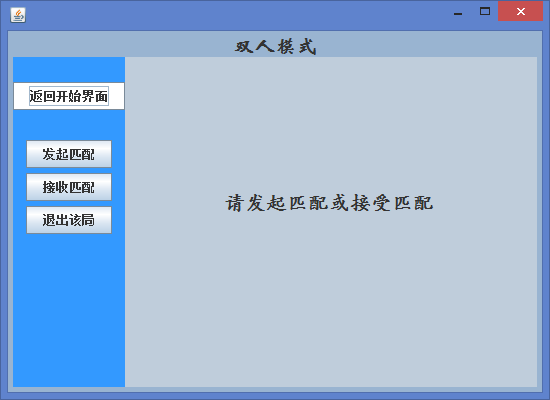
单人模式：开始游戏，停止游戏，返回开始界面，检测输入合法性，显示检测结果，猜中数字串，破纪录

双人模式：发起匹配，接受匹配，退出该局，返回开始界面，开始游戏，发送信息，接受信息，文本区显示，猜中数字串，破纪录

## 4.2测试内容

（1）菜单模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **功能** | **预期输入** | **预期输出** | **实际结果** | **备注** |
| 进入游戏界面 | 点击单人 | 进入单人模式 | Pass |  |
| 进入游戏界面 | 点击双人 | 进入双人模式 | Pass |  |
| 查看游戏记录 | 无 | 显示游戏记录界面 | Pass |  |
| 查看规则等文本文档内容 | 无 | 显示文本内容界面 | Pass |  |
| 退出游戏 | 点击退出 | 弹出退出确认对话框 | Pass |  |
| 退出游戏 | 点击叉号 | 弹出退出确认对话框 | Pass |  |

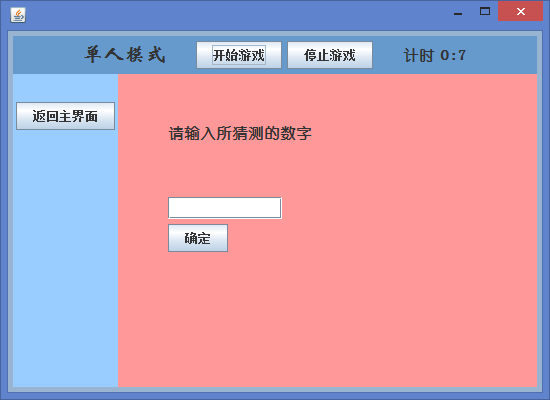
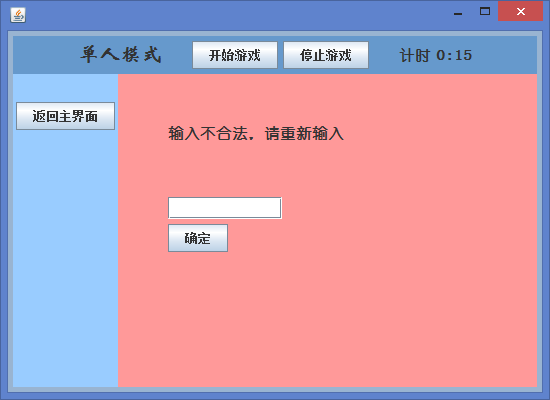
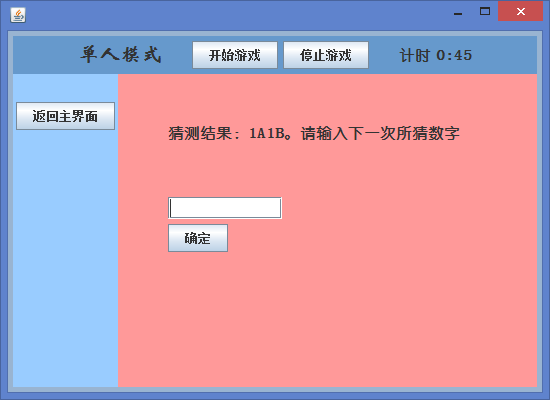
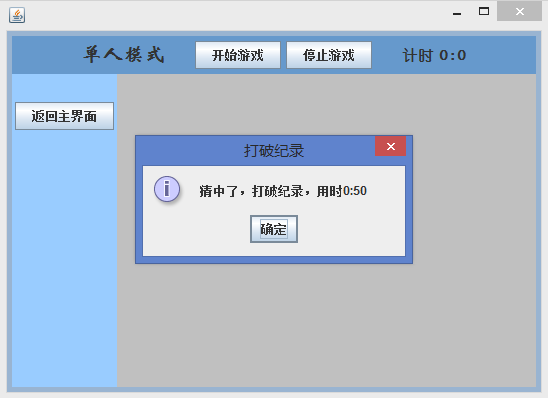
 

(2)游戏模块

1)单人模式

开始游戏，停止游戏，返回开始界面，检测输入合法性，显示检测结果，猜中数字串，破纪录

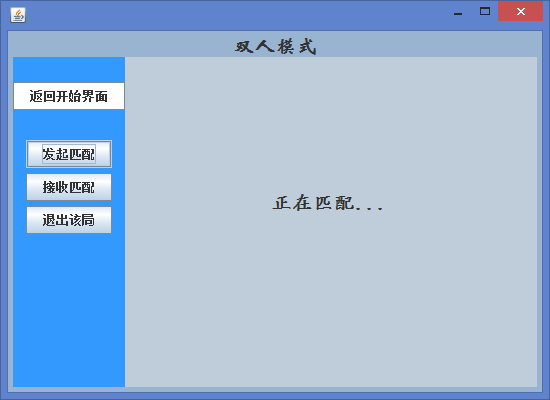
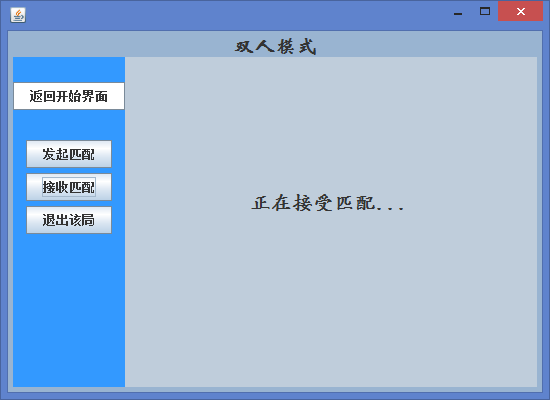
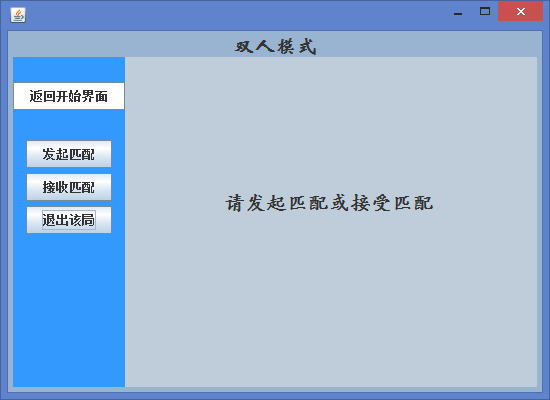
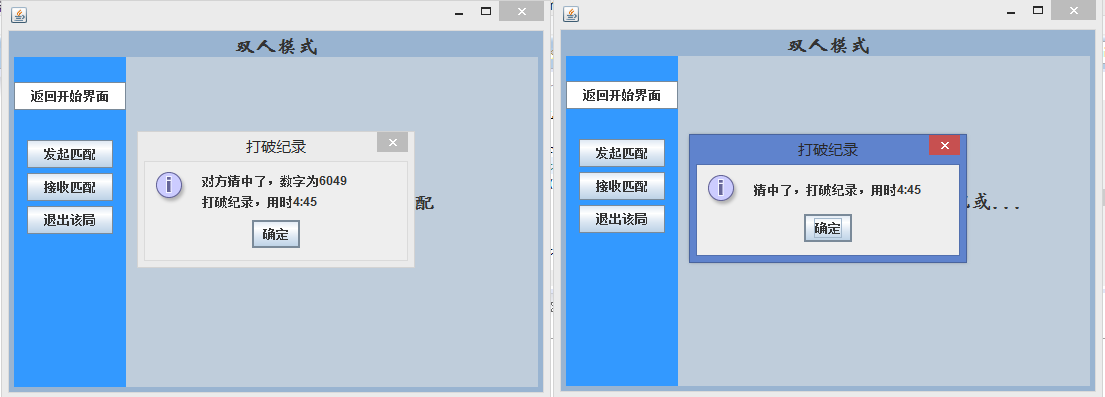
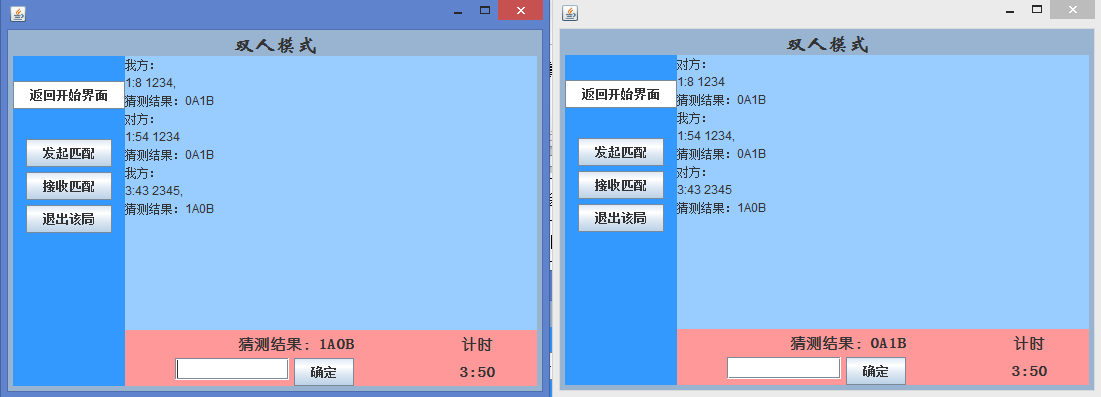
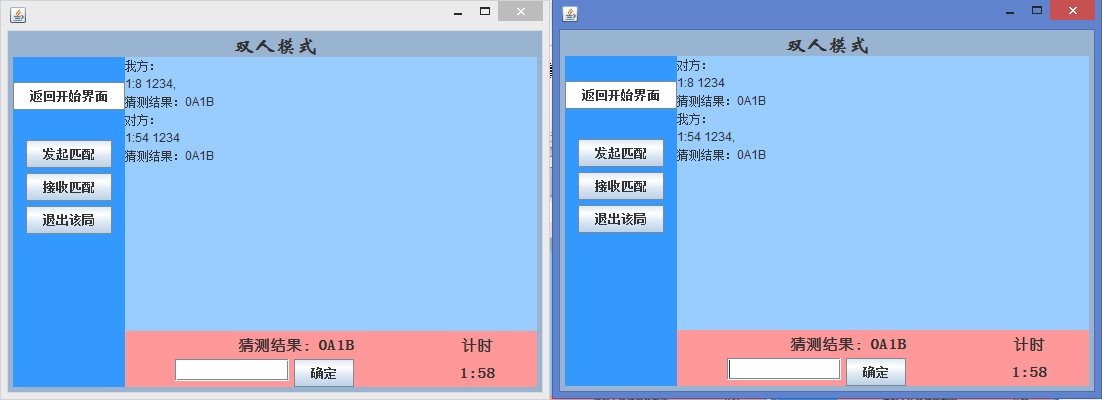
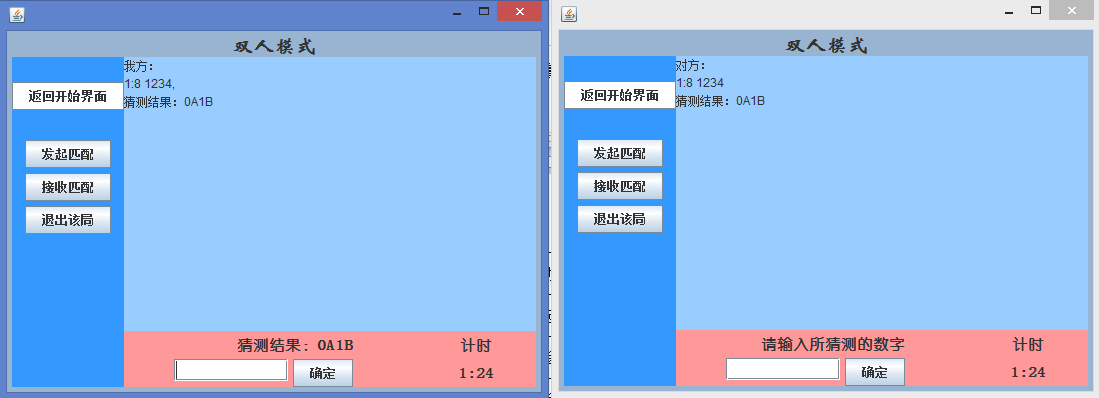
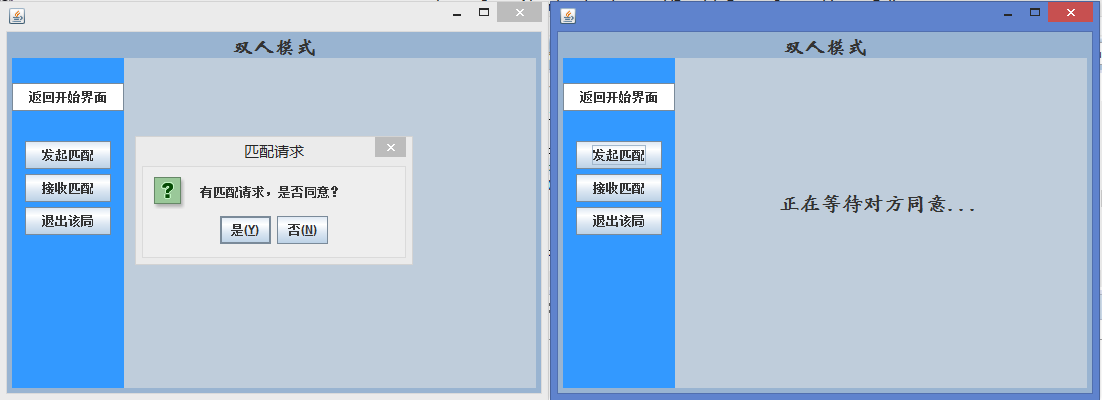
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **功能** | **预期输入** | **预期输出** | **实际结果** | **备注** |
| 开始游戏 | 无 | 界面切换，计时开始 | Pass |  |
| 停止游戏 | 无 | 界面切换，计时重置 | Pass |  |
| 返回开始界面 | 无 | 返回开始界面 | Pass |  |
| 检测输入合法性 | 无 | 显示输入不合法 | Pass |  |
| 检测输入合法性 | Abcd | 显示输入不合法 | Pass |  |
| 检测输入合法性 | 123 | 显示输入不合法 | Pass |  |
| 检测输入合法性 | 1233 | 显示输入不合法 | Pass |  |
| 显示检测结果 | 1234 | nAmB | Pass |  |
| 显示检测结果 | 猜中 | 弹出猜中对话框 | Pass |  |
| 破纪录 | 破纪录 | 弹出破纪录对话框 | Pass |  |

2）双人模式

发起匹配，接受匹配，退出该局，返回开始界面，开始游戏，发送信息，接受信息，文本区显示，猜中数字串，破纪录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **功能** | **预期输入** | **预期输出** | **实际结果** | **备注** |
| 发起匹配 | 无 | 显示正在匹配 | Pass |  |
| 接受匹配 | 无 | 显示正在接受匹配 | Pass |  |
| 接受匹配 | 对方发起匹配 | 显示匹配确认对话框 | Pass |  |
| 退出该局 | 无 | 返回最初界面 | Pass |  |
| 返回开始界面 | 无 | 返回开始主界面 | Pass |  |
| 开始游戏 | 同意匹配 | 双方开始游戏 | Pass |  |
| 发送信息 | 1234 | 双方文本区更新 | Pass |  |
| 接受信息 | 1234 | 双方文本区更新 | Pass |  |
| 文本区显示 | 1234 | 双方文本区更新 | Pass |  |
| 猜中数字串 | 猜中 | 弹出猜中对话框 | Pass |  |
| 破纪录 | 破纪录 | 弹出破纪录对话框 | Pass |  |

# 个人小结

在这次项目中，多线程和socket编程是两大难点，绕在一起纠结了很久，总是会出各种各样的问题。比如有的线程一直不睡眠，导致一直占用资源，其他线程无法执行。比如想控制线程的死亡，自己摸索出设置布尔型变量作为run()函数while循环条件的方法。通信流程真的是改了很久才逐渐完善的，总是规划好了但一上手就发现考虑不周到要改动，这时候真是对创造了我们现在这个网络世界中各种通信方法和标准的前辈们倍感尊敬了。

另外还有一些别的古怪错误。比如由于猜测双方有一方先关闭socket，使另一方无法正常读，异常处理中也关闭socket，于是服务器客户端线程都关闭了，可如果这次打破了记录要更新数据库，结果没来得及更新线程就死亡了，这种错误会以为是数据库操作出了什么问题，但其实是线程之间的影响，十分有迷惑性。比如读文件时，一直循环用同一个数组获取文件输出流读出的内容，但用完一次后，又不将数组值置空，于是读到文件末尾时数组读不满，数组末尾的一些空间就依然还是上一次读到的值，导致最后输出总有一些字符是重复的。

这些错误其实和编程的习惯也有关系，编程时不注意变量及其值的状态，不注意线程的状态，不注意逻辑的完整性和连贯性，就会出现这种好像思维黑洞一样的错误，不过解决这些错误是提高自己的途径。

最后，对于项目中类的设计，其实我觉得还是不好，由于刚学线程和网络编程，不知道如何去划分这些部分与界面控制之间的联系，导致这几个类之间耦合程度很大，还有服务器和客户端之间其实很多功能都是一样或者相似的，比如对一次猜测的处理，对收到的信息的处理，完全可以再抽象出一个类，但这次项目中还没能完成这个工作，这也导致在修改这些部分的代码时，总要修改两遍，还经常忘记自己哪一边还没有改动，给自己的编程增加了不少麻烦。

希望自己可以继续努力，不断完善自己的编程能力和实践学到的编程思想，再接再厉。

# 参考文献

[1]《java.awt.IllegalComponentStateException error with JComboBox》

https://community.oracle.com/thread/1562812

[2]《控件使用错误:component must be showing on the screen to determine its location》 http://www.myexception.cn/j2se/22016.html

[3]《Java Socket 多线程聊天室》 whenever 2016-05-25

https://www.cnblogs.com/whenever/p/5526420.html

[4]《Java Swing 实时刷新JTextArea，以显示不断append的内容？》 谭荣\_1996 2017-05-08

https://www.cnblogs.com/tanrong/p/6825215.html

[5]《JavaSwing 表格JTable内容居中 自定义TableModel》 挨踢南java 2016-01-22

http://blog.csdn.net/xujinnan/article/details/50562938

[6]《JFrame窗体居中显示》 iii1860 2014-08-04

http://blog.csdn.net/III1860/article/details/38363863

[7]《java JFrame 关闭窗口时确认》 Derek 2011-10-15

http://blog.csdn.net/linghao00/article/details/6875797

[8]《java 如何 向txt文件写入换行符？》

http://bbs.csdn.net/topics/390397478

[9]《java获取当前路径的几种方法》 Franson 2016-08-02

https://www.cnblogs.com/franson-2016/p/5728280.html