《Java 语言程序设计》开放式大作业报告

大作业题目	网络联机游戏-狼人杀			类型	游戏
班　号	120L0221		学　号	120L022115 120L022109 120L022116 120L022130	
所在院系	基础学部	学　期	2021 年夏季学期	任课教师	于海宁
类型	综合设计型				

实验目的：

- 掌握程序设计的基本算法和简单数据结构基础，能够综合运用基本控制语句、算法和数据结构，以及自顶向下、逐步求精的模块化设计方法和面向对象的设计方法，能够设计具有小规模的系统级Java语言程序，提高系统编程能力；
- 针对计算相关的复杂工程问题，能够使用恰当的算法和数据结构，完成计算、统计、排序、检索、匹配等相关的软件系统的构造、测试与实现；
- 能够基于面向对象的思想进行程序的设计与实现；
- 掌握常用的程序调试和测试方法。

实验要求：

- 采用自顶向下、逐步求精的模块化设计思想设计一个小型信息库管理系统，或者闯关式游戏程序。
- 要求解释说明采用了什么数据结构和算法，为什么选择这种数据结构或算法，系统实现过程中遇到了哪些问题，这些问题是如何解决的，还有什么问题尚未解决，今后打算从哪几个方面进行改进，本设计的亮点和难点在哪里，实验结果如何，有哪些收获和学习体会；
- 编写程序完成以下大作业内容并完成大作业报告。

实验内容：

设计并实现一个网络联机的狼人杀小游戏（有图形化界面），游戏由以下流程组成：

1、 各玩家进行连接

2、 进入等候房间，等待房主（作为服务器的主机）开始游戏。

3、 进行游戏

3.0 抽取胜负

3.1 天黑请闭眼：狼人选择，预言家选择

3.2 狼人完成选择后，判断是否有人死亡（且女巫解药未使用），若有，向女巫询问是否使用解药

3.3 若女巫毒药未使用，向女巫询问是否使用毒药

3.4 判断猎人是否在当局死亡，若死亡，向猎人询问是否发动技能。

3.5 天亮请睁眼：宣布最后死亡情况

3.6 活着的玩家轮流发言

3.7 所有活着的玩家经进行投票

3.8 统计投票结果

3.9 若有人投票出局，判断是否为猎人，若为猎人，询问是否使用技能

3.10 判断是否得出胜负

3.11 若未得出，进行下一轮游戏（步骤 3.1），若得出胜负，结束游戏

4、结束游戏，宣布结果，广播玩家存活情况及身份

实验环境：

操作系统：Win10

Java 版本：开发版本为 JDK16，经测试 JAVA8 不能直接编译代码，但可运行 jar 包

集成开发环境：Eclipse 及 IntelliJ IDEA

外部库：无

输入输出设计：

程序输入的数据：

用户名：String 类型，便于玩家区分自己和其他玩家（若不输入，可自动生成且保证不重复）

IP：String，用于和房主（作为服务器的主机）进行连接

若输入错误的 IP，将会告知用户，服务器未开启

发言字符串：String 类型，用于发言，与其他玩家交流

程序输出的数据：

程序输出的数据都在图形化界面内部展示

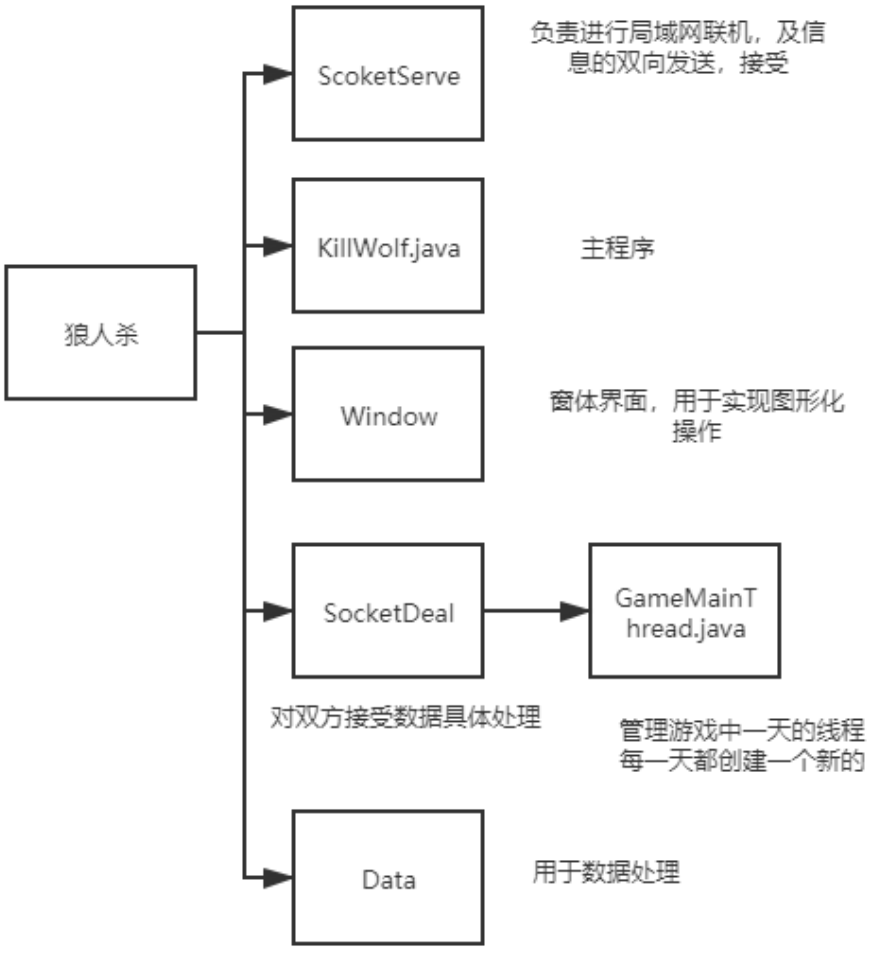
均为 String 型

用户体验：借助 javax.swing 下的类实现了图形化界面，用户只需要进行点击和少量的输入即可

系统设计与实现:

1. 系统功能模块划分

对系统进行自顶向下的模块分解，画出系统各个功能模块之间的结构图如下：



2. 类设计

本系统总计设计了36个类，每个类的数据成员和成员函数设计如下所示。

(1)KillWolf类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	server	sServer	服务端对象
2	client	sClient	客户端对象
3	freeSpeak	boolean	是否自由发言
4	isServer	boolean	是否为服务器
5	Serverlp	String	服务器 IP
6	Mip	String	用户自己的 IP
7	UserName	String	用户自己的名称
8	Users	DataUser[]	人员数组
9	UsersLen	int	数组占用数
10	UsersRealLen	int	人数
11	loginwindow	LoginWindow	登录房间页面
12	WaitRoom	WaitRoom	等待房间页面
13	GameMain	GameMain	游戏房间页面
14	SelectWindow	SelectWindow	选择页面

序号	成员函数	函数功能	函数参数	函数返回值
1	main	主函数，获取 IP 启动登录页面	String	void
2	goWaitRoom	进入等待房间	无	void
3	goGameMain	进入游戏房间	无	void

(2) serverData类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	StartSpeakName	String	第一位发言的用户名
2	nowSpeakUser	String	正在发言的用户名
3	deathUser	String	狼人刀杀用户
4	deathUser2	String	女巫毒药用户
5	deathUser3	String	猎人枪杀用户
6	deathUser4	String	投票出局用户
7	GameMainThread	GameMainThread	当前天的主线程
8	Day	int	游戏天数
9	night	boolean	是否夜晚
10	playMode	boolean	游戏模式
11	freeSpeak	boolean	是否自由发言
12	ServerUsers	DataUser[]	服务端用户
13	UsersLen	int	服务端用户数组占用
14	UsersRealLen	int	服务端用户数量
15	selection	SelectionEvent[]	选择活动数组
16	selectionLen	int	选择活动数组占用

序号	成员函数	函数功能	函数参数	函数返回值
1	getUsersWithID	通过身份获取用户对象	String	DataUser[]
2	ArrayDatatoName	将用户对象数组转换为用户名数组	DataUser[]	String[]
3	getLivingUsers	获取未出局用户对象	boolean	Object[]
4	isAllWolfDied	判断所有狼死亡		boolean
5	isAllPeoDied	判断所有平民死亡		boolean
6	isAllGodDied	判断所有神死亡		boolean
7	whoWin	判断是否胜利		String
8	getUser	通过用户名获取用户	String	DataUser

(3) DataUser类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	sk	Socket	用户端口对象
2	port	int	用户端口号
3	name	String	用户名
4	live	boolean	是否存活
5	identify	String	身份
7	havePoison	boolean	有毒药
8	haveAntidote	boolean	有解药

(4) SelectCallBack接口的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	run	处理选择回调	string	void

(5) SelectionEvent类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	Selection	SelectionEvent[]	静态 所有选择活动
2	SelectionLen	int	静态 所有选择活动数量
4	open	boolean	是否正在开启
5	users	String[]	被投票用户
6	count	int[]	计数
7	openusers	String[]	参与用户
8	target	String[]	目标用户
9	timeLimit	int	时间限制
10	cb	SelectionEventCallBack	回调函数
11	server	sServer	服务器

序号	成员函数	函数功能	函数参数	函数返回值
1	SelectionEvent	构造选择活动	Sting,SelectionEventCallBack	
2	update	更新选择	String	void
3	isFinish	判断是否完成		boolean
4	getResult	获取结果		int
5	closeSelectio	结束	sServer,boolean	boolean
6	openSeletion	开启	sServer,string,boolean,Datauser	void
7	UserQuit	用户退出	String	

(6) ThreadWait类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	s	sServer	服务器

序号	成员函数	函数功能	函数参数	函数返回值
1	ThreadWait	构造线程等待	sServer	
2	run	执行线程等待		void

(7)SelectionEventCallBack接口的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	run	回调函数	String,int	void

(8) ClientDeal类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	run	回调函数	String,socket	void

(9) SleepThread类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	sss	int	等待时长

序号	成员函数	函数功能	函数参数	函数返回值
1	SleepThread	构造线程等待	string	
2	run	执行线程等待		void

(10) GameMainThread类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	YuYancompleted	boolean	预言是否完成
2	Wizardcompleted	boolean	女巫是否完成

序号	成员函数	函数功能	函数参数	函数返回值
1	run	执行		void
2	gameNight	入夜		void

3	gameWizard	女巫选择	string	void
4	gameDay	白天		void

(11) TimeTha类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	dayWork	公布结果		void
2	gamevote	投票		void
3	gameHunt	猎人	string	void
4	gameNext	下一步		void
5	gameEnd	结束		void
6	isWin	是否胜利		boolean

(12)ServerDeal类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	run	处理回调	String,socket	void
2	nextSpeak	下一个发言	String,int	boolean

(13)SpeakThread类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明	取值范围
1	user	String	正在发言的用户	
2	Day	int	天数	

序号	成员函数	函数功能	函数参数	函数返回值
1	SpeakThread	构造线程等待	String,int	
2	run	执行		void

(14)WolfSelect类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	run	执行狼人选择	String,int	void

(15) VoteSelection类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	run	执行投票选择	String,int	void

(16) GameMain类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	freeSpeak	boolean	是否自由发言
2	UserListData	string	存活 JList 数据
3	UserListTipData	string	提示 JList 数据
4	UserOutListData	string	淘汰 JList 数据
5	ClockRun	boolean	是否正在运行 Clock

序号	成员函数	函数功能	函数参数	函数返回值
1	reset	启动		void
3	forbidSubmit	禁用发言		void
4	ableSubmit	启用发言		void

(17)TimeTh类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	startSpeakClock	启用发言时钟		void
2	submit	提交发言		void
3	addToScreen	添加到公屏	String	void
4	setDark	设置窗口风格		void
5	setWhite	设置窗口风格		void
28	getUserListData		String	
29	setUserListData		String	void
30	getUserOutListData		String	
31	setUserOutListData		String	void

(18) ClientDeal类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	ServerButton	JButton	作为服务器按钮
2	LoginButton	JButton	登录按钮
3	userName	JTextField	用户名称输入框
4	Serverlp	JTextField	服务器地址输入框
5	window	JFrame	窗口
6	MIP	String	自己的 ip

序号	成员函数	函数功能	函数参数	函数返回值
1	Reset	重置	string	void

(19) ButtonAction类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	actionPerformed	按钮点击	ActionEvent event	void

(20) LinkAction类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	actionPerformed	按钮点击	ActionEvent event	void
2	createClient	创建客户端	string	void

3	setName	设置名称		boolean

（21）SelectWindow类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明
1	isRunCB	boolean	是否已回调
2	ReceivelD	String	特征值
3	ReceiveFromServer	boolean	是否联网投票
4	DisableParent	boolean	禁用父窗口
5	ParentWindow	JFrame	父窗口
6	window	JFrame	窗口
7	CenterLabel	JLabel	中心标签
8	List	JList String	列表
9	name	String	名称
10	ConfirmButton	JButton	确认按钮
11	CloseButton	JButton	关闭按钮
12	originUsers	String	原始用户
13	usersTip	String	用户提示
14	targetUsers	String	目标用户
15	selectCount	int	选择计数
16	LastSelect	String	最后选择
17	isSendFinal	boolean	是否已发送
18	DarkMode	boolean	窗口风格
19	cb	SelectCallBack	回调函数

序号	成员函数	函数功能	函数参数	函数返回值
1	setColorMode	设置风格	boolean	void
2	start	开始	String,boolean,int	void
3	SelectWindow	构造	String,boolean,JFrame)	

（22）TimeTh类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	setTitle	设置标题	string	void
2	SELECTInfo		string	void

（23）ButtonAction类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	ButtonAction	按钮点击回调	string	
2	actionPerformed		(ActionEvent)	void
	getSelectionAndClose		String	void

（24）WaitRoom类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明

1	LoginButton	JButton	登录按钮
2	peoCount	JTextField	平民人数
3	wolfCount	JTextField	狼人数
4	wizardOption	JCheckBox	启用女巫
5	hunterOption	JCheckBox	启用猎人
6	yuyanjiaOption	JCheckBox	启用预言家
7	winmodeOption	JCheckBox	启用屠城模式
8	speakOption	JCheckBox	启用自由发言
9	UserList	JList	用户列表
10	window	JFrame	窗口
11	Userlenlabel	JLabel	
12	Charlenlabel	JLabel	
13	CharLen	int	
14	peoLen	int	
15	wolfLen	int	
16	wizardLen	int	
17	hunterLen	int	
18	yuyanjiaLen	int	

序号	成员函数	函数功能	函数参数	函数返回值
1	reset		String,boolean	void
2	setUserCount()		String	void

（25）CheckValueChanged 类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	CheckValueChanged		String	
2	itemStateChanged()		ItemEvent	void

（26）TextListener类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	TextListener()		String	
2	insertUpdate(DocumentEvent)		DocumentEven	void
3	removeUpdate(DocumentEvent)		DocumentEven	void
4	changedUpdate(DocumentEvent)		DocumentEvent	void

（27）CountKeyListener类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值

1	keyTyped		KeyEvent	void
2	keyPressed(KeyEvent)		KeyEvent	void
3	keyReleased()		KeyEvent	void

（30）CallBack接口的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	run		String, Socket	void

（31）ClientThread类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明	取值范围
1	PORT	int		
2	Log	boolean		
3	sk	Socket		
4	cb	CallBack		

序号	成员函数	函数功能	函数参数	函数返回值
1	wlog		string	void
2	close0			boolean
3	send		string	boolean
4	start		String,CallBack)	boolean

（32）sIP类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明	取值范围
1	Log	boolean		

序号	成员函数	函数功能	函数参数	函数返回值
1	getIP		string	

（33）GetRealLocalIP类的数据成员和成员函数设计：

序号	成员函数	函数功能	函数参数	函数返回值
1	getRealIP()		string	

（34）sServer类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明	取值范围
1	PORT	int		
2	Log	boolean		
3	ss	ServerSocket		
4	users	Socket		
5	usersLen	int		
6	cb	CallBack		

序号	成员函数	函数功能	函数参数	函数返回值

1	wlog(String)		string	void
2	close			boolean
3	sendto(String		String,socket	boolean
4	send(String)		string	boolean
5	start(CallBack)		CallBack	boolean

（35）ServerThreadAC 类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明	取值范围
1	id	int		
2	Log	boolean		

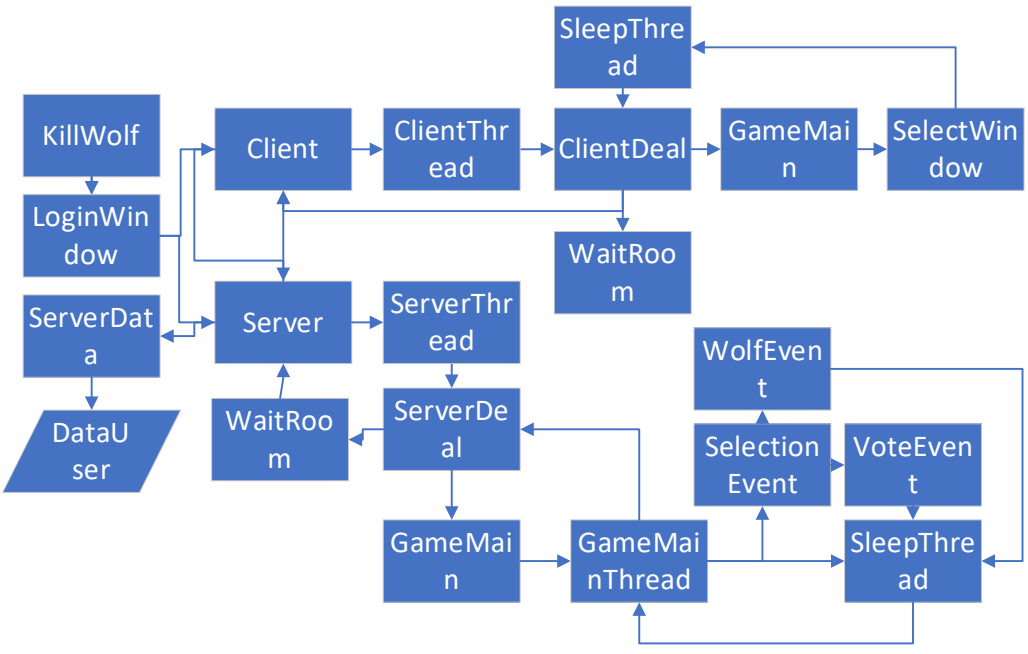
序号	成员函数	函数功能	函数参数	函数返回值
1	run			void

（36）ServerThread类的数据成员和成员函数设计：

序号	数据成员	数据类型	意义说明	取值范围
1	sk	Socket		
2	id	int		
3	Log	boolean		

序号	成员函数	函数功能	函数参数	函数返回值
1	ServerThread		Socket	
2	run			void

类及类之间的关系如图所示：



3. 数据结构

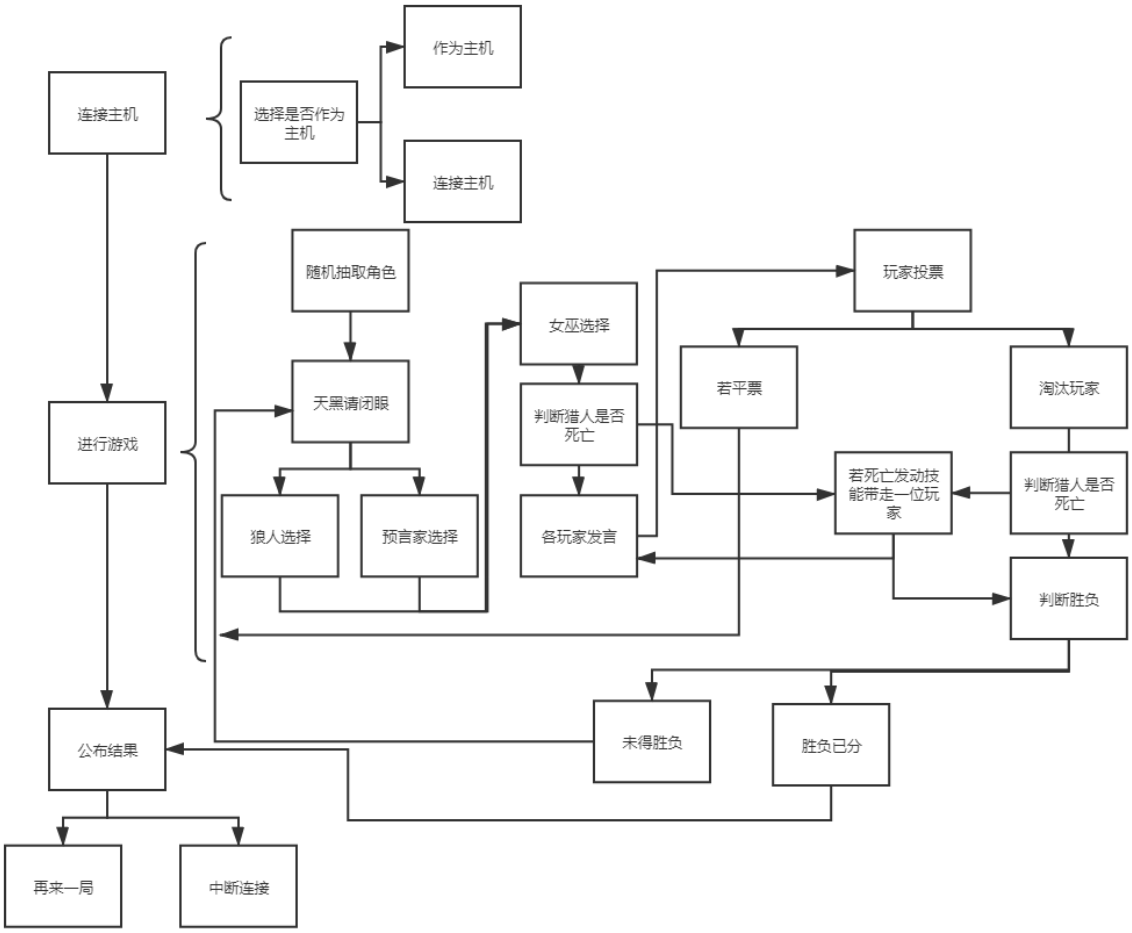
整型数组、对象数组、容器类

4. 算法

枚举/递推/迭代/分类统计、排序/查找、递归算法

5. 程序流程图

系统总体流程图如下：



实验过程中遇到的问题及解决方法与思路：

问题 1：用户选择过程中主进程阻塞。

原因：主线程 Sleep。

解决方法：启用新线程进行 Sleep，再执行回调。

问题 2：用户淘汰的列表更新。

原因：用户断线、多端数据异步通信、底层数据库和图形页面通信。

解决方法：启用静态变量记录窗口，在用户断线时对所有用户的所有开放窗口进行广播。

问题 3：用户中途退出导致发言顺序出错

原因：记录第一位发言的玩家的 ID 进行轮流发言，直到再次轮到该玩家则结束发言。

解决方法：用户退出则重新开始发言。

问题 4：游戏主体窗口公屏显示问题。

原因：对 swing 中组件使用问题。

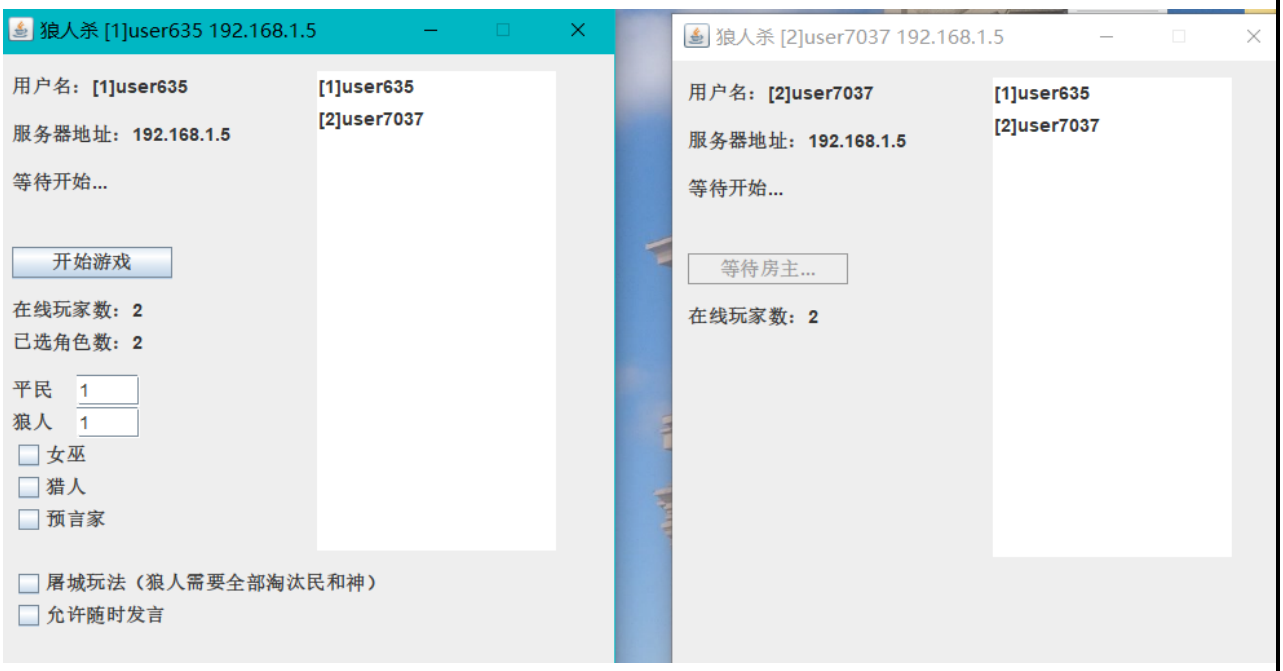
解决方法：增加滚动条：增加 JPanel, JScrollPane 并进行设置-设置屏幕显示：

```
screen.setEditable(false);//设置只读
        screen.setLineWrap(true);        //激活自动换行功能
        screen.setWrapStyleWord(true);           // 激活断行不断字功能
```

测试用例和系统测试结果：

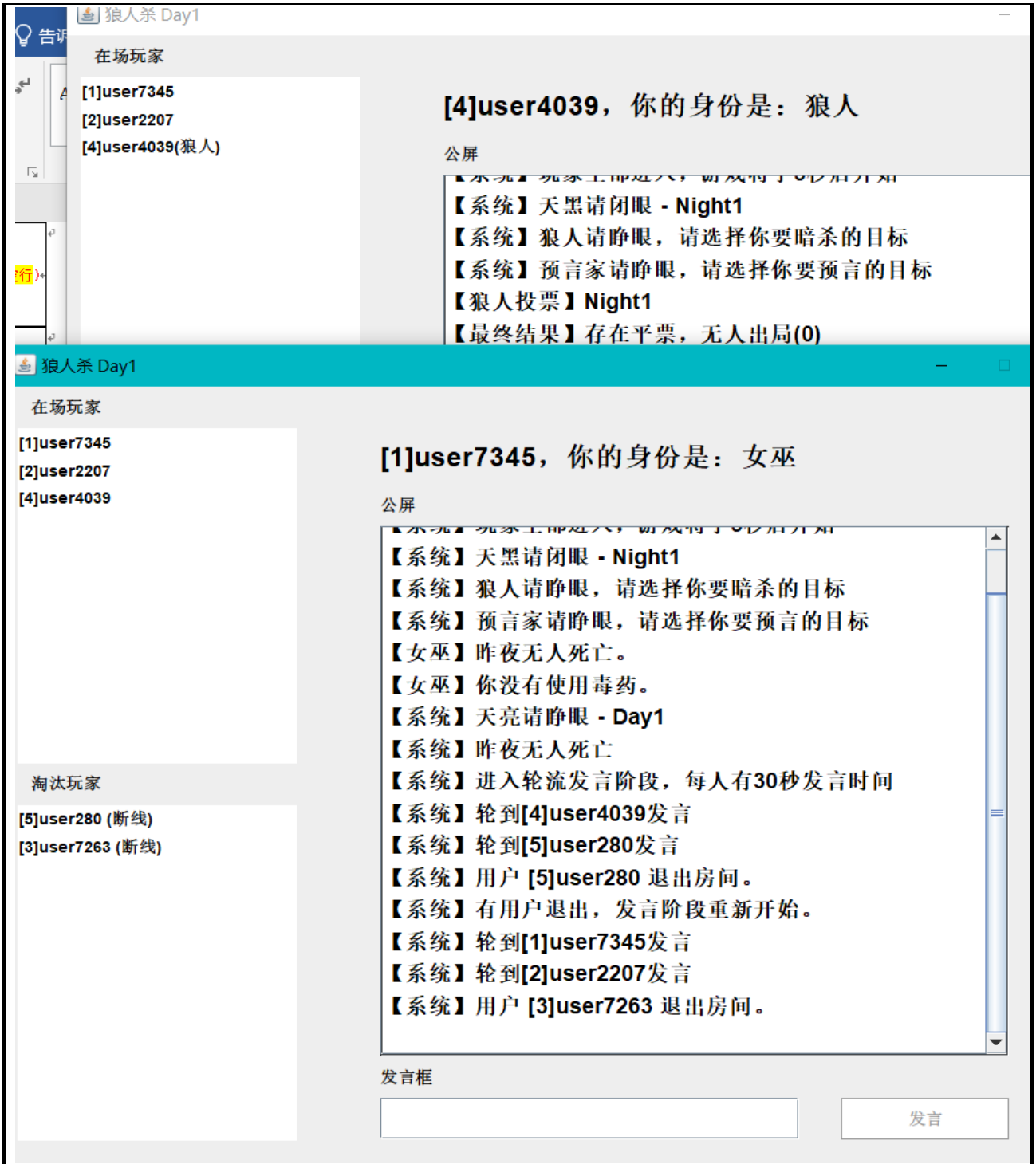
测试用例 1：

测试用例 2：



测试用例 3：


狼人杀 Night1
在场玩家
[1]user7345
[2]user2207
[3]user7263
[4]user4039
[5]user280
[3]user7263，你的身份是：平民
公屏
【系统】等待玩家...
【系统】玩家全部进入，游戏将于3秒后开始
【系统】天黑请闭眼 - Night1
【系统】狼人请睁眼，请选
【系统】预言家请睁眼，请
淘汰玩家
发言框
20s 狼人出没
请选择你要暗杀的人
(0)[1]user7345
(0)[2]user2207
(0)[3]user7263
(0)[4]user4039 (狼人)
(0)[5]user280
暗杀他
弃权
10s 预言
请选择你要预言的人
[1]user7345
[2]user2207
[3]user7263
[4]user4039
[5]user280
预言
弃权
发言
[1]user7345，
【系统】等待玩
【系统】玩家全
在场玩家
[1]user7345
[2]user2207
[3]user7263
[4]user4039 (狼人)
[5]user280
淘汰玩家
[4]user4039，你的身份是：狼人
公屏
【系统】等待玩家...
【系统】玩家全部进入，游戏将于3秒后开始
【系统】天黑请闭眼 - Night1
【系统】狼人请睁眼，请选择你要暗杀的目标
【系统】预言家请睁眼，请选择你要预言的目标
【狼人投票】Night1
【最终结果】存在平票，无人出局(0)
【系统】天亮请睁眼 - Day1
【系统】昨夜无人死亡
【系统】进入轮流发言阶段，每人有30秒发言时间
【系统】轮到[4]user4039发言
发言框
发言(9s)


狼人杀 Day1
在场玩家
[1]user7345
[2]user2207
[4]user4039(狼人)
[4]user4039，你的身份是：狼人
公屏
【系统】天黑请闭眼 - Night1
【系统】狼人请睁眼，请选择你要暗杀的目标
【系统】预言家请睁眼，请选择你要预言的目标
【狼人投票】Night1
【最终结果】存在平票，无人出局(0)
狼人杀 Day1
在场玩家
[1]user7345
[2]user2207
[4]user4039
[1]user7345，你的身份是：女巫
公屏
【系统】天黑请闭眼 - Night1
【系统】狼人请睁眼，请选择你要暗杀的目标
【系统】预言家请睁眼，请选择你要预言的目标
【女巫】昨夜无人死亡。
【女巫】你没有使用毒药。
【系统】天亮请睁眼 - Day1
【系统】昨夜无人死亡
【系统】进入轮流发言阶段，每人有30秒发言时间
【系统】轮到[4]user4039发言
【系统】轮到[5]user280发言
【系统】用户 [5]user280 退出房间。
【系统】有用户退出，发言阶段重新开始。
【系统】轮到[1]user7345发言
【系统】轮到[2]user2207发言
【系统】用户 [3]user7263 退出房间。
淘汰玩家
[5]user280 (断线)
[3]user7263 (断线)
发言框
发言

程序源代码

```
1   //文件名 KillWolf.java
2   package KillWolf;
3   import java.net.Socket;
4
5   import java.util.Scanner;
6
7   import javax.swing.JFrame;
8
9   import KillWolf.Data.DataUser;
10  import KillWolf.Data.SelectionEvent;
11  import KillWolf.Data.SelectionEventCallBack;
12  import KillWolf.KillWolf.serverData;
13  import KillWolf.SocketDeal.GameMainThread;
14  import KillWolf.Window.GameMain;
15  import KillWolf.Window.LoginWindow;
16  import KillWolf.Window.SelectWindow;
17  import KillWolf.Window.WaitRoom;
18  import SocketServe.*;
19
20
21  public class KillWolf {
22      public class serverData {
23          public static String StartSpeakName = null;
24          //public static int startSpeakUserArrayId;
25          public static String nowSpeakUser;
26          //public static int nowSpeakUserArrayId;
27          public static String deathUser = null;      //狼人杀死/女巫解救
28          public static String deathUser2 = null; //女巫毒死          为**NONE**表示无人死亡
29          public static String deathUser3 = null; //猎人杀死
30          public static String deathUser4 = null; //投票出局
31          public static GameMainThread GameMainThread = null;
32          public static int Day = 0;
33          public static boolean night = false;
34          public static boolean playMode = false; //true 表示屠城玩法，false 表示屠边玩法
35          public static boolean freeSpeak = false;
36          public static DataUser ServerUsers[] = new DataUser[50];
37          public static int UsersLen = 0;
38          public static int UsersRealLen = 0;
39          public static SelectionEvent Selection[] = new SelectionEvent[200];
40          public static int SelectionLen = 0;
41          public static DataUser[] getUsersWithID(String identify) {
42              int[] a = new int[ServerUsers.length];
43              int n = 0;
44              for(int i=0;i< UsersLen;i++) {
```

```
45                      if(ServerUsers[i] != null)
46                      if(ServerUsers[i].identify.equals(identify)) {
47                          a[n] = i;
48                          n++;
49                      }
50                  }
51                  DataUser[] b = new DataUser[n];
52                  for(int i=0;i<n;i++) b[i] = ServerUsers[a[i]];
53                  return b;
54              }
55              public static String[] ArrayDatatoName(DataUser[] d) {
56                  int[] a = new int[d.length];
57                  int n = 0;
58                  for(int i=0;i<d.length;i++) {
59                      if(d[i] != null)
60                      if(d[i].live) {
61                          a[n] = i;
62                          n++;
63                      }
64                  }
65                  String[] b = new String[n];
66                  for(int i=0;i<n;i++) {
67                      b[i] = d[a[i]].name;
68                  }
69                  return b;
70              }
71              public static Object[] getLivingUsers(boolean onlyName) {
72                  int[] a = new int[ServerUsers.length];
73                  int n = 0;
74                  for(int i=0;i<UsersLen;i++) {
75                      if(ServerUsers[i] != null)
76                      if(ServerUsers[i].live) {
77                          a[n] = i;
78                          n++;
79                      }
80                  }
81                  String[] b = new String[n];
82                  DataUser[] c = new DataUser[n];
83                  for(int i=0;i<n;i++) {
84                      if(onlyName) b[i] = ServerUsers[a[i]].name;
85                      else c[i] = ServerUsers[a[i]];
86                      System.out.println("LIVING : "+ServerUsers[a[i]].name);
87                  }
88                  if(onlyName) return b;
```

```
89                  else return c;
90              }
91              public static boolean isAllWolfDied() {
92                  DataUser[] b = getUsersWithID("狼人");
93                  for(int i=0;i<b.length;i++) if(b[i].live) return false;
94                  return true;
95              }
96              public static boolean isAllPeoDied() {
97                  DataUser[] b = getUsersWithID("平民");
98                  for(int i=0;i<b.length;i++) if(b[i].live) return false;
99                  return true;
100             }
101             public static boolean isAllGodDied() {
102                 DataUser[] b = getUsersWithID("猎人");
103                 for(int i=0;i<b.length;i++) if(b[i].live) return false;
104                 b = getUsersWithID("预言家");
105                 for(int i=0;i<b.length;i++) if(b[i].live) return false;
106                 b = getUsersWithID("女巫");
107                 for(int i=0;i<b.length;i++) if(b[i].live) return false;
108                 return true;
109             }
110             public static String whoWin() {
111                 for (int i=0;i<serverData.UsersLen;i++){
112                     if(serverData.ServerUsers[i]!=null){
113                         System.out.println(serverData.ServerUsers[i].name +" "
114         +serverData.ServerUsers[i].identify+" "+serverData.ServerUsers[i].live);
115                     }
116                 }
117                 if(playMode) {
118                     if(isAllWolfDied()) return "平民";
119                     if(isAllPeoDied() && isAllGodDied()) return "狼人";
120                     return "无";
121                 }else {
122                     if(isAllWolfDied()) return "平民";
123                     if(isAllPeoDied() || isAllGodDied()) return "狼人";
124                     return "无";
125                 }
126             }
127             public static DataUser getUser(String name) {
128                 for(int i=0;i<UsersLen;i++) {
129                     if(ServerUsers[i] != null)
130                     if(ServerUsers[i].name.equals(name)) {
131                         return ServerUsers[i];
132                     }
```

```
133                 }
134                 return null;
135             }
136
137     }
138
139     public static sServer server;
140     public static sClient client;
141     public static boolean freeSpeak = false;
142     public static boolean isServer = false;
143     public static String ServerIp;
144     public static String Mip;
145     public static String UserName;
146     public static DataUser Users[] = new DataUser[50];
147     public static int UsersLen = 0;
148     public static int UsersRealLen = 0;
149     public static LoginWindow loginwindow;
150     public static WaitRoom WaitRoom;
151     public static GameMain GameMain;
152     public static SelectWindow SelectWindow;
153     public static void main(String[] args) {
154         //Scanner sn = new Scanner(System.in);
155         Mip = sIP.getIP();
156
157         loginwindow = new LoginWindow();
158         loginwindow.reset(Mip);
159
160         System.out.println("Server on IP: "+Mip);
161
162     }
163
164
165     public static void goWaitRoom() {
166
167
168         loginwindow.window.setVisible(false);
169         WaitRoom = new WaitRoom();
170         WaitRoom.reset(Mip,UserName,isServer);
171     }
172     public static void goGameMain() {
173         WaitRoom.window.setVisible(false);
174         GameMain = new GameMain();
175         GameMain.reset();
176
```

```
177         }
178 }
179 // CallBack.java
180 package SocketServe;
181 /*
182 *       author: wbx
183 */
184 import java.net.Socket;
185
186 public interface CallBack {
187         public void run(String text,Socket sk);
188 }
189 // sClient.java
190 package SocketServe;
191 /*
192 *       author: wbx
193 */
194 import java.net.*;
195 import java.io.*;
196 import java.util.Scanner;
197
198 public class sClient {
199         public int PORT = 15648;
200         public boolean Log = true;
201         public Socket sk = null;
202         public CallBack cb = null;
203         void wlog(String log) {
204                 if(Log) System.out.println(log);
205         }
206         public boolean close() {
207                 try {
208                         sk.close();
209                 }catch(Exception e) {
210                         e.printStackTrace();
211                         return false;
212                 }
213                 return true;
214         }
215         public boolean send(String Text)
216         {
217                 try{
218                         PrintStream ps = new PrintStream(sk.getOutputStream());
219                         ps.println(Text);
220                         ps.flush();
```

```
221                 //if(Text.equals("END")) {
222                 //      sk.close();
223                 //}else {
224                 this.wlog(sk.getLocalPort()+" Client Send: "+Text);
225                 //}
226             }
227             catch(Exception e){
228                 e.printStackTrace();
229                 return false;
230             }
231             return true;
232         }
233         public boolean start(String ServerIp,CallBack cb) //= "127.0.0.1")
234         {
235             this.cb = cb;
236             try{
237                 sk = new Socket(ServerIp,PORT);
238             }
239             catch(Exception e){
240                 e.printStackTrace();
241                 return false;
242             }
243             this.wlog("Connecting Server on "+ServerIp+","+PORT);
244             try {
245                 ClientThread st = new ClientThread(sk);
246                 st.start();
247             }catch(Exception e) {
248                 e.printStackTrace();
249                 return false;
250             }
251             this.wlog("Server is success connected on "+ServerIp+","+PORT);
252             return true;
253         }
254         //
255         class ClientThread extends Thread
256         {
257             Socket sk;
258             public ClientThread(Socket sk){
259                 this.sk= sk;
260             }
261             public void run() {
262                 try {
263                     Scanner sn = new Scanner(sk.getInputStream());
264                     while(true) {
```

```
                            if(sn.hasNextLine()) {
                                String line = sn.nextLine();//br.readLine();
                                //System.out.println(sk.getLocalPort()+" Client Receive: "+line);
                                cb.run(line,sk);
                            }else {
                                try {
                                    sk.sendUrgentData(0);
                                } catch (IOException e) {
                                    System.out.println("Disconnect from Server");
                                    cb.run("Disconnect from Server",sk);
                                    break;
                                }


                            }
                        }
                    } catch (IOException e1) {
                        e1.printStackTrace();
                    }
                }
            }
        }
        //sIP.java
        package SocketServe;
        /*
        *       author: wbx
        */
        import java.net.*;
        import java.util.Enumeration;

        public class sIP {
            public static boolean Log = true;
            public static String getIP() {//获取本机 IP，返回空为失败
                String MIP = "";
                try{
                    //InetAddress addr = InetAddress.getLocalHost();
                    //MIP = addr.getHostAddress();
                    //if(Log) System.out.println("本机 IP 地址："+MIP);
                    //String hostname = addr.getHostName();
                    //if(Log) System.out.println("本机名称："+hostname);
                    //getAllIpAddress();
                    MIP = GetRealLocalIP.getRealIP();
                    //    System.out.println();
                }catch(Exception e){
```

```
309                 e.printStackTrace();
310                 return "";
311             }
312             return MIP;
313         }
314         /**
315
316         * 获取本地真正的 IP 地址，即获得有线或者无线 WiFi 地址。
317
318         * 过滤虚拟机、蓝牙等地址
319
320         * @author yins
321
322         * @date 2018 年 8 月 12 日  下午 9:53:58
323
324         */
325
326         public class GetRealLocalIP {
327         /**
328
329         * 获取本地真正的 IP 地址，即获得有线或者无线 WiFi 地址。
330
331         * 过滤虚拟机、蓝牙等地址
332
333         * @author yins
334
335         * @date 2018 年 8 月 12 日下午 9:56:35
336
337         * @return
338
339         */
340
341         public static String getRealIP() {
342         try {
343         Enumeration<NetworkInterface> allNetInterfaces = NetworkInterface.getNetworkInterfaces();
344
345         while (allNetInterfaces.hasMoreElements()) {
346         NetworkInterface netInterface = (NetworkInterface) allNetInterfaces.nextElement();
347
348         // 去除回环接口，子接口，未运行和接口
349
350         if (netInterface.isLoopback() || netInterface.isVirtual()|| !netInterface.isUp())continue;
351         if (!netInterface.getDisplayName().contains("Intel")&&
352             !netInterface.getDisplayName().contains("Realtek")) continue;
```

```
353         Enumeration<InetAddress> addresses = netInterface.getInetAddresses();
354         System.out.println(netInterface.getDisplayName());
355
356         while (addresses.hasMoreElements()) {
357         InetAddress ip = addresses.nextElement();
358         if (ip != null) {
359         // ipv4
360
361         if (ip instanceof Inet4Address) {
362         System.out.println("ipv4 = " + ip.getHostAddress());
363
364         return ip.getHostAddress();
365
366         }
367
368         }
369
370         }
371
372         break;
373
374         }
375
376         } catch (SocketException e) {
377         System.err.println("Error when getting host ip address"
378
379         + e.getMessage());
380
381         }
382
383         return null;
384
385         }
386
387         }
388         /*
389
390           * This method is used to get all ip addresses from the network interfaces.
391           * network interfaces: eth0, wlan0, l0, vmnet1, vmnet8
392
393         public static void getAllIpAddress() {
394               try {
395                     //get all network interface
396                     Enumeration<NetworkInterface>                    allNetworkInterfaces                    =
```

```
    NetworkInterface.getNetworkInterfaces();
397                 NetworkInterface networkInterface = null;
398
399                 //check if there are more than one network interface
400                 while (allNetworkInterfaces.hasMoreElements()) {
401                     //get next network interface
402                     networkInterface = allNetworkInterfaces.nextElement();
403                     //output interface's name
404                     //System.out.println("network interface: " + networkInterface.getDisplayName());
405
406                                         //get all ip address that bound to this network interface
407                     Enumeration<InetAddress> allInetAddress = networkInterface.getInetAddresses();
408
409                     InetAddress ipAddress = null;
410
411                     //check if there are more than one ip addresses
412                     //band to one network interface
413                     while (allInetAddress.hasMoreElements()) {
414                         //get next ip address
415                         ipAddress = allInetAddress.nextElement();
416                         if (ipAddress != null && ipAddress instanceof Inet4Address) {
417
418                             System.out.println("ip address: " + ipAddress.getHostAddress());
419                         }
420                     }
421                 }
422
423             } catch (SocketException e) {
424                 e.printStackTrace();
425             }
426         }//end method getAllIpAddress*/
427 }
428 //sSever.java
429 package SocketServe;
430 /*
431 *       author: wbx
432 */
433 import java.net.*;
434 import java.io.*;
435 import java.util.Scanner;
436
437 public class sServer {
438         public int PORT = 15648;
439         public boolean Log = true;
```

```
440     public ServerSocket ss = null;
441     public Socket[] users = new Socket[50];
442     public int usersLen = 0;
443     public CallBack cb = null;
444     void wlog(String log) {
445         if(Log) System.out.println(log);
446     }
447     public boolean close() {
448         try {
449             ss.close();
450         }catch(Exception e) {
451             e.printStackTrace();
452             return false;
453         }
454         return true;
455     }
456     public boolean sendto(String Text,Socket sk) {
457         if(sk == null) return false;
458         PrintStream ps;
459         try {
460             ps = new PrintStream(sk.getOutputStream());
461             ps.println(Text);
462             ps.flush();
463             this.wlog("Send to "+sk.getPort()+" : "+Text);
464         } catch (IOException e) {
465             // TODO Auto-generated catch block
466             e.printStackTrace();
467             return false;
468         }
469
470         return true;
471     }
472     public boolean send(String Text) {
473         try{
474             for(int i=0;i<usersLen;i++) {
475                 Socket sk = users[i];
476                 if(sk == null) continue;
477                 PrintStream ps = new PrintStream(sk.getOutputStream());
478                 ps.println(Text);
479                 ps.flush();
480                 this.wlog("Send "+(i+1)+"/"+usersLen+" to "+sk.getPort()+" : "+Text+"####");
481                 Thread.sleep(10);
482             }
483         }
```

```
484             catch(Exception e){
485                 e.printStackTrace();
486                 return false;
487             }
488             return true;
489         }
490         public boolean start(CallBack cb) //= "127.0.0.1")
491         {
492             this.cb = cb;
493             try {
494                 ss = new ServerSocket(PORT);
495                 ServerThreadAC st = new ServerThreadAC();
496                 st.start();
497             } catch (IOException e) {
498                 // TODO Auto-generated catch block
499                 e.printStackTrace();
500                 return false;
501             }
502             return true;
503         }
504         //????????¿ɔ??○???
505         public class ServerThreadAC extends Thread
506         {
507             //String ServerIp;
508             int id = 0;
509             boolean Log = true;
510             //public ServerThreadAC(String ServerIp){
511             //       this.ServerIp = ServerIp;
512             //}
513             public void run() {
514                 try {
515                     if(Log) System.out.println("Server is open on "+PORT);
516                     //??h??while??????????fi??Ö?????ɔ??????????
517                     while(true){
518                         Socket sk = ss.accept();//????????????Ö?ڸ?????
519                         ServerThread st = new ServerThread(sk);
520                         st.start();
521                     }
522                 }catch(Exception e) {
523                     e.printStackTrace();
524                 }
525             }
526         }
527         //????????ɔ??○???
```

```
528     public class ServerThread extends Thread
529     {
530         Socket sk;
531         int id = 0;
532         boolean Log = true;
533         public ServerThread(Socket sk){
534             this.sk= sk;
535         }
536         public void run() {
537             try {
538                 users[usersLen] = sk;
539                 this.id = usersLen;
540                 usersLen++;
541                 if(Log) System.out.println("New User in "+sk.getPort());
542                 cb.run("New User: "+sk.getPort(),sk);
543                 Scanner sn = new Scanner(sk.getInputStream());
544                 while(true) {
545                     if(sn.hasNextLine()) {
546                         String line = sn.nextLine();//br.readLine();
547                         //System.out.println("Server Receive from "+sk.getPort()+": "+line);
548                         cb.run(line,sk);
549                     }else {
550                         try {
551                             sk.sendUrgentData(0);
552                         } catch (IOException e) {
553                             System.out.println("User "+sk.getPort()+" has disconnected.");
554                             cb.run("User Disconnect "+sk.getPort(),sk);
555                             users[this.id] = null;
556                             break;
557                         }
558
559
560                     }
561                 }
562             } catch (IOException e1) {
563                 e1.printStackTrace();
564             }
565         }
566     }
567 }
568
569 //DataUser.java
570 package KillWolf.Data;
571
```

```
572 import java.net.Socket;
573
574 public class DataUser {
575         public Socket sk;
576         public int port;
577         public String name;
578         public boolean live;
579         public String identify;
580         public boolean hasConfirmID; //被预言家预言了，仅在预言家的客户端可以使用
581         public boolean havePoison = false;
582         public boolean haveAntidote = false;
583 }
584 //SelectCallBack.java
585 package KillWolf.Data;
586
587 public interface SelectCallBack {
588         public void run(String Selection);
589 }
590 // SelectionEvent.java
591 package KillWolf.Data;
592
593 import KillWolf.KillWolf;
594 import SocketServe.sServer;
595
596 public class SelectionEvent {
597         public static SelectionEvent Selection[] = new SelectionEvent[200];
598         public static int SelectionLen = 0;
599         public static void UserQuit(String name) {
600                 for(SelectionEvent i : Selection) {
601                         if(i!=null) {
602                                 if(i.open) {
603                                         for(int j=0;j<i.openusers.length;j++) {
604                                                 if(i.openusers[j].equals(name)) {
605                                                         if(i.target[j] == null) {
606                                                                 i.target[j] = "**NONE**";
607                                                                 i.isFinish();
608                                                                 break;
609                                                         }
610                                                 }
611                                         }
612                                 }
613                         }
614                 }
615         }
```

```
616     public boolean open;
617
618     public String[] users; //要选的目标人
619     public int[] count;
620
621     public String[] openusers; //需要选人的人
622     public String[] target;
623
624     public String ReceiveID;
625     public int timeLimit;
626     public SelectionEventCallBack cb;
627     public sServer server;
628     public SelectionEvent(String ReceiveID,String[] users,int timeLimit,SelectionEventCallBack cb) {
629         this.users = new String[users.length];
630         this.count = new int[users.length];
631         this.cb = cb;
632         this.ReceiveID = ReceiveID;
633         this.timeLimit= timeLimit;
634         for(int i=0;i<users.length;i++)
635             if(users[i] != null) this.users[i] = users[i];
636             else    this.users[i] = "**NONE**";
637     }
638     public void update(String user,String toUser) {
639         for(int i=0;i<users.length;i++) {
640             if(users[i].equals(user)) {
641                 for( int j=0;j<openusers.length;j++) {
642                     if(openusers[j].equals(user)) {
643                         target[j] = toUser;
644                     }
645                 }
646             }
647             if(users[i].equals(toUser))       count[i]++;
648         }
649         System.out.println("Finish ? " + isFinish());
650     }
651     public boolean isFinish() {
652         System.out.println("###### IS FINISH ? " + target.length);
653         for(String i : target) {
654             if(i == null) {
655                 //System.out.println("#######################");
656                 return false;
657             }
658         }
659         closeSelection(this.server,false);
```

```
660             return true;
661         }
662         public int[] getResult() {
663             return count;
664         }
665         public boolean closeSelection(sServer server,boolean run) {
666             //if(run) if(this.cb != null) this.cb.run(ReceiveID, count, users, target);
667             System.out.println("CLOSESELECTION");
668             if(this.open) {
669                 this.open = false;
670                 for(int i=0;i<openusers.length;i++){
671                     if(openusers[i]!=null){
672                         DataUser user = KillWolf.serverData.getUser(openusers[i]);
673                         if(user!=null)
674                         server.sendto("CLOSESELECTION"+ReceiveID,user.sk);
675                     }
676                 }
677                 //没选的默认弃权
678                 if(this.cb != null) this.cb.run(ReceiveID, count, users, target, openusers);
679             }
680             return true;
681         }
682         public void openSeletion(sServer server,DataUser du[],int RealLen,
683 String WindowTitle,String ButtonTitle,String Tip,boolean isDark) {
684             this.open = true;
685             this.server = server;
686             Selection[SelectionLen] = this;
687             SelectionLen++;
688
689
690
691             String STR = "";
692             for(String i : users) {
693                 STR = STR +"\t"+ i;
694             }
695             openusers = new String[RealLen];
696             this.target = new String[RealLen];
697             int j = 0;
698             for(DataUser i : du) {
699                 if(i != null) {
700                     openusers[j] = i.name;
701                     server.sendto("OPENSELETION"+timeLimit+"\t\t"+ReceiveID+"\t"+
702 STR+"\t\t"+WindowTitle+"\t\t"+Tip+"\t\t"+ButtonTitle+"\t\t"+ (isDark?"DARK":"LIGHT"),i.sk);
703  //RECEIVEID \t\t user1 \t user2
```

```
                        j++;
                    }
                }
                ThreadWait a = new ThreadWait(server);
                a.start();
            }
            public class ThreadWait extends Thread
            {
                sServer s;
                public ThreadWait(sServer server) {
                    s = server;
                }
                public void run() {
                    try {
                        Thread.sleep(timeLimit*1000);
                        closeSelection(s,false);
                    }catch(Exception e) {
                        e.printStackTrace();
                    }
                }
            }
        }
        // SelectionEventCallBack.java
        package KillWolf.Data;

        public interface SelectionEventCallBack {
            public void run(String ReceiveID,int[] count,String users[],String target[],String[] openUsers);
        }
        // ClientDeal.java
        package KillWolf.SocketDeal;

        import java.awt.Color;
        import java.io.IOException;
        import java.net.Socket;

        import javax.swing.JFrame;
        import javax.swing.JOptionPane;

        import KillWolf.Data.SelectCallBack;
        import KillWolf.KillWolf;
        import KillWolf.Data.DataUser;
        import KillWolf.Window.GameMain;
        import KillWolf.Window.SelectWindow;
        import SocketServe.CallBack;
```

```
748
749 public class ClientDeal implements CallBack {
750     //public SelectWindow SelectWindow;
751     public void run(String text,Socket sk){
752         System.out.println(sk.getLocalPort()+" Client Receive: "+text+"####");
753         if(text.startsWith("ENTERSUCCESS"))
754         {
755             KillWolf.UserName = text.substring(12);
756             KillWolf.loginwindow.LoginButton.setEnabled(false);
757             KillWolf.loginwindow.userName.setEnabled(false);
758             KillWolf.goWaitRoom();
759         }else if(text.startsWith("ENTERFAIL")){
760             JOptionPane.showMessageDialog(null, text.substring(9));
761             /*try {
762                 sk.close();
763             } catch (IOException e) {
764                 // TODO Auto-generated catch block
765                 e.printStackTrace();
766             }*/
767         }else if(text.startsWith("NEWWAITUSERLIST")){
768             String users[] = text.substring(15).split("\t");
769             //KillWolf.WaitRoom.setList
770             for(int i=0;i<users.length;i++) {
771                 KillWolf.Users[i]= new DataUser();
772                 KillWolf.Users[i].name = users[i];
773             }
774             KillWolf.UsersRealLen = KillWolf.UsersLen = users.length;
775             //KillWolf.WaitRoom.UserList.setListData(users);
776             KillWolf.WaitRoom.setUserCount(users);
777
778             System.out.println(sk.getLocalPort()+" USERS: " + KillWolf.UserName+"\t"
779                                     +KillWolf.UsersRealLen);
780         }else if(text.startsWith("Disconnect from Server")){
781             if(KillWolf.GameMain != null) {
782                 KillWolf.GameMain.getFrame().setAlwaysOnTop(true);
783                 JOptionPane.showMessageDialog(KillWolf.GameMain.getFrame(),
784                     "服务器连接丢失");
785             }else {
786                 JOptionPane.showMessageDialog(null, "服务器连接丢失");
787             }
788             //JOptionPane.setDefaultLocale(null);
789             System.exit(0);
790         }else if(text.startsWith("USERQUIT")){
791             String user = text.substring(8),STR = "";
```

```
792                 for(int i=0;i<KillWolf.Users.length;i++) {
793                     if(KillWolf.Users[i] != null)
794                         if(KillWolf.Users[i].name.equals(user)) {
795                             KillWolf.Users[i] = null;
796                         }else {
797                             STR = STR + "\t" +KillWolf.Users[i].name;
798                         }
799                 }
800                 String users[] = STR.substring(1).split("\t");
801                 //KillWolf.WaitRoom.UserList.setListData();
802                 KillWolf.UsersRealLen--;
803                 //KillWolf.WaitRoom.settitle("玩家数量："+ KillWolf.UsersRealLen)//+"  总数："
804                         +KillWolf.UsersLen);
805                 KillWolf.WaitRoom.setUserCount(users);
806
807                 if(KillWolf.GameMain == null) return;
808
809                 int usdi=0,temp=1;
810                 for(int i=0;i<KillWolf.GameMain.UserOutListData.length;i++){
811                     if(KillWolf.GameMain.UserOutListData[i].startsWith(user+" ")){
812                         temp=0;
813                         KillWolf.GameMain.UserOutListData[i] =
814                                 KillWolf.GameMain.UserOutListData[i] + "(断线)";
815                         break;
816                     }//else usod[i]=KillWolf.GameMain.UserOutListData[i];
817                 }
818
819                 String usod[]=new String[KillWolf.GameMain.UserOutListData.length+temp];
820                 for(int i=0;i<KillWolf.GameMain.UserOutListData.length;i++){
821                     usod[i] = KillWolf.GameMain.UserOutListData[i];
822                 }
823
824                 if(temp==1){
825                     String usda[]=new String[KillWolf.GameMain.UserListData.length-1],
826                             usd[]=new String[KillWolf.GameMain.UserListData.length-1],
827                             usdt[]=new String[KillWolf.GameMain.UserListData.length-1];
828                     usod[usod.length-1] = user+" (断线)";
829                     for(int i=0;i<KillWolf.GameMain.UserListData.length;i++) {
830                         if(KillWolf.GameMain.UserListData[i].equals(user))
831                         {
832                             usod[usod.length-1]  =  usod[usod.length-1]  +
    KillWolf.GameMain.UserListTipData[i];
833                         }else{
834                             usd[usdi]=KillWolf.GameMain.UserListData[i];
```

```
835                             usdt[usdi]=KillWolf.GameMain.UserListTipData[i];
836                             usda[usdi]=usd[usdi]+usdt[usdi];
837                             usdi++;
838                         }
839                     }
840                     KillWolf.GameMain.UserListData = usd;
841                     KillWolf.GameMain.UserListTipData = usdt;
842                     KillWolf.GameMain.getUserList().setListData(usda);
843                 }
844                 KillWolf.GameMain.UserOutListData = usod;
845                 KillWolf.GameMain.getOutUserList().setListData(usod);
846                 KillWolf.GameMain.addToScreen("【系统】用户 "+user+" 退出房间。");
847                 //KillWolf.UsersLen = KillWolf.Users.length;
848             }else if(text.startsWith("GAMESTART")) {
849                 String[] STR = text.substring(9).split("#u#");
850                 DataUser[] du = new DataUser[STR.length];
851                 if(STR[0].equals("TRUE")){
852                     KillWolf.freeSpeak = true;
853                 }else{
854                     KillWolf.freeSpeak = false;
855                 }
856                 for(int i=0;i<STR.length-1;i++) {
857                     du[i] = new DataUser();
858                     String[] p = STR[i+1].split("\t");
859
860                     du[i].name = p[0];
861                     du[i].identify = p[1];
862                 }
863                 KillWolf.Users = du;
864                 KillWolf.goGameMain();
865             }else if(text.startsWith("SELECT")) {
866                 if(KillWolf.SelectWindow != null) KillWolf.SelectWindow.SELECTInfo(text.substring(6));
867             }else if(text.startsWith("OPENSELETION")) {
868                 String a[] = (text.substring(12).split("\t\t")) ;
869                 JFrame fra = null;
870                 if(KillWolf.GameMain!=null) fra = KillWolf.GameMain.getFrame();
871                 KillWolf.SelectWindow = new SelectWindow(a[3],true,a[1],true,fra);
872                 if(a[6].equals("DARK"))KillWolf.SelectWindow.setColorMode(true);
873                 String[] b = a[2].split("\t");//KillWolf.GameMain.UserListData;
874
875                 String myId = KillWolf.GameMain.userSelf.identify;
876                 String[] tip = new String[b.length];
877                 if(myId.equals("狼人")) {//|| myId.equals("预言家")
878                     for (int j = 0; j < b.length; j++) {
```

```
879                             for (int i = 0; i < KillWolf.UsersLen; i++) {
880                                 if (KillWolf.Users[i] != null) {
881                                     if (KillWolf.Users[i].name.equals(b[j])) {
882                                         if(myId.equals("狼人")){
883                                             if(KillWolf.Users[i].identify.equals("狼人")){
884                                                 tip[j] = "("+KillWolf.Users[i].identify+")";
885                                             }
886                                         }/*else       if(myId.equals("  预  言  家  "  )&&
    KillWolf.Users[i].hasConfirmID){
887                                             {
888                                                 if(KillWolf.Users[i].identify.equals("狼人")){
889       “)";
890                                                 }
891                                             }
892
893                                         }*/else{
894                                             tip[j] = "";
895                                         }
896                                     }
897                                 }
898                             }
899                         }
900                 }
901
902                     KillWolf.SelectWindow.start(a[4], a[5], b, true,null, Integer.parseInt(a[0]),tip);
903             }else if(text.startsWith("CLOSESELECTION")) {
904                     KillWolf.SelectWindow.getSelectionAndClose(text.substring(14));
905             }else if(text.startsWith("MESSAGE")) {
906                     if(text.substring(7).startsWith("【系统】天黑请闭眼")){
907                         KillWolf.GameMain.setDark();
908                         KillWolf.GameMain.getFrame().setTitle("狼人杀  "+text.substring(19));
909                         //天黑请闭眼  - Night
910                     }
911                     if(text.substring(7).startsWith("【系统】天亮请睁眼")){
912                         KillWolf.GameMain.getFrame().setTitle("狼人杀  "+text.substring(19));
913                         KillWolf.GameMain.setWhite();
914                     }
915                     KillWolf.GameMain.addToScreen(text.substring(7));
916             }else if(text.startsWith("YUYANJIASTART")){
917                     if(!KillWolf.GameMain.userSelf.identify.equals("预言家"))return;
918
919                     KillWolf.SelectWindow  =  new  SelectWindow("  预  言  ",false,"  预  言  家
    ",true,KillWolf.GameMain.getFrame());
920                     KillWolf.SelectWindow.setColorMode(true);
```

```
921                 //String[] users=new String[]
922                 SelectCallBack cb = Selection -> {
923                     System.out.println("SelectFinal:" + Selection);
924                     //System.out.println("预言："+st);
925                     if(Selection.equals("**NONE**")) {
926                         KillWolf.GameMain.addToScreen("【预言】你没有进行预言。");
927                         KillWolf.SelectWindow.getSelectionAndClose("预言家");
928                         KillWolf.client.send("YUYANEND");
929                         return;
930                     }
931                     String str = "";
932                     for(int i=0;i<KillWolf.UsersLen;i++){
933                         if(KillWolf.Users[i] == null) continue;
934                         if(KillWolf.Users[i].name.equals(Selection)){
935                             for(int j=0;j<KillWolf.GameMain.UserListData.length;j++){
936                                 System.out.println("####################List:"                    +
    KillWolf.GameMain.UserListData[j]+"#");
937                                 if(KillWolf.GameMain.UserListData[j].equals(Selection)){
938                                     str = (KillWolf.Users[i].identify.equals("狼人")?"坏人":"好人");
939                                     KillWolf.GameMain.UserListTipData[j] = "(" + str + ")";
940                                     str = "预言："+KillWolf.Users[i].name+"是"+str;
941                                     KillWolf.SelectWindow.CenterLabel.setText(str);
942                                     KillWolf.SelectWindow.CenterLabel.setForeground(Color.yellow);
943                                     break;
944                                 }
945                             }
946
947                         }
948                     }
949                     String [] usn = KillWolf.GameMain.UserListData;
950                     String [] usnt = KillWolf.GameMain.UserListTipData;
951                     String [] usna = new String[usn.length];
952                     for (int i=0;i<usn.length;i++) {
953                         usna[i]=usn[i]+" "+usnt[i];
954                     }
955                     KillWolf.GameMain.getUserList().setListData(usna);
956                     KillWolf.SelectWindow.getSelectionAndClose("预言家");
957                     KillWolf.GameMain.addToScreen("【预言】"+str.substring(3));
958                     KillWolf.client.send("YUYANEND");
959                     //JOptionPane.showMessageDialog(KillWolf.GameMain.getFrame(),str);
960
961                 };
962                 KillWolf.SelectWindow.start("请选择你要预言的人","预言",
    KillWolf.GameMain.UserListData,true,cb,20,KillWolf.GameMain.UserListTipData);
```

```
963                 SleepThread sth = new SleepThread("预言家");
964                 sth.start();
965
966                 //System.out.println("预言："+st);
967                 //if(st.equals("**NONE**")) {
968             //          KillWolf.GameMain.addToScreen("【预言】你没有进行预言。");
969                 //}
970             }
971             else if(text.startsWith("HUNTERSTART")){
972                 if(!KillWolf.GameMain.userSelf.identify.equals("猎人"))return;
973                 int res=JOptionPane.showConfirmDialog(KillWolf.GameMain.getFrame(), "你已死亡，是
    否使用技能？", "消息", JOptionPane.YES_NO_OPTION);
974                 if(res==JOptionPane.YES_OPTION){
975
    if(text.substring(11).startsWith("LAST"))KillWolf.client.send("HUNTSTARTLASTYES");
976                     else KillWolf.client.send("HUNTSTARTYES");
977                     System.out.println("选择是后执行的代码");      //点击“是”后执行这个代码块
978                 }else{
979
    if(text.substring(11).startsWith("LAST"))KillWolf.client.send("HUNTSTARTLASTYES");
980                     else KillWolf.client.send("HUNTSTARTNO");
981                     System.out.println("选择否后执行的代码");      //点击“否”后执行这个代码块
982                     return;
983                 }
984
985             }else if(text.startsWith("HUNTSELECT")){
986                 if(!KillWolf.GameMain.userSelf.identify.equals("猎人"))return;
987
988                 KillWolf.SelectWindow = new SelectWindow("开枪",false,"猎人技能
    ",true,KillWolf.GameMain.getFrame());
989                 KillWolf.SelectWindow.setColorMode(true);
990
991                 SelectCallBack cb = Selection -> {
992                     System.out.println("SelectFinal:" + Selection);
993                     //System.out.println("预言："+st);
994                     if(Selection.equals("**NONE**")) {
995                         KillWolf.GameMain.addToScreen("【猎人】你没有开枪。");
996                         KillWolf.SelectWindow.getSelectionAndClose("猎人技能");
997
    if(text.substring(10).startsWith("LAST"))KillWolf.client.send("HUNTSELELASTNO");
998                         else KillWolf.client.send("HUNTSELENO");
999                         return;
1000                    }
1001                    KillWolf.GameMain.addToScreen("【猎人】你对"+Selection+"进行射击。");
```

```
1002                    KillWolf.SelectWindow.getSelectionAndClose("猎人技能");
1003
    if(text.substring(10).startsWith("LAST"))KillWolf.client.send("HUNTSELELASTYES"+KillWolf.UserNa
    me+"\t"+Selection);
1004                    else KillWolf.client.send("HUNTSELEYES"+KillWolf.UserName+"\t"+Selection);
1005                };
1006                KillWolf.SelectWindow.start("你要对谁开枪？", "开枪",
    KillWolf.GameMain.UserListData,true,cb,20,null);
1007                SleepThread sth = new SleepThread("猎人技能");
1008                sth.start();
1009            }else if(text.equals("FORBIDSPEAK")){
1010                KillWolf.GameMain.forbidSubmit();
1011            }else if(text.equals("ENABLESPEAK")){
1012                KillWolf.GameMain.ableSubmit();
1013            }else if(text.equals("ENABLESPEAKEND")){
1014                KillWolf.freeSpeak = true;
1015                KillWolf.GameMain.ableSubmit();
1016
1017
1018                String usda[]=new String[KillWolf.GameMain.UserListData.length];
1019                for(int i=0;i<KillWolf.GameMain.UserListData.length;i++) {
1020                    for(int j=0;j<KillWolf.UsersLen;j++){
1021                        if(KillWolf.Users[j]==null)continue;
1022                        if(KillWolf.Users[j].name.equals(KillWolf.GameMain.UserListData[i])){
1023                            usda[i]=KillWolf.GameMain.UserListData[i]+"
    ("+KillWolf.Users[j].identify+")";
1024                        }
1025                    }
1026                }
1027                KillWolf.GameMain.getUserList().setListData(usda);
1028
1029            }else if(text.equals("ENABLESPEAKAC")){
1030                KillWolf.GameMain.ableSubmit();
1031                KillWolf.GameMain.startSpeakClock();
1032            }else if(text.startsWith("WIZARDSTART")) {
1033                if(!KillWolf.GameMain.userSelf.identify.equals("女巫"))return;
1034                String name = text.substring(11);
1035                if(KillWolf.GameMain.userSelf.haveAntidote) {
1036                    if(name.equals("**NONE**")) {
1037                        KillWolf.client.send("WIZARDANTINO");
1038                        return;
1039                    }
1040                    int res=JOptionPane.showConfirmDialog(KillWolf.GameMain.getFrame(), "昨夜
    "+name+"死亡，是否使用解药？", "消息", JOptionPane.YES_NO_OPTION);
```

```
1041                    if(res==JOptionPane.YES_OPTION){
1042                        KillWolf.GameMain.userSelf.haveAntidote = false;
1043                        KillWolf.client.send("WIZARDANTIYES");
1044                        System.out.println("选择是后执行的代码");      //点击“是”后执行这个代码块
1045                    }else{
1046                        KillWolf.client.send("WIZARDANTINO");
1047                        System.out.println("选择否后执行的代码");      //点击“否”后执行这个代码块
1048                        return;
1049                    }
1050                }else {
1051                    KillWolf.client.send("WIZARDANTINONE");
1052                }
1053            }else if(text.startsWith("WIZARDPOS")) {
1054                if(!KillWolf.GameMain.userSelf.identify.equals("女巫"))return;
1055
1056                KillWolf.SelectWindow = new SelectWindow("毒药",false,"女巫毒药",true,KillWolf.GameMain.getFrame());
1057                KillWolf.SelectWindow.setColorMode(true);
1058                //String[] users=new String[]
1059                SelectCallBack cb = Selection -> {
1060                    System.out.println("SelectFinal:" + Selection);
1061                    //System.out.println("预言："+st);
1062                    if(Selection.equals("**NONE**")) {
1063                        KillWolf.GameMain.addToScreen("【女巫】你没有使用毒药。");
1064                        KillWolf.SelectWindow.getSelectionAndClose("女巫毒药");
1065                        KillWolf.client.send("WIZARDPOSNO");
1066                        return;
1067                    }
1068                    KillWolf.GameMain.addToScreen("【女巫】你对"+Selection+"使用了毒药。");
1069                    KillWolf.SelectWindow.getSelectionAndClose("女巫毒药");
1070                    KillWolf.client.send("WIZARDPOSYES"+Selection);
1071                    //JOptionPane.showMessageDialog(KillWolf.GameMain.getFrame(),str);
1072                };
1073                KillWolf.SelectWindow.start("你要对谁使用毒药？","使用",KillWolf.GameMain.UserListData,true,cb,20,KillWolf.GameMain.UserListTipData);
1074                SleepThread sth = new SleepThread("女巫毒药");
1075                sth.start();
1076            }else if(text.startsWith("USERDIED")){
1077                String user = text.substring(8),STR = "";
1078                /*for(int i=0;i<KillWolf.Users.length;i++) {
1079                    if(KillWolf.Users[i] != null)
1080                        if(KillWolf.Users[i].name.equals(user)) {
1081                            KillWolf.Users[i].live = false;
1082                        }else {
```

```
1083                            //STR = STR + "\t" +KillWolf.Users[i].name;
1084                        }
1085                }
1086                String users[] = STR.substring(1).split("\t");
1087                *///KillWolf.WaitRoom.UserList.setListData();
1088                //KillWolf.UsersRealLen--;
1089                //KillWolf.WaitRoom.settitle(" 玩 家 数 量 ： "+ KillWolf.UsersRealLen)//+" 总 数 ：
    "+KillWolf.UsersLen);
1090                //KillWolf.WaitRoom.setUserCount(users);
1091
1092                //if(KillWolf.GameMain == null) return;
1093                int pp=0;
1094                for(;pp<KillWolf.UsersLen;pp++){
1095                    if(KillWolf.Users[pp]==null)continue;
1096                    if(KillWolf.Users[pp].name.equals(user)){
1097                        pp=-1;
1098                        break;
1099                    }
1100                }
1101                if(pp!=-1) return;
1102
1103                String usda[]=new String[KillWolf.GameMain.UserListData.length-1],
1104                        usd[]=new String[KillWolf.GameMain.UserListData.length-1],
1105                        usdt[]=new String[KillWolf.GameMain.UserListData.length-1],
1106                        usod[]=new String[KillWolf.GameMain.UserOutListData.length+1];
1107                int usdi=0;
1108                for(int i=0;i<KillWolf.GameMain.UserOutListData.length;i++){
1109                    usod[i]=KillWolf.GameMain.UserOutListData[i];
1110                }
1111                usod[usod.length-1] = user+" (淘汰)";
1112                if(user.equals(KillWolf.UserName)){
1113
    KillWolf.GameMain.getIdentity().setText(KillWolf.UserName+"("+KillWolf.GameMain.userSelf.identify
    +")，你已被淘汰");
1114                }
1115                for(int i=0;i<KillWolf.GameMain.UserListData.length;i++) {
1116                    if(KillWolf.GameMain.UserListData[i].equals(user))
1117                    {
1118                        usod[usod.length-1]  =  usod[usod.length-1]  +
    KillWolf.GameMain.UserListTipData[i];
1119                    }else{
1120                        usd[usdi]=KillWolf.GameMain.UserListData[i];
1121                        usdt[usdi]=KillWolf.GameMain.UserListTipData[i];
1122                        usda[usdi]=usd[usdi]+usdt[usdi];
```

```
1123                            usdi++;
1124                        }
1125                    }
1126
1127                    KillWolf.GameMain.UserListData = usd;
1128                    KillWolf.GameMain.UserListTipData = usdt;
1129                    KillWolf.GameMain.UserOutListData = usod;
1130                    KillWolf.GameMain.getUserList().setListData(usda);
1131                    KillWolf.GameMain.getOutUserList().setListData(usod);
1132//                  System.out.println(text);
1133//                  String[] result = text.substring(8).split("\t");
1134//                  String[] users = new String[KillWolf.GameMain.UserListData.length- result.length];
1135//                                                                  String[] outUsers = new
    String[KillWolf.GameMain.UserOutListData.length+result.length];
1136//                  for(int j=0;j< result.length;j++){
1137//
    if(result[j].equals(KillWolf.GameMain.userSelf.name))KillWolf.GameMain.userSelf.live=false;
1138//                        for(int k=0;k< KillWolf.GameMain.UserListData.length;k++){
1139//                            if(KillWolf.GameMain.UserListData[k].startsWith(result[j])){
1140//                                result[j]=KillWolf.GameMain.UserListData[k];
1141//                                KillWolf.GameMain.UserListData[k]=null;
1142//                            }
1143//                        }
1144//                  }
1145//                  int i=0;
1146//                  for (;i<KillWolf.GameMain.UserOutListData.length;i++){
1147//                      outUsers[i] = KillWolf.GameMain.UserOutListData[i];
1148//                  }
1149//                  int j=i;
1150//                  for(;i<outUsers.length;i++){
1151//                      outUsers[i]=result[i-j]+"(淘汰)";
1152//                  }
1153//                  KillWolf.GameMain.UserOutListData=outUsers;
1154//                  KillWolf.GameMain.getOutUserList().setListData(outUsers);
1155//                  int i2 = 0;
1156//                  for(int i1 = 0;i1< KillWolf.GameMain.UserListData.length;i1++){
1157//
    if(KillWolf.GameMain.UserListData[i1]!=null&& !KillWolf.GameMain.UserListData[i1].equals("**NON
    E**")){
1158//                            users[i2++]=KillWolf.GameMain.UserListData[i1];
1159//                        }
1160//                  }
1161//                  KillWolf.GameMain.UserListData = users;
1162//                  KillWolf.GameMain.getUserList().setListData(users);
```

```
1163                }
1164
1165
1166
1167                    //      +timeLimit+"\t\t"+ReceiveID+"\t"+STR,du[j].sk); //RECEIVEID \t\t user1 \t user2
1168
1169        }
1170
1171}
1172
1173
1174class SleepThread extends Thread{
1175        String sss;
1176        public SleepThread(String a) {
1177                this.sss = a;
1178        }
1179        public void run() {
1180                try {
1181                        Thread.sleep(20*1000);
1182                } catch (InterruptedException e) {
1183                        e.printStackTrace();
1184                }
1185                KillWolf.SelectWindow.getSelectionAndClose(sss);
1186        }
1187}
1188// GameMainThread.java
1189package KillWolf.SocketDeal;
1190
1191import KillWolf.KillWolf;
1192import KillWolf.Data.DataUser;
1193import KillWolf.Data.SelectionEvent;
1194import KillWolf.Window.GameMain;
1195
1196public class GameMainThread extends Thread{
1197        public boolean YuYanCompleted = false;
1198        public boolean WizardCompleted = false;
1199        public void run() {
1200                KillWolf.serverData.deathUser = "**NONE**";
1201                KillWolf.serverData.deathUser2 = "**NONE**";
1202                KillWolf.serverData.deathUser3 =    "**NONE**";
1203                KillWolf.serverData.deathUser4 =    "**NONE**";
1204                gameNight();
1205        }
1206        public void gameNight(){
```

```
1207            if(KillWolf.serverData.GameMainThread.isWin()) return;
1208
1209            KillWolf.serverData.Day++;
1210            KillWolf.serverData.night = true;
1211
1212
1213            KillWolf.server.send("MESSAGE【系统】天黑请闭眼 - Night"+KillWolf.serverData.Day);
1214            KillWolf.server.send("FORBIDSPEAK");
1215
1216            //KillWolf.server.send("ENABLESPEAK");
1217
1218            String[] allUserName = (String[])KillWolf.serverData.getLivingUsers(true);
1219            //给狼人弹窗
1220            DataUser[] wolfUser0 = KillWolf.serverData.getUsersWithID("狼人");
1221            DataUser[] wolfUser = new DataUser[wolfUser0.length];
1222            int wi = 0;
1223            for(int k=0;k<wolfUser0.length;k++) {
1224                if(wolfUser0[k].live) {
1225                    wolfUser[wi] = wolfUser0[k];
1226                    wi++;
1227                }
1228            }
1229            //String[] wolfUserName = KillWolf.serverData.ArrayDatatoName(wolfUser);
1230
1231            try {
1232                Thread.sleep(2*1000);
1233            } catch (InterruptedException e) {
1234                e.printStackTrace();
1235            }
1236            KillWolf.server.send("MESSAGE【系统】狼人请睁眼，请选择你要暗杀的目标");
1237            KillWolf.serverData.Selection[KillWolf.serverData.SelectionLen] = new
    SelectionEvent(Integer.toString(KillWolf.serverData.SelectionLen),allUserName,30,new WolfSelect());
1238            KillWolf.serverData.Selection[KillWolf.serverData.SelectionLen].openSeletion(KillWolf.server,
    wolfUser, wi,"狼人出没","暗杀他","请选择你要暗杀的人",true);
1239            KillWolf.serverData.SelectionLen++;
1240            //预言家弹窗
1241            KillWolf.server.send("MESSAGE【系统】预言家请睁眼，请选择你要预言的目标");
1242            DataUser d[] = KillWolf.serverData.getUsersWithID("预言家");
1243            if (d.length == 1 && d[0].live) {
1244
1245                YuYanCompleted = false;
1246                KillWolf.server.sendto("YUYANJIASTART",d[0].sk);
1247            }else YuYanCompleted = true;
1248
```

```
1249            //女巫弹窗在狼人结束才开始，此处先置为 false
1250            WizardCompleted = false;
1251
1252
1253            //等待预言、女巫完成再进入白天，进入白天自动进行判断是否能进入，在女巫结束还会调
    用一次。
1254            try {
1255                Thread.sleep(20*1000);
1256                YuYanCompleted = true;//等 20 秒，预言一定已经完成了
1257                gameDay();
1258            } catch (InterruptedException e) {
1259                // TODO Auto-generated catch block
1260                e.printStackTrace();
1261            }
1262        }
1263
1264        public void gameWizard(String diedUser) {
1265            DataUser d[] = KillWolf.serverData.getUsersWithID("女巫");
1266            //System.out.println(null.length);
1267            if (d.length == 1    && d[0].live) {
1268                WizardCompleted = false;
1269                KillWolf.server.sendto("WIZARDSTART"+diedUser,d[0].sk);
1270            }else {
1271                if(!diedUser.equals("**NONE**")) {
1272                    KillWolf.serverData.getUser(diedUser).live = false;
1273                }
1274                WizardCompleted = true;
1275                KillWolf.serverData.GameMainThread.gameDay();
1276            }
1277        }
1278        boolean isRunGameDay = false;
1279        public void gameDay() {
1280            if(isRunGameDay) return;
1281            if(!YuYanCompleted || !WizardCompleted) return;
1282            //如果都完成了则继续
1283            String death = KillWolf.serverData.deathUser;
1284            String death2 = KillWolf.serverData.deathUser2;
1285            isRunGameDay = true;
1286
1287            KillWolf.server.send("MESSAGE【系统】天亮请睁眼  - Day"+KillWolf.serverData.Day);
1288            try {
1289                Thread.sleep(1000);
1290            } catch (InterruptedException e) {
1291                e.printStackTrace();
```

```
1292            }
1293            String death3 = KillWolf.serverData.deathUser3;
1294            if(death2.equals("**NONE**") && death.equals("**NONE**")) {
1295                KillWolf.server.send("MESSAGE【系统】昨夜无人死亡");
1296            }else {
1297                if(!death.equals("**NONE**")){
1298                    DataUser u = KillWolf.serverData.getUser(death);
1299                    if(u!=null) u.live =false;
1300                    KillWolf.server.send("USERDIED"+death);
1301                }
1302                if(!death2.equals("**NONE**")){
1303                    DataUser u = KillWolf.serverData.getUser(death2);
1304                    if(u!=null) u.live =false;
1305                    KillWolf.server.send("USERDIED"+death2);
1306                }
1307                if(!death3.equals("**NONE**")){
1308                    DataUser u = KillWolf.serverData.getUser(death3);
1309                    if(u!=null) u.live =false;
1310                    KillWolf.server.send("USERDIED"+death3);
1311                }
1312
    //KillWolf.server.send("USERDIED"+(death.equals("**NONE**")?"":death)+(death2.equals("**NONE**
    ")?"":("\t"+death2))+(death3.equals("**NONE**")?"":("\t"+death3)));
1313                if(death2.equals("**NONE**"))KillWolf.server.send("MESSAGE【系统】昨夜
    "+(death.equals("**NONE**")?"":death)+(death3.equals("**NONE**")?"":("、"+death3))+"死亡");
1314                else KillWolf.server.send("MESSAGE【系统】昨夜
    "+(death.equals("**NONE**")?"":death) +(death2.equals("**NONE**")?"":("、
    "+death2))+(death3.equals("**NONE**")?"":("、"+death3))+"死亡");
1315            }
1316
1317            if(KillWolf.serverData.GameMainThread.isWin())return;
1318
1319            //判断猎人是否存活，若死亡，向猎人弹窗，使用技能
1320            DataUser user = KillWolf.serverData.getUser(death);
1321            DataUser user2 = KillWolf.serverData.getUser(death2);
1322            if(user == null && user2 == null) dayWork();
1323            else if(user != null && user.identify.equals("猎人
    ")){KillWolf.server.sendto("HUNTERSTART",user.sk);}
1324            else if(user2 != null && user2.identify.equals("猎人
    ")){KillWolf.server.sendto("HUNTERSTART",user2.sk);}
1325            else dayWork();
1326
1327            //向所有玩家弹窗，投票
1328
```

```
1329            //向公屏输出，投票结果，若平票，则再次投票
1330
1331            //再次判断猎人是否存活，若死亡，向猎人弹窗
1332
1333//            }
1334            //KillWolf.server.send("MESSAGE【系统】"+whoWin+"胜，将在 5 秒后退出该窗口");
1335            //重置游戏
1336        }
1337        class TimeTha extends Thread{
1338            public void run() {
1339                try {
1340                    Thread.sleep(1000*120);
1341                    gameVote();
1342                }catch(Exception e) {
1343                    e.printStackTrace();
1344                }
1345            }
1346        }
1347        public void dayWork (){
1348            String death = KillWolf.serverData.deathUser;
1349            String death2 = KillWolf.serverData.deathUser2;
1350            //String death3 = KillWolf.serverData.deathUser3;
1351            String firstdeath = "";
1352
1353
1354            //}else if(death2.equals("**NONE**") && !death.equals("**NONE**")){
1355//                //KillWolf.server.send("MESSAGE【系统】昨夜"+death+"死亡");
1356//                firstdeath = death;
1357//            }else if(death2.equals("**NONE**") && death.equals("**NONE**")) {
1358//                //KillWolf.server.send("MESSAGE【系统】昨夜无人死亡");
1359//            }
1360//            else if(!death2.equals("**NONE**") && !death.equals("**NONE**")){
1361//                firstdeath = death;
1362//                //KillWolf.server.send("MESSAGE【系统】昨夜"+death+"、"+death2+"死亡");
1363//            }
1364            if(!death.equals("**NONE**") ){
1365                firstdeath = death;
1366            }else if(!death2.equals("**NONE**") ){
1367                firstdeath = death2;
1368            }
1369
1370            if(KillWolf.serverData.GameMainThread.isWin())return;
1371            //判断是否获胜
1372
```

```
1373            //玩家轮流发言
1374            if(KillWolf.serverData.freeSpeak){
1375                KillWolf.server.send("MESSAGE【系统】现在有 120 秒的交流时间");
1376                KillWolf.server.send("ENABLESPEAKAC");
1377                TimeTha spt = new TimeTha();
1378                spt.start();
1379                return;
1380            }
1381            KillWolf.server.send("MESSAGE【系统】进入轮流发言阶段，每人有 30 秒发言时间");
1382            try {
1383                Thread.sleep(2000);
1384            } catch (InterruptedException e) {
1385                e.printStackTrace();
1386            }
1387            DataUser user = null;//(DataUser[]) KillWolf.serverData.getUser(firstdeath);
1388
1389            if(firstdeath.equals("")) for(int j=0;j<KillWolf.serverData.UsersLen;j++){
1390                if(KillWolf.serverData.ServerUsers[j] != null)
1391                    if(KillWolf.serverData.ServerUsers[j].live)                    firstdeath                    =
    KillWolf.serverData.ServerUsers[j].name;
1392            }
1393            for(int i = 0;i<KillWolf.serverData.UsersLen;i++){
1394                if(KillWolf.serverData.ServerUsers[i]==null) continue;
1395
1396                if(firstdeath.equals(KillWolf.serverData.ServerUsers[i].name)){
1397                    int j=i+1;
1398                    for(;j<KillWolf.serverData.UsersLen;j++){
1399                        if(KillWolf.serverData.ServerUsers[j] != null)
1400                            if(KillWolf.serverData.ServerUsers[j].live)                user                =
    KillWolf.serverData.ServerUsers[j];
1401                    }
1402                    if(user == null) for(j=0;j<i;j++){
1403                        if(KillWolf.serverData.ServerUsers[j] != null)
1404                            if(KillWolf.serverData.ServerUsers[j].live)                user                =
    KillWolf.serverData.ServerUsers[j];
1405                    }
1406                    if(user == null) {
1407                        KillWolf.serverData.GameMainThread.gameNext();
1408                        System.out.println("ERROR !!!!!!!!!!!!");
1409                        return;
1410                    }
1411                    KillWolf.serverData.StartSpeakName = user.name;
1412                    KillWolf.serverData.nowSpeakUser = user.name;
1413                    KillWolf.server.send("MESSAGE【系统】轮到"+user.name+"发言");
```

```
1414                        KillWolf.server.sendto("ENABLESPEAKAC",user.sk);
1415
1416                        SpeakThread spt = new SpeakThread(user.name,KillWolf.serverData.Day);
1417                        spt.start();
1418                        break;
1419                    }
1420                    //if(i==0)
1421
1422            }
1423        }
1424
1425        public void gameVote() {
1426            KillWolf.server.send("MESSAGE【系统】发言结束，现在开始进行投票");
1427
1428            //投票弹窗
1429            String[] allUserName = (String[])KillWolf.serverData.getLivingUsers(true);
1430            DataUser[] allUser = (DataUser[]) KillWolf.serverData.getLivingUsers(false);
1431            //String[] wolfUserName = KillWolf.serverData.ArrayDatatoName(wolfUser);
1432
1433            KillWolf.serverData.Selection[KillWolf.serverData.SelectionLen] = new
    SelectionEvent(Integer.toString(KillWolf.serverData.SelectionLen),allUserName,20,new VoteSelection());
1434            KillWolf.serverData.Selection[KillWolf.serverData.SelectionLen].openSeletion(KillWolf.server,
    allUser, allUser.length,"开始投票","投他","请选择",false);
1435            KillWolf.serverData.SelectionLen++;
1436
1437        }
1438        public void gameHunt(String death) {
1439
1440            DataUser user = KillWolf.serverData.getUser(death);
1441            if (user != null && user.identify.equals("猎人")) {
1442                KillWolf.server.sendto("HUNTERSTARTLAST", user.sk);
1443            }
1444        }
1445        public void gameNext(){
1446            if(!isWin()){
1447                (KillWolf.serverData.GameMainThread = new GameMainThread()).start();
1448            }
1449        }
1450        public boolean isWin(){
1451            String whoWin = KillWolf.serverData.whoWin();
1452            if(!(whoWin.equals("无"))){
1453                KillWolf.server.send("MESSAGE【系统】游戏结束，"+whoWin+"胜");
1454
1455                gameEnd();
```

```
1456                    return true;
1457                    //KillWolf.server.send("MESSAGE【系统】将在 5 秒后关闭窗口");
1458                }
1459                return false;
1460        }
1461        public void gameEnd(){
1462                KillWolf.server.send("MESSAGE【系统】游戏统计");
1463                DataUser[] d = KillWolf.serverData.ServerUsers;
1464                for(int i=0;i<KillWolf.serverData.UsersLen;i++){
1465                    if(d[i]!=null){
1466                        KillWolf.server.send("MESSAGE【系统】"+d[i].name+" "+d[i].identify+" "+(d[i].live?"
    存活":"淘汰"));
1467                    }
1468                }
1469                KillWolf.serverData.freeSpeak = true;
1470                KillWolf.server.send("ENABLESPEAKEND");
1471
1472        }
1473}
1474// ServerDeal.java
1475package KillWolf.SocketDeal;
1476
1477import java.net.Socket;
1478import java.security.PublicKey;
1479
1480import KillWolf.Data.SelectionEventCallBack;
1481import KillWolf.KillWolf;
1482import KillWolf.Data.DataUser;
1483import KillWolf.Data.SelectionEvent;
1484import KillWolf.KillWolf.serverData;
1485import SocketServe.CallBack;
1486import com.sun.source.tree.ReturnTree;
1487
1488public class ServerDeal implements CallBack {
1489        public void run(String text,Socket sk){
1490                System.out.println(sk.getPort()+" Server Receive: "+text+"####");
1491                if(text.startsWith("ENTER")){
1492                    if(KillWolf.GameMain!=null) {
1493                        KillWolf.server.sendto("ENTERFAIL 游戏已经开始，请稍后再试",sk);
1494                        return;
1495                    }
1496                    String name = text.substring(5);
1497                    for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
1498                        if(KillWolf.serverData.ServerUsers[i]!=null)
```

```
1499                        if(name.equals(KillWolf.serverData.ServerUsers[i].name)) {
1500                            //System.out.println("NAME: "+name+" "+i);
1501                            KillWolf.server.sendto("ENTERFAIL 当前房间存在重名，请更换用户名后再尝
    试连接。",sk);
1502                            return;
1503                        }
1504                    }
1505
    KillWolf.server.sendto("ENTERSUCCESS["+(KillWolf.serverData.UsersLen+1)+"]"+name,sk);
1506                    KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen] = new DataUser();
1507
    KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen].name="["+(KillWolf.serverData.UsersLe
    n+1)+"]"+name;
1508                    KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen].sk=sk;
1509                    KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen].port=sk.getPort();
1510                    KillWolf.serverData.UsersLen++;
1511                    KillWolf.serverData.UsersRealLen++;
1512                    String STR = "";
1513                    for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
1514                        if(KillWolf.serverData.ServerUsers[i]!=null)
1515                        STR = STR +'\t'+ KillWolf.serverData.ServerUsers[i].name;
1516                    }
1517                    KillWolf.server.send("NEWWAITUSERLIST"+STR.substring(1));
1518                    return;
1519                }
1520                else if(text.equals("GETUSERS")) {
1521                    String STR = "";
1522                    for(DataUser i : KillWolf.serverData.ServerUsers) {
1523                        if(i!=null)
1524                        STR = STR +'\t'+ i.name;
1525                    }
1526                    KillWolf.server.send(STR.substring(1));
1527                    return;
1528                }else if(text.startsWith("User Disconnect ")) {
1529                    int port = Integer.parseInt(text.substring(16));
1530                    for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
1531                        if(KillWolf.serverData.ServerUsers[i]!=null)
1532                        if(port == KillWolf.serverData.ServerUsers[i].port) {
1533                            System.out.println("NAME QUIT: "+KillWolf.serverData.ServerUsers[i].name);
1534
    KillWolf.server.send("USERQUIT"+KillWolf.serverData.ServerUsers[i].name);//KillWolf.server.sendto("
    ENTERFAIL",sk);
1535                            SelectionEvent.UserQuit(KillWolf.serverData.ServerUsers[i].name);
1536                            KillWolf.serverData.ServerUsers[i] = null;
```

```
1537                        KillWolf.serverData.UsersRealLen--;
1538                        //添加选择时用户退出
1539                        return;
1540                    }
1541                }
1542
1543
1544            }else if(text.startsWith("SELECT")) {
1545                KillWolf.server.send(text);
1546            }else if(text.startsWith("FINALSELECT")) {
1547                String s[] = text.substring(11).split("\t");
1548                SelectionEvent se = KillWolf.serverData.Selection[Integer.parseInt(s[0])];
1549                se.update(s[1],s[2]);
1550            }else if(text.startsWith("GAMEREADY")){
1551                serverData.getUser(text.substring(9)).live = true;
1552                for(int i=0;i<serverData.UsersLen;i++){
1553                    if(serverData.ServerUsers[i]!=null)
1554                    if(!serverData.ServerUsers[i].live) return;
1555                }
1556                KillWolf.server.send("MESSAGE【系统】玩家全部进入，游戏将于 3 秒后开始");
1557                try {
1558                    Thread.sleep(3000);
1559                } catch (InterruptedException e) {
1560                    e.printStackTrace();
1561                }
1562                KillWolf.serverData.Day = 0;
1563                KillWolf.serverData.night = false;
1564
1565                (KillWolf.serverData.GameMainThread = new GameMainThread()).start();
1566            }else if(text.startsWith("USERSPEAK")){
1567                String[] submitMsg = text.substring(9).split("\t");
1568                KillWolf.server.send("MESSAGE【玩家】"+submitMsg[0]+" ： "+submitMsg[1]);
1569                if(!KillWolf.serverData.freeSpeak) nextSpeak(submitMsg[0],serverData.Day);
1570
1571            }else if(text.startsWith("YUYANEND")){
1572                KillWolf.serverData.GameMainThread.YuYanCompleted = true;
1573                KillWolf.serverData.GameMainThread.gameDay();
1574            }else if(text.startsWith("WIZARDANTI")) {
1575                String result = text.substring(10);
1576                if(result.equals("YES")) {
1577                    DataUser u = KillWolf.serverData.getUser(KillWolf.serverData.deathUser);
1578                    if(u!=null) u.live = true;
1579                    KillWolf.server.sendto("MESSAGE【女巫】你为"+KillWolf.serverData.deathUser+"使
    用了解药。", sk);
```

```
1580                    KillWolf.serverData.deathUser = "**NONE**";
1581                    KillWolf.serverData.GameMainThread.WizardCompleted = true;
1582                    KillWolf.serverData.GameMainThread.gameDay();
1583                }else if(result.equals("NO")){
1584                    if(!KillWolf.serverData.deathUser.equals("**NONE**")) {
1585                        DataUser u = KillWolf.serverData.getUser(KillWolf.serverData.deathUser);
1586                        if(u!=null) u.live = false;
1587                        KillWolf.server.sendto("MESSAGE 【 女 巫 】 昨 夜 "+KillWolf.serverData.deathUser+"死亡，你没有使用解药。", sk);
1588                    }
1589                    else KillWolf.server.sendto("MESSAGE【女巫】昨夜无人死亡。", sk);
1590                    KillWolf.server.sendto("WIZARDPOS", sk);
1591                }else if(result.equals("NONE")){
1592                    if(!KillWolf.serverData.deathUser.equals("**NONE**")){
1593                        DataUser u = KillWolf.serverData.getUser(KillWolf.serverData.deathUser);
1594                        if(u!=null) u.live = false;
1595                    }
1596                    KillWolf.server.sendto("WIZARDPOS", sk);
1597                }
1598            }else if(text.startsWith("WIZARDPOS")) {
1599                String result = text.substring(9);
1600                KillWolf.serverData.deathUser2 = "**NONE**";
1601                if(result.startsWith("YES")) {
1602                    DataUser u = KillWolf.serverData.getUser(result.substring(3));
1603                    if(u!=null) u.live = false;
1604                    KillWolf.serverData.deathUser2 = result.substring(3);
1605                }else if(result.equals("NO")){
1606
1607                }else if(result.equals("NONE")){
1608
1609                }
1610                KillWolf.serverData.GameMainThread.WizardCompleted = true;
1611                KillWolf.serverData.GameMainThread.gameDay();
1612            }
1613            else if(text.startsWith("HUNTSTART")){
1614                String result = text.substring(9);
1615                if(result.equals("YES")){
1616                    KillWolf.server.sendto("HUNTSELECT", sk);
1617                }else if(result.equals("NO")){
1618                    KillWolf.server.sendto("MESSAGE【猎人】你已死亡，没有使用技能。",sk);
1619                    KillWolf.serverData.GameMainThread.dayWork();
1620                }else if(result.startsWith("LAST")){
1621                    if(result.substring(4).equals("YES")){
1622                        KillWolf.server.sendto("HUNTSELECTLAST", sk);
```

```
1623                    }else if(result.substring(4).equals("NO")) {
1624                        KillWolf.server.sendto("MESSAGE【猎人】你已死亡，没有使用技能。", sk);
1625                        serverData.GameMainThread.gameNext();
1626                    }       //KillWolf.serverData.GameMainThread.dayWork();
1627                }
1628            }else if(text.startsWith("HUNTSELE")){
1629                //System.out.println("猎人技能成功#########");
1630                String result = text.substring(8);
1631                if(result.startsWith("YES")) {
1632                    String[] u = result.substring(3).split("\t");
1633                    DataUser us = KillWolf.serverData.getUser(u[1]);
1634                    if(us!=null) us.live = false;
1635                    KillWolf.serverData.deathUser3 = u[1];
1636                    System.out.println("猎人技能成功#########");
1637                    KillWolf.server.send("MESSAGE【猎人】"+u[0]+"死前开枪杀死了"+
    serverData.deathUser3);
1638                    KillWolf.server.send("USERDIED"+serverData.deathUser3);
1639                    KillWolf.serverData.GameMainThread.dayWork();
1640                }
1641                else if(result.equals("NO")){
1642                    KillWolf.serverData.GameMainThread.dayWork();
1643                }else if (result.startsWith("LAST")){
1644                    result = result.substring(4);
1645                    if(result.startsWith("YES")) {
1646                        String[] u = result.substring(3).split("\t");
1647                        DataUser us = KillWolf.serverData.getUser(u[1]);
1648                        if(us!=null) us.live = false;
1649                        //KillWolf.serverData.getUser(u[1]).live = false;
1650                        KillWolf.serverData.deathUser3 = u[1];
1651                        System.out.println("猎人技能成功#########");
1652                        KillWolf.server.send("MESSAGE【猎人】"+u[0]+"死前开枪杀死了"+
    serverData.deathUser3);
1653                        KillWolf.server.send("USERDIED"+serverData.deathUser3);
1654                        //KillWolf.serverData.GameMainThread.dayWork();
1655                        try {
1656                            Thread.sleep(2000);
1657                        } catch (InterruptedException e) {
1658                            e.printStackTrace();
1659                        }
1660                    }
1661                    else if(result.equals("NO")) {
1662                    }
1663                    serverData.GameMainThread.gameNext();
1664                }
```

```
1665
1666            }
1667
1668            //KillWolf.server.send("["+sk.getPort()+"]"+text);
1669        }
1670        public static boolean nextSpeak(String str,int Day){
1671            System.out.println("NOW SPEAK:"+KillWolf.serverData.nowSpeakUser+" CALLBACK:"+str);
1672            if(!KillWolf.serverData.nowSpeakUser.equals(str) || Day != serverData.Day) return false;
1673            else {
1674                DataUser[] liveUsers = (DataUser[]) KillWolf.serverData.getLivingUsers(false);
1675                String lastSpeak = str;
1676                if(serverData.StartSpeakName.equals("**VOTE**")) return true;
1677                for(int i = 0;i<liveUsers.length;i++){
1678                    if(lastSpeak.equals(liveUsers[i].name)){
1679                        KillWolf.server.sendto("FORBIDSPEAK",liveUsers[i].sk);
1680
1681                        if(i==liveUsers.length-1) i=-1;
1682                        i=i+1;
1683
1684                        System.out.println("CHANGE  SPEAK  "+KillWolf.serverData.nowSpeakUser+"
    TO "+liveUsers[i].name);
1685                        if(liveUsers[i].name.equals(serverData.StartSpeakName)){
1686                            //一轮发言结束
1687                            serverData.StartSpeakName = "**VOTE**";
1688                            KillWolf.serverData.GameMainThread.gameVote();
1689                            return true;
1690                        }
1691                        KillWolf.serverData.nowSpeakUser = liveUsers[i].name;
1692                        KillWolf.server.send("MESSAGE【系统】轮到"+liveUsers[i].name+"发言");
1693                        KillWolf.server.sendto("ENABLESPEAKAC",liveUsers[i].sk);
1694                        SpeakThread spt = new SpeakThread(liveUsers[i].name, serverData.Day);
1695                        spt.start();
1696                        return true;
1697                    }
1698                    //if(i==0)
1699
1700                }
1701                //找遍了所有人，但还没找到上一个发言人，可能已经退出，从第一个人开始重新发
    言。
1702                KillWolf.server.send("MESSAGE【系统】有用户退出，发言阶段重新开始。");
1703                for(int j=0;j<KillWolf.serverData.UsersLen;j++){
1704                    if(KillWolf.serverData.ServerUsers[j] != null)
1705                        if(KillWolf.serverData.ServerUsers[j].live) {
1706                            serverData.StartSpeakName = KillWolf.serverData.ServerUsers[j].name;
```

```
1707                    KillWolf.serverData.nowSpeakUser = KillWolf.serverData.ServerUsers[j].name;
1708                    KillWolf.server.send("MESSAGE 【系统】轮到"+KillWolf.serverData.ServerUsers[j].name+"发言");
1709                    KillWolf.server.sendto("ENABLESPEAKAC",KillWolf.serverData.ServerUsers[j].sk);
1710                    SpeakThread spt = new SpeakThread(KillWolf.serverData.ServerUsers[j].name, serverData.Day);
1711                    spt.start();
1712                    break;
1713                }
1714            }
1715            return true;
1716        }
1717    }
1718
1719
1720}
1721
1722class SpeakThread extends Thread{
1723    String user;
1724    int Day;
1725    public SpeakThread (String user,int Day){
1726        this.user = user;
1727        this.Day = Day;
1728    }
1729    @Override
1730    public void run() {
1731        try {
1732            Thread.sleep(30*1000);
1733        }catch (InterruptedException e){
1734            e.printStackTrace();
1735        }
1736        ServerDeal.nextSpeak(user,Day);
1737    }
1738
1739
1740}
1741class WolfSelect implements SelectionEventCallBack {
1742    public void run(String ReceiveID,int[] count,String users[],String target[],String[] openusers) {
1743        String name = "**NONE**",sameName = "**NONE**";
1744        int max = 0,samemax = 0;
1745        System.out.println("####狼人投票--------");
1746        String STR = "Night"+KillWolf.serverData.Day;
```

```
1747            for(int i=0;i<users.length;i++){
1748                if(max < count[i]){
1749                    max = count[i];
1750                    name = users[i];
1751                }else if(max == count[i]) {
1752                    sameName = users[i];
1753                    samemax = max;
1754                }
1755                System.out.println("#"+users[i]+" "+count[i]);
1756                if(count[i] == 0) continue;
1757                STR = STR +"\n"+ "("+count[i]+")"+users[i];
1758                boolean first = true;
1759                for(int j=0;j<target.length;j++) {
1760                    if(target[j] != null)
1761                    if(target[j].equals(users[i])) {
1762                        if(first) {
1763                            first = false;
1764                            STR = STR + " <- "+openusers[j];
1765                        }else STR = STR +"、"+openusers[j];
1766                    }
1767                }
1768            }
1769            System.out.println(name + "--------"+ max);
1770//            KillWolf.serverData.getUser(name).live = false;
1771            String tip ="";
1772            if(name == "**NONE**") {
1773                name = "存在平票，无人出局";
1774                KillWolf.serverData.deathUser="**NONE**";
1775            }else if(samemax == max){
1776                name = "存在平票，无人出局";
1777                KillWolf.serverData.deathUser="**NONE**";
1778            }else {
1779                tip = "被狼人暗杀";
1780                KillWolf.serverData.deathUser=name;
1781                //KillWolf.serverData.getUser(KillWolf.serverData.deathUser).live = false;
1782            }
1783            DataUser[] wu = KillWolf.serverData.getUsersWithID("狼人");
1784            String[] s = STR.split("\n");
1785            for(int i=0;i<wu.length;i++) {
1786                for(int j=0;j<s.length;j++) {
1787                    KillWolf.server.sendto("MESSAGE【狼人投票】"+s[j],wu[i].sk);
1788                }
1789                KillWolf.server.sendto("MESSAGE【最终结果】"+name+"("+max+")"+tip,wu[i].sk);
1790            }
```

```
1791            KillWolf.serverData.GameMainThread.gameWizard(KillWolf.serverData.deathUser);
1792        }
1793}
1794class VoteSelection implements SelectionEventCallBack {
1795
1796        public void run(String ReceiveID,int[] count,String users[],String target[],String[] openusers) {
1797            String name = "**NONE**",sameName = "**NONE**";
1798            int max = 0,samemax = 0;
1799            System.out.println("####投票--------");
1800            String STR = "投票结束";
1801            for(int i=0;i<users.length;i++){
1802                if(max < count[i]){
1803                    max = count[i];
1804                    name = users[i];
1805                }else if(max == count[i]) {
1806                    sameName = users[i];
1807                    samemax = max;
1808                }
1809                System.out.println("#"+users[i]+" "+count[i]);
1810                if(count[i] == 0) continue;
1811                STR = STR +"\n"+ "("+count[i]+")"+users[i];
1812                boolean first = true;
1813                for(int j=0;j<target.length;j++) {
1814                    if(target[j] != null)
1815                    if(target[j].equals(users[i])) {
1816                        if(first) {
1817                            first = false;
1818                            STR = STR + " <- "+openusers[j];
1819                        }else STR = STR +"、"+openusers[j];
1820                    }
1821                }
1822            }
1823            System.out.println(name + "--------"+ max);
1824//            KillWolf.serverData.getUser(name).live = false;
1825            String tip ="";
1826            if(name == "**NONE**") {
1827                name = "存在平票，无人出局";
1828                KillWolf.serverData.deathUser4="**NONE**";
1829            }else if(samemax == max){
1830                name = "存在平票，无人出局";
1831                KillWolf.serverData.deathUser4="**NONE**";
1832            }else {
1833                tip = "投票出局";
1834                KillWolf.serverData.deathUser4=name;
```

```
1835                DataUser u =KillWolf.serverData.getUser(KillWolf.serverData.deathUser4);
1836                if(u!=null) u.live = false;
1837            }
1838
1839//            System.out.println("--------------------"+STR);
1840//            DataUser[] wu = KillWolf.serverData.getUsersWithID("狼人");
1841            //DataUser[] wu = new DataUser[users.length];
1842//            if(KillWolf.serverData.getLivingUsers(false) instanceof DataUser[])
1843//              wu = (DataUser[]) KillWolf.serverData.getLivingUsers(false);
1844            String id = "";
1845            //参与本场投票的人
1846            for (int i =0;i<users.length;i++){
1847                //wu[i]=KillWolf.serverData.getUser(users[i]);
1848                if(users[i].equals(name))
1849                {
1850                    DataUser u = KillWolf.serverData.getUser(users[i]);
1851                    if(u!=null)       id = u.identify;
1852                }
1853            }
1854            String[] s = STR.split("\n");
1855            //for(int i=0;i<wu.length;i++) {
1856                for(int j=0;j<s.length;j++) {
1857                    KillWolf.server.send("MESSAGE【投票】"+s[j]);
1858                    //KillWolf.server.sendto();
1859                }
1860                //if("".equals(id))
1861                //else KillWolf.server.sendto("MESSAGE【最终结果】"+name+"("+max+")"+tip+"，他是
    "+id,wu[i].sk);
1862            //}
1863            try {
1864                Thread.sleep(2*1000);
1865            } catch (InterruptedException e) {
1866                e.printStackTrace();
1867            }
1868            KillWolf.server.send("MESSAGE【最终结果】"+name+"("+max+")"+tip);
1869            if(!"存在平票，无人出局".equals(name))KillWolf.server.send("USERDIED"+name);
1870
1871            if(serverData.GameMainThread.isWin()) return;
1872
1873            try {
1874                Thread.sleep(2000);
1875            } catch (InterruptedException e) {
1876                e.printStackTrace();
1877            }
```

```
1878            if(id.equals("猎人"))KillWolf.serverData.GameMainThread.gameHunt(name);
1879            else{
1880                serverData.GameMainThread.gameNext();
1881            }
1882
1883    }
1884}
1885// GameMain.java
1886package KillWolf.Window;
1887
1888import KillWolf.Data.DataUser;
1889import KillWolf.KillWolf;
1890
1891import javax.swing.*;
1892import java.awt.*;
1893import java.awt.event.ActionEvent;
1894import java.awt.event.ActionListener;
1895
1896public class GameMain {
1897//      public static void main(String[] args) {
1898//          GameMain game = new GameMain();
1899//          game.reset();
1900//      }
1901      public boolean freeSpeak = false;
1902      private JFrame frame = new JFrame("开始游戏");
1903      private JPanel panel = new JPanel();
1904      //玩家列表
1905      private JLabel userListLabel = new JLabel("在场玩家");
1906      private JList<String> userList = null;
1907      //出局玩家列表
1908      private JLabel outUserListLabel = new JLabel("淘汰玩家");
1909      private JList<String> OutUserList = null;
1910      //
1911      private JLabel identity = null;
1912      //公屏
1913      private JLabel screenStr = new JLabel("公屏");
1914      private JTextArea screen = new JTextArea();
1915      private JScrollPane jsp = null;
1916//      //投票按钮
1917//      private JButton vote = new JButton("投票");
1918      //发言框
1919      private JLabel submitBoxStr = new JLabel("发言框");
1920      private JTextField    submitStr = new JTextField();
1921      //发言按钮
```

```
1922    private JButton submit = new JButton("发言");
1923    public String[] UserListData ;
1924    public String[] UserListTipData ;
1925    public String[] UserOutListData ;
1926    public DataUser userSelf=null;
1927    public void reset(){
1928        //本地玩家信息初始化
1929
1930        for(DataUser user : KillWolf.Users){
1931            if(user != null) {
1932                if (user.identify.equals("女巫")) {
1933                    user.haveAntidote = true;
1934                    user.havePoison = true;
1935                }
1936                user.live = true;
1937            }
1938        }
1939        //设置窗体参数
1940        frame.setSize(800,600);
1941        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
1942        frame.setLayout(null);
1943        frame.setResizable(false);
1944        frame.setVisible(false);
1945        frame.setLocationRelativeTo(null);
1946        //在场玩家列表设置
1947        userListLabel.setBounds(20,5,100,20);
1948        frame.add(userListLabel);
1949        String[] items1=new String[10];
1950        userList=new JList(items1);       //创建 JList
1951        userList.setBounds(10, 30, 200, 240);
1952        userList.setFixedCellHeight(20);
1953        frame.add(userList);
1954        //淘汰玩家列表设置
1955        outUserListLabel.setBounds(20,275,100,20);
1956        frame.add(outUserListLabel);
1957        String[] items2=new String[10];
1958        OutUserList=new JList(items2);       //创建 JList
1959        OutUserList.setBounds(10, 300, 200, 240);
1960        OutUserList.setFixedCellHeight(20);
1961        frame.add(OutUserList);
1962
1963
1964        //顶端字体设置
1965        DataUser[] users= KillWolf.Users;
```

```
1966            String[] usn = new String[KillWolf.UsersRealLen];
1967            String[] usna = new String[KillWolf.UsersRealLen];
1968            String[] usnt = new String[KillWolf.UsersRealLen];
1969            int n = 0;
1970            for (int i=0;i<KillWolf.UsersLen;i++) {
1971                DataUser user = users[i];
1972                if(user == null) continue;
1973                if(user.name.equals(KillWolf.UserName)){
1974                    userSelf=user;
1975                }
1976                usn[n]= user.name;
1977                //System.out.println("name:" + user.name);
1978                usna[n]=usn[n];
1979                usnt[n] = "";
1980                n++;
1981            }
1982            n=0;
1983            if(userSelf.identify.equals("狼人")){
1984                for (int i=0;i<KillWolf.UsersLen;i++) {
1985                    DataUser user = users[i];
1986                    if(user == null) continue;
1987                    if(user.identify.equals("狼人")) {
1988                        usnt[n] = "(狼人)";
1989                    }
1990                    usna[n]=usn[n]+" "+usnt[n];
1991                    n++;
1992                }
1993            }
1994            userList.setListData(usna);
1995            UserListData = usn;
1996            UserListTipData = usnt;
1997            UserOutListData = new String[0];
1998
1999            identity = new JLabel(userSelf.name +"，你的身份是：" +userSelf.identify); ;
2000            identity.setBounds(270,0,450,100);
2001            identity.setFont(new Font(Font.DIALOG, Font.BOLD, 20));
2002            frame.add(identity);
2003
2004            //公屏设置
2005            panel.setBounds(270,100,450,380);
2006            panel.setLayout(null);
2007            jsp = new JScrollPane(screen);
2008            jsp.setBounds(0,0,450,380);
2009
```

```
2010            screen.setBounds(0,0,450,380);
2011            screen.setFont(new Font(Font.DIALOG,Font.BOLD,16));
2012            screen.setEditable(false);
2013            screen.setLineWrap(true);            //激活自动换行功能
2014            screen.setWrapStyleWord(true);                  // 激活断行不断字功能
2015            screenStr.setBounds(270,60,100,50);
2016
2017            jsp.setViewportView(screen);
2018//            screen.setBackground(Color.black);
2019//            jsp.setBackground(Color.BLUE);
2020//            panel.setBackground(Color.green);
2021            //jsp.setOpaque(false);
2022            jsp.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
2023            panel.add(jsp);
2024            frame.add(panel);
2025            frame.add(screenStr);
2026
2027            submitBoxStr.setBounds(270,470,100,50);
2028            submitStr.setBounds(270,510,300,30);
2029            submitStr.setFont(new Font(Font.DIALOG,Font.BOLD,15));
2030            //submitStr.setText("请发言");
2031            submitStr.setColumns(3);
2032            frame.add(submitBoxStr);
2033            frame.add(submitStr);
2034
2035            submit.setBounds(600,510,120,30);
2036            submit.addActionListener(new ActionListener() {
2037                  public void actionPerformed(ActionEvent e) {
2038                        submit();
2039                  }
2040            });
2041            frame.add(submit);
2042
2043            this.freeSpeak = KillWolf.freeSpeak;
2044            ableSubmit();
2045
2046            frame.setVisible(true);
2047            KillWolf.client.send("GAMEREADY"+KillWolf.UserName);
2048            addToScreen("【系统】等待玩家...");
2049      }
2050      public void forbidSubmit(){
2051            if(freeSpeak)return;
2052            ClockRun = false;
2053            submit.setText("发言");
```

```
2054            this.submit.setEnabled(false);
2055        }
2056        public void ableSubmit(){
2057            //if(freeSpeak)return;
2058            this.submit.setEnabled(true);
2059        }
2060        public boolean ClockRun = false;
2061        class TimeTh extends Thread{
2062            int time;
2063            public TimeTh(int time) {
2064                this.time = time-1;
2065            }
2066            public void run() {
2067                try {
2068                    submit.setText("发言("+time+"s)");
2069                    Thread.sleep(1000);
2070                    while(time>1 && ClockRun) {
2071                        time--;
2072                        submit.setText("发言("+time+"s)");
2073                        Thread.sleep(1000);
2074
2075                    }
2076                    submit.setText("发言");
2077                    forbidSubmit();
2078                }catch(Exception e) {
2079                    e.printStackTrace();
2080                }
2081            }
2082        }
2083        public void startSpeakClock(){
2084            if(freeSpeak){
2085                TimeTh tt = new TimeTh(120);
2086                ClockRun = true;
2087                tt.start();
2088                return;
2089            }
2090            TimeTh tt = new TimeTh(30);
2091            ClockRun = true;
2092            tt.start();
2093        }
2094        public void submit(){
2095            System.out.println(submitStr.getText());
2096            if(submitStr != null && (submitStr.getText() != null && !"".equals(submitStr.getText()))){
2097                KillWolf.client.send("USERSPEAK"+userSelf.name+"\t"+submitStr.getText());
```

```
2098                submitStr.setText("");
2099                if(!KillWolf.freeSpeak){
2100                    this.submit.setEnabled(false);
2101                    ClockRun = false;
2102                    submit.setText("发言");
2103                }
2104
2105            }
2106
2107        }
2108        public void addToScreen(String str){
2109            //添加问本到公屏
2110            screen.append(str+"\r\n");
2111            //显示最新内容
2112            screen.setCaretPosition(screen.getText().length());
2113        }
2114        public    void setDark(){
2115            Color c = new Color(-12828863);
2116            if(userList != null) {
2117                userList.setBackground(c);
2118                userList.setForeground(Color.white);
2119            }
2120            if(OutUserList != null) {
2121                OutUserList.setBackground(c);
2122                OutUserList.setForeground(Color.white);
2123            }
2124            if(jsp != null){
2125                jsp.setBackground(c);
2126                jsp.setForeground(Color.white);
2127            }
2128            if(screen != null){
2129                screen.setBackground(c);
2130                screen.setForeground(Color.white);
2131            }
2132            if(submitStr != null){
2133                submit.setBackground(c);
2134                submit.setForeground(Color.white);
2135            }
2136            if(userListLabel != null)userListLabel.setForeground(Color.white);
2137            if(outUserListLabel != null)outUserListLabel.setForeground(Color.white);
2138            if(screenStr != null)screenStr.setForeground(Color.white);
2139            if(submitBoxStr != null)submitBoxStr.setForeground(Color.white);
2140            if(identity != null)identity.setForeground(Color.white);
2141            if(frame != null) frame.getContentPane().setBackground(Color.black);
```

```
2142
2143//            if(CenterLabel != null) CenterLabel.setForeground(Color.white);
2144      }
2145      public    void setWhite(){
2146            if(userList != null) {
2147                  userList.setBackground(Color.white);
2148                  userList.setForeground(Color.black);
2149            }
2150            if(OutUserList != null) {
2151                  OutUserList.setBackground(Color.white);
2152                  OutUserList.setForeground(Color.black);
2153            }
2154            if(screen != null){
2155                  screen.setBackground(Color.white);
2156                  screen.setForeground(Color.black);
2157            }
2158//            if(jsp != null){
2159//                  jsp.setBackground(Color.white);
2160//                  jsp.setForeground(Color.black);
2161//            }
2162            if(submitStr != null){
2163                  submit.setBackground(Color.white);
2164                  submit.setForeground(Color.black);
2165            }
2166            if(userListLabel != null)userListLabel.setForeground(Color.black);
2167            if(outUserListLabel != null)outUserListLabel.setForeground(Color.black);
2168            if(screenStr != null)screenStr.setForeground(Color.black);
2169            if(submitBoxStr != null)submitBoxStr.setForeground(Color.black);
2170            if(identity != null)identity.setForeground(Color.black);
2171            if(frame != null) frame.getContentPane().setBackground(new Color(-1118482));
2172
2173//            if(CenterLabel != null) CenterLabel.setForeground(Color.white);
2174      }
2175
2176      public JFrame getFrame() {
2177            return frame;
2178      }
2179
2180      public void setFrame(JFrame frame) {
2181            this.frame = frame;
2182      }
2183
2184      public JLabel getUserListLabel() {
2185            return userListLabel;
```

```
2186    }
2187
2188    public void setUserListLabel(JLabel userListLabel) {
2189        this.userListLabel = userListLabel;
2190    }
2191
2192    public JList<String> getUserList() {
2193        return userList;
2194    }
2195
2196    public void setUserList(JList<String> userList) {
2197        this.userList = userList;
2198    }
2199
2200    public JLabel getOutUserListLabel() {
2201        return outUserListLabel;
2202    }
2203
2204    public void setOutUserListLabel(JLabel outUserListLabel) {
2205        this.outUserListLabel = outUserListLabel;
2206    }
2207
2208    public JList<String> getOutUserList() {
2209        return OutUserList;
2210    }
2211
2212    public void setOutUserList(JList<String> outUserList) {
2213        OutUserList = outUserList;
2214    }
2215
2216    public JLabel getIdentity() {
2217        return identity;
2218    }
2219
2220    public void setIdentity(JLabel identity) {
2221        this.identity = identity;
2222    }
2223
2224    public JLabel getScreenStr() {
2225        return screenStr;
2226    }
2227
2228    public void setScreenStr(JLabel screenStr) {
2229        this.screenStr = screenStr;
```

```
2230    }
2231
2232    public JTextArea getScreen() {
2233        return screen;
2234    }
2235
2236    public void setScreen(JTextArea screen) {
2237        this.screen = screen;
2238    }
2239
2240    public JLabel getSubmitBoxStr() {
2241        return submitBoxStr;
2242    }
2243
2244    public void setSubmitBoxStr(JLabel submitBoxStr) {
2245        this.submitBoxStr = submitBoxStr;
2246    }
2247
2248    public JTextField getSubmitStr() {
2249        return submitStr;
2250    }
2251
2252    public void setSubmitStr(JTextField submitStr) {
2253        this.submitStr = submitStr;
2254    }
2255
2256    public JButton getSubmit() {
2257        return submit;
2258    }
2259
2260    public void setSubmit(JButton submit) {
2261        this.submit = submit;
2262    }
2263
2264    public String[] getUserListData() {
2265        return UserListData;
2266    }
2267
2268    public void setUserListData(String[] userListData) {
2269        UserListData = userListData;
2270    }
2271
2272    public String[] getUserOutListData() {
2273        return UserOutListData;
```

```
2274        }
2275
2276        public void setUserOutListData(String[] userOutListData) {
2277            UserOutListData = userOutListData;
2278        }
2279}
2280// LoginWindow.java
2281package KillWolf.Window;
2282
2283import java.awt.event.ActionEvent;
2284import java.awt.event.ActionListener;
2285import java.util.Random;
2286
2287import javax.swing.JButton;
2288import javax.swing.JFrame;
2289import javax.swing.JLabel;
2290import javax.swing.JOptionPane;
2291import javax.swing.JTextField;
2292
2293import KillWolf.KillWolf;
2294import KillWolf.SocketDeal.ClientDeal;
2295import KillWolf.SocketDeal.ServerDeal;
2296import SocketServe.sClient;
2297import SocketServe.sServer;
2298
2299public class LoginWindow {
2300        public JButton ServerButton = null;
2301        public JButton LoginButton = null;
2302        public JTextField userName = null;
2303        public JTextField ServerIp = null;
2304        public JFrame window = null;
2305        public String MIP = null;
2306        public void reset(String ip) {
2307            window = new JFrame("狼人杀  V.0.1");
2308            window.setVisible(false);
2309            window.setSize(600, 200);//设置大小
2310            window.setLocationRelativeTo(null);//设置居中
2311            window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//设置可关闭
2312            window.setLayout(null);//设置绝对布局（窗口里面的内容不会随着窗口的改变而改变）
2313            window.setResizable(false);//设置窗口不可拉伸改变大小
2314            //设置用户名标签
2315            JLabel username_label =new JLabel("用户名");
2316            username_label.setBounds(100,30,100,20);
2317            window.add(username_label);
```

```
2318            //设置文本框
2319            userName = new JTextField();
2320            userName.setBounds(160, 30, 300, 20);
2321            Random r = new Random();
2322            //r.nextInt();
2323            r.setSeed(System.currentTimeMillis());
2324            userName.setText("user"+r.nextInt(10000));
2325            window.add(userName);
2326            //设置标签
2327            JLabel ServerIp_label =new JLabel("服务器地址");
2328            ServerIp_label.setBounds(80,80,100,20);
2329            window.add(ServerIp_label);
2330            //设置文本框
2331            ServerIp = new JTextField();
2332            ServerIp.setBounds(160, 80, 300, 20);
2333            ServerIp.setText(ip);
2334            this.MIP = ip;
2335            window.add(ServerIp);
2336            //JPasswordField pwd=new JPasswordField();//隐藏密码
2337            //pwd.setBounds(150, 200, 300, 50);
2338            //window.add(pwd);
2339            //设置按钮
2340            LoginButton = new JButton("连接服务器");
2341            //Login.setEnabled(false);
2342            LoginButton.setBounds(180, 120, 100, 20);
2343            window.add(LoginButton);
2344            ServerButton = new JButton("作为服务器");
2345            //Login.setEnabled(false);
2346            ServerButton.setBounds(320, 120, 100, 20);
2347            window.add(ServerButton);
2348            //设置作为主机动作
2349            ButtonAction ServerAction = new ButtonAction();
2350            ServerButton.addActionListener(ServerAction);
2351            //设置连接服务器动作
2352            LinkAction LinkAction = new LinkAction();
2353            LoginButton.addActionListener(LinkAction);
2354            window.setVisible(true);//设置面板可见
2355        }
2356        class ButtonAction implements ActionListener{
2357            public void actionPerformed(ActionEvent event){
2358                if(!setName()) return;
2359                ServerIp.setText(MIP);
2360                KillWolf.server = new sServer();
2361                ServerDeal cb = new ServerDeal();
```

```
2362                if(!KillWolf.server.start(cb)){
2363                    JOptionPane.showMessageDialog(null, "端口被占用，服务器启动失败！");
2364                    return;
2365                }
2366                if(!createClient(MIP)){
2367                    JOptionPane.showMessageDialog(null, "客户端启动失败！");
2368                    return;
2369                }
2370                //window.setTitle(ServerIp.getText());
2371                KillWolf.ServerIp = MIP;
2372                KillWolf.isServer = true;
2373
2374                ServerIp.setEnabled(false);
2375                ServerButton.setEnabled(false);
2376                KillWolf.client.send("ENTER"+KillWolf.UserName);
2377
2378                //window.setTitle(ServerIp.getText());
2379            }
2380        }
2381        class LinkAction implements ActionListener{
2382            public void actionPerformed(ActionEvent event){
2383                if(!setName()) return;
2384                if(KillWolf.ServerIp == null) {
2385                    if(!createClient(ServerIp.getText())){
2386                        JOptionPane.showMessageDialog(null, "客户端启动失败!可能是服务器未连接
    成功。");
2387                        return;
2388                    }
2389                    KillWolf.ServerIp = ServerIp.getText();
2390                    KillWolf.isServer = false;
2391                }
2392
2393                ServerIp.setEnabled(false);
2394                ServerButton.setEnabled(false);
2395                KillWolf.client.send("ENTER"+KillWolf.UserName);
2396            }
2397        }
2398        public boolean createClient(String IP) {
2399            KillWolf.client = new sClient();
2400            ClientDeal cb = new ClientDeal();
2401            //System.out.println("Connecting Server on "+IP+"...");
2402            return KillWolf.client.start(IP,cb);
2403        }
2404        public boolean setName() {
```

```
2405            if( userName.getText().equals("")) {
2406                JOptionPane.showMessageDialog(null, "请输入用户名");
2407                return false;
2408            }
2409            KillWolf.UserName = userName.getText();
2410            return true;
2411
2412        }
2413}
2414// SelectWindow.java
2415package KillWolf.Window;
2416
2417import java.awt.Color;
2418import java.awt.event.ActionEvent;
2419import java.awt.event.ActionListener;
2420
2421import javax.swing.JButton;
2422import javax.swing.JFrame;
2423import javax.swing.JLabel;
2424import javax.swing.JList;
2425import javax.swing.JRootPane;
2426import javax.swing.ListSelectionModel;
2427import javax.swing.event.ListSelectionEvent;
2428import javax.swing.event.ListSelectionListener;
2429
2430import KillWolf.KillWolf;
2431import KillWolf.Data.SelectCallBack;
2432
2433
2434/*
2435      构造类：String name,boolean ReceiveFromServer,String ReceiveID,boolean DisableParent,JFrame
      ParentWindow;
2436      方法：设置标题（可用于显示剩余时间） setTitle(title)
2437            启动 start(String tip,String ButtonTitle,String[] users, boolean canClose,SelectCallBack cb)
2438            关闭 getSelectionAndClose()
2439      接口：SelectCallBack 实现方法： run(String Selection) 当选择完毕后会回调，如果 Selection 为
      **NONE** 表示没有选择/弃权
2440
2441      for example:
2442            SelectWindow = new SelectWindow("选择",true,"123",true,loginwindow.window);
2443            String[] b = {"a","b","c",UserName};
2444            SelectWindow.start("请选择", "投票", b, true,cb);
2445
2446            cb 另需构造
```

```
2447*/
2448
2449
2450public class SelectWindow {
2451    public boolean isRunCB = false;
2452    public String ReceiveID;
2453    public boolean ReceiveFromServer;
2454    public boolean DisableParent;
2455    public JFrame ParentWindow;
2456    public JFrame window;
2457    public JLabel CenterLabel;
2458    public JList<String> List;
2459    public String name;
2460    public JButton ConfirmButton;
2461    public JButton CloseButton;
2462    public String[] originUsers;
2463    public String[] usersTip;
2464    public String[] targetUsers;
2465    public int[] selectCount;
2466    public String LastSelect = "**NONE**";
2467    public boolean isSendFinal = false;
2468    public SelectWindow(String name,boolean ReceiveFromServer,String ReceiveID,boolean
    DisableParent,JFrame ParentWindow) {
2469        this.ReceiveFromServer = ReceiveFromServer;
2470        this.ReceiveID = ReceiveID;
2471        this.DisableParent = DisableParent;
2472        this.ParentWindow = ParentWindow;
2473        this.name = name;
2474    }
2475    public boolean DarkMode = false;
2476    public SelectCallBack cb;
2477    public void setColorMode(boolean dark) {
2478        this.DarkMode = dark;
2479        if(dark) {
2480            if(List != null) {
2481                List.setBackground(Color.black);
2482                List.setForeground(Color.white);
2483            }
2484            if(window != null) window.getContentPane().setBackground(Color.black);
2485            if(CenterLabel != null) CenterLabel.setForeground(Color.white);
2486            //if()
2487        }
2488    }
2489    public void start(String tip,String ButtonTitle,String[] users, boolean canClose,SelectCallBack cb, int
```

```
    TimeLimit, String[] usertip) {
2490            this.cb = cb;
2491            if(usertip==null) {
2492                usertip = new String[users.length];
2493            }
2494            this.usersTip = usertip;
2495            window = new JFrame(name);
2496            window.setVisible(false);
2497            window.setSize(300, 400);//设置大小
2498            window.setLocationRelativeTo(null);//设置居中
2499            window.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);//设置可关闭
2500            window.setLayout(null);//设置绝对布局（窗口里面的内容不会随着窗口的改变而改变）
2501            window.setResizable(false);//设置窗口不可拉伸改变大小
2502            window.setAlwaysOnTop(true);
2503            window.setUndecorated(true);                                    //不显示标
    题栏,最大化,最小化,退出按钮
2504            window.getRootPane().setWindowDecorationStyle(JRootPane.WARNING_DIALOG);//      使
    frame 只剩下标题栏
2505            CenterLabel =new JLabel(tip);
2506            CenterLabel.setBounds(10,10,260,20);
2507            window.add(CenterLabel);
2508            users = users.clone();
2509            originUsers = new String[users.length];
2510            targetUsers = new String[users.length];
2511            for(int i=0;i<users.length;i++) {
2512                originUsers[i] = users[i];
2513                targetUsers[i] = "";
2514                if(usersTip[i]==null) usersTip[i]="";
2515                if(ReceiveFromServer) users[i] = "(0)"+users[i]+" "+usersTip[i];
2516                else users[i] = users[i]+" "+usersTip[i];
2517
2518            }
2519            selectCount = new int[users.length];
2520            List = new JList(users);       //创建 JList
2521            List.setBounds(10, 40, 260, 260);
2522            List.setFixedCellHeight(20);
2523            List.addListSelectionListener(new ListSelectionHandler());
2524            List.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
2525            window.add(List);
2526
2527            ConfirmButton = new JButton(ButtonTitle);
2528            ConfirmButton.setEnabled(false);
2529            ConfirmButton.setBounds(30, 310, 100, 40);
2530            ConfirmButton.addActionListener(new ButtonAction("confirm"));
```

```
2531            window.add(ConfirmButton);
2532
2533
2534            CloseButton = new JButton("弃权");
2535            if(canClose) {
2536                CloseButton.setEnabled(true);
2537            }else {
2538                CloseButton.setEnabled(false);
2539            }
2540            CloseButton.setBounds(160, 310, 100, 40);
2541            CloseButton.addActionListener(new ButtonAction("cancel"));
2542            window.add(CloseButton);
2543
2544
2545            if(this.DisableParent && this.ParentWindow != null) this.ParentWindow.setEnabled(false);
2546
2547            TimeTh at = new TimeTh(TimeLimit);
2548            at.start();
2549
2550            setColorMode(this.DarkMode);
2551
2552            window.setVisible(true);
2553        }
2554        class TimeTh extends Thread{
2555            int time;
2556            public TimeTh(int time) {
2557                this.time = time-1;
2558            }
2559            public void run() {
2560                try {
2561                    window.setTitle(time+"s "+name);
2562                    while(time>0) {
2563                        Thread.sleep(1000);
2564                        time--;
2565                        window.setTitle(time+"s "+name);
2566                    }
2567                    window.setTitle(name);
2568                    ConfirmButton.setEnabled(false);
2569                    CloseButton.setEnabled(false);
2570                }catch(Exception e) {
2571                    e.printStackTrace();
2572                }
2573            }
2574        }
```

```
2575    public void setTitle(String Title) {
2576        window.setTitle(name + Title);
2577    }
2578    public void SELECTInfo(String ServerText) {
2579        if(!ReceiveFromServer) return;
2580        String[] c = ServerText.split("\t");
2581        // c[0] ReceiveID ,c[1] 发出者 ,c[2] 目标者, c[3] 原始目标（被取消）
2582        String[] nu = new String[this.originUsers.length];
2583
2584        if(c[0].equals(this.ReceiveID)) {
2585
2586            for(int i=0;i<this.originUsers.length;i++) {
2587                //nu[i] = this.originUsers[i];
2588
2589                if(this.originUsers[i].equals(c[1])) {
2590                    if(c[2].equals("**NONE**")) targetUsers[i] = "  弃权";
2591                    else targetUsers[i] = " -> " + c[2];
2592                }
2593                if(this.originUsers[i].equals(c[2])) {
2594                    selectCount[i]++;
2595                    System.out.println(this.originUsers[i]+" +1 => "+ selectCount[i]);
2596                }
2597                if(this.originUsers[i].equals(c[3])) {
2598                    selectCount[i]--;
2599                    System.out.println(this.originUsers[i]+" -1 => "+ selectCount[i]);
2600                }
2601                System.out.println(this.originUsers[i]+" "+c[2]+" "+ selectCount[i]);
2602                //System.out.println("LIST0: "+nu[i]);
2603                //System.out.println("LISTc: "+selectCount[i]);
2604            }
2605            for(int i=0;i<nu.length;i++) {
2606                nu[i] = "(" +selectCount[i] + ")" +originUsers[i]+" "+usersTip[i]+ targetUsers[i];
2607                //System.out.println("LIST: "+nu[i]);
2608            }
2609            List.setListData(nu);
2610        }
2611
2612    }
2613    class ListSelectionHandler implements ListSelectionListener {
2614        public void valueChanged(ListSelectionEvent e) {
2615            if(e.getValueIsAdjusting()) {
2616                String t;
2617                if(List.getSelectedIndex() == -1) {
2618                    //t = "**NONE**";
```

```
2619                        return;
2620                        }
2621                    else t = originUsers[List.getSelectedIndex()];
2622                    System.out.printf("Select: %s\n",t);
2623                    ConfirmButton.setEnabled(true);
2624
2625                    String a = LastSelect;
2626                    if(List.getSelectedIndex() == -1)
2627                        LastSelect = "**NONE**";
2628                    else LastSelect = originUsers[List.getSelectedIndex()];
2629
2630                    if(ReceiveFromServer)
    KillWolf.client.send("SELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+t+"\t"+a);
2631                }
2632
2633                //output.append("LeadSelectionIndex is " + lsm.getLeadSelectionIndex() + "\n");
2634
2635
2636            }
2637        }
2638        class ButtonAction implements ActionListener{//开始游戏
2639            String ID;
2640            public ButtonAction(String ID) {
2641                this.ID = ID;
2642            }
2643            public void actionPerformed(ActionEvent event){
2644                isSendFinal = true;
2645                List.setEnabled(false);
2646                CloseButton.setEnabled(false);
2647                ConfirmButton.setEnabled(false);
2648                isRunCB = true;
2649                if(ID.equals("confirm")) {
2650                    //List.setEnabled(false);
2651                    if(ReceiveFromServer) {
2652
    KillWolf.client.send("FINALSELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+LastSelect);
2653                    }
2654                    if(cb != null) cb.run(LastSelect);
2655                }else {
2656                    //取消
2657                    if(ReceiveFromServer) {
2658
    KillWolf.client.send("SELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+"**NONE**"+"\t"+LastSelec
    t);
```

```
2659
    KillWolf.client.send("FINALSELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+"**NONE**");
2660                        }
2661                        if(cb != null) cb.run("**NONE**");
2662                    }
2663
2664
2665
2666            }
2667        }
2668        public String getSelectionAndClose(String ReceiveID) {
2669            if(!ReceiveID.equals(this.ReceiveID)) return "";
2670            boolean a = isRunCB;
2671            isRunCB = true;
2672            window.setVisible(false);
2673            window.setEnabled(false);
2674            if(!a && cb != null) {
2675                cb.run(LastSelect);
2676            }
2677            if(!isSendFinal && ReceiveFromServer){
2678                isSendFinal = true;
2679
    KillWolf.client.send("FINALSELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+LastSelect);
2680            }
2681            if(DisableParent && ParentWindow != null) ParentWindow.setEnabled(true);
2682
2683            return LastSelect;
2684        }
2685}
2686// WaitRoom.java
2687package KillWolf.Window;
2688
2689import javax.swing.*;
2690import javax.swing.border.Border;
2691import javax.swing.event.*;
2692import javax.swing.text.BadLocationException;
2693import javax.swing.text.Document;
2694
2695import KillWolf.KillWolf;
2696import KillWolf.Data.DataUser;
2697import KillWolf.SocketDeal.ClientDeal;
2698
2699import java.awt.*;
2700import java.awt.event.*;
```

```
2701import java.util.Random;
2702
2703import SocketServe.*;
2704
2705public class WaitRoom {
2706	public JButton LoginButton = null;
2707	public JTextField peoCount = null;
2708	public JTextField wolfCount = null;
2709	public JCheckBox wizardOption;
2710	public JCheckBox hunterOption;
2711	public JCheckBox yuyanjiaOption;
2712	public JCheckBox winmodeOption;
2713	public JCheckBox speakOption;
2714	//public JCheckBox optionD;
2715	public JList UserList;
2716	public JFrame window = null;
2717	public JLabel Userlenlabel;
2718	public JLabel Charlenlabel;
2719	public String MIP = null;
2720	public int CharLen = 0;
2721	public int peoLen = 1;
2722	public int wolfLen = 1;
2723	public int wizardLen = 0;
2724	public int hunterLen = 0;
2725	public int yuyanjiaLen = 0;
2726	public void reset(String ip,String name,boolean isServer) {
2727		KillWolf.serverData.freeSpeak = false;
2728		KillWolf.serverData.playMode = false;
2729
2730		window = new JFrame("狼人杀  "+name+" "+ip);
2731		window.setVisible(false);
2732		window.setSize(400, 420);//设置大小
2733		window.setLocationRelativeTo(null);//设置居中
2734		window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//设置可关闭
2735		window.setLayout(null);//设置绝对布局（窗口里面的内容不会随着窗口的改变而改变）
2736		window.setResizable(false);//设置窗口不可拉伸改变大小
2737		//设置用户名标签
2738		JLabel username_label =new JLabel("用户名："+name);
2739		username_label.setBounds(10,10,200,20);
2740		window.add(username_label);
2741		//设置标签
2742		JLabel ServerIp_label =new JLabel("服务器地址："+KillWolf.ServerIp);
2743		ServerIp_label.setBounds(10,40,200,20);
2744		window.add(ServerIp_label);
```

```
2745            //设置标签
2746            Userlenlabel =new JLabel("在线玩家数："+KillWolf.UsersRealLen);
2747            Userlenlabel.setBounds(10,150,200,20);
2748            window.add(Userlenlabel);
2749            JLabel label2 =new JLabel("等待开始...");
2750            label2.setBounds(10,70,200,20);
2751            window.add(label2);
2752            this.MIP = ip;
2753            //设置按钮
2754
2755            if(isServer) {
2756                LoginButton = new JButton("开始游戏");
2757                //LoginButton.setEnabled(false);
2758            }else {
2759                LoginButton = new JButton("等待房主开始");
2760                LoginButton.setEnabled(false);
2761            }
2762            LoginButton.setBounds(10, 120, 100, 20);
2763            window.add(LoginButton);
2764
2765            LinkAction LinkAction = new LinkAction();
2766            LoginButton.addActionListener(LinkAction);
2767
2768
2769            String[] items=new String[10];
2770            UserList=new JList(items);       //创建 JList
2771            UserList.setBounds(200, 10, 150, 300);
2772            UserList.setFixedCellHeight(20);
2773            window.add(UserList);
2774
2775            if(KillWolf.isServer) {
2776                //设置文本框
2777                Charlenlabel =new JLabel("已选角色数："+CharLen);
2778                Charlenlabel.setBounds(10,170,200,20);
2779                window.add(Charlenlabel);
2780
2781                JLabel Peolenlabel =new JLabel("平民");
2782                Peolenlabel.setBounds(10,200,40,20);
2783                window.add(Peolenlabel);
2784                peoCount = new JTextField();
2785                peoCount.setBounds(50,200,40,20);
2786                peoCount.setText(Integer.toString(peoLen));
2787                   window.add(peoCount);
2788
```

```
2789                JLabel Wolflenlabel =new JLabel("狼人");
2790                Wolflenlabel.setBounds(10,220,40,20);
2791            window.add(Wolflenlabel);
2792            wolfCount = new JTextField();
2793            wolfCount.setBounds(50,220,40,20);
2794            wolfCount.setText(Integer.toString(wolfLen));
2795                window.add(wolfCount);
2796
2797
2798                Document dt = wolfCount.getDocument();
2799            dt.addDocumentListener(new TextListener("wolf"));
2800            wolfCount.addKeyListener(new CountKeyListener());
2801
2802
2803            Document dt2 = peoCount.getDocument();
2804            dt2.addDocumentListener(new TextListener("people"));
2805            peoCount.addKeyListener(new CountKeyListener());
2806
2807                //TextOnValueChanged WAL = new TextOnValueChanged();
2808                //wolfCount.addActionListener(WAL);
2809                //peoCount.addActionListener(WAL);
2810
2811                wizardOption = new JCheckBox("女巫");
2812                wizardOption.setBounds(10,240,80,20);
2813            hunterOption = new JCheckBox("猎人");
2814            hunterOption.setBounds(10,260,80,20);
2815            yuyanjiaOption = new JCheckBox("预言家");
2816            yuyanjiaOption.setBounds(10,280,100,20);
2817            //optionD = new JCheckBox("D.");
2818
2819
2820            wizardOption.addItemListener(new CheckValueChanged("wizard"));
2821            hunterOption.addItemListener(new CheckValueChanged("hunter"));
2822            yuyanjiaOption.addItemListener(new CheckValueChanged("yuyanjia"));
2823
2824            window.add(wizardOption);
2825            window.add(hunterOption);
2826            window.add(yuyanjiaOption);
2827
2828
2829            winmodeOption = new JCheckBox("屠城玩法（狼人需要全部淘汰民和神）");
2830                winmodeOption.setBounds(10,320,300,20);
2831            speakOption = new JCheckBox("允许随时发言");
2832            speakOption.setBounds(10,340,100,20);
```

```
2833
2834                winmodeOption.addItemListener(new CheckValueChanged("获胜玩法"));
2835                speakOption.addItemListener(new CheckValueChanged("发言"));
2836
2837                window.add(winmodeOption);
2838                window.add(speakOption);
2839
2840                updateCharCount(false);
2841            }
2842            window.setVisible(true);//设置面板可见
2843
2844
2845            //run(new ListTest(),250,375);
2846
2847        }
2848        public void setUserCount(String users[]) {
2849            Userlenlabel.setText("在线玩家数："+KillWolf.UsersRealLen);
2850            UserList.setListData(users);
2851            UserList.setFixedCellHeight(20);
2852
2853            updateCharCount(false);
2854            //window.setTitle(Title);
2855        }
2856        class LinkAction implements ActionListener{//开始游戏
2857            public void actionPerformed(ActionEvent event){
2858                if(updateCharCount(true)) {
2859                    LoginButton.setEnabled(false);
2860                    Random r = new Random();
2861                    //r.nextInt();
2862                    r.setSeed(System.currentTimeMillis());
2863                    int n;
2864                    if(wizardOption.isSelected()) {
2865                        do{n = r.nextInt(KillWolf.serverData.UsersLen);}
2866                        while(KillWolf.serverData.ServerUsers[n] == null ||
    KillWolf.serverData.ServerUsers[n].identify != null);
2867                        KillWolf.serverData.ServerUsers[n].identify = "女巫";
2868                    }
2869                    if(hunterOption.isSelected()) {
2870                        do{n = r.nextInt(KillWolf.serverData.UsersLen);}
2871                        while(KillWolf.serverData.ServerUsers[n] == null ||
    KillWolf.serverData.ServerUsers[n].identify != null);
2872                        KillWolf.serverData.ServerUsers[n].identify = "猎人";
2873                    }
2874                    if(yuyanjiaOption.isSelected()) {
```

```
2875                        do{n = r.nextInt(KillWolf.serverData.UsersLen);}
2876                        while(KillWolf.serverData.ServerUsers[n] == null ||
    KillWolf.serverData.ServerUsers[n].identify != null);
2877                        KillWolf.serverData.ServerUsers[n].identify = "预言家";
2878                    }
2879                    for(int i=0;i<wolfLen;i++){
2880                        do{n = r.nextInt(KillWolf.serverData.UsersLen);}
2881                        while(KillWolf.serverData.ServerUsers[n] == null ||
    KillWolf.serverData.ServerUsers[n].identify != null);
2882                        KillWolf.serverData.ServerUsers[n].identify = "狼人";
2883                    }
2884
2885                    for(int i=0;i<KillWolf.serverData.UsersLen;i++){
2886                        if(KillWolf.serverData.ServerUsers[i] != null &&
    KillWolf.serverData.ServerUsers[i].identify == null) {
2887                            KillWolf.serverData.ServerUsers[i].identify = "平民";
2888                        }
2889                    }
2890
2891
2892                    String STR = "";
2893                    for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
2894                        DataUser p = KillWolf.serverData.ServerUsers[i];
2895                        if(p!=null)STR = STR + "#u#" + p.name+"\t"+p.identify;
2896                    }
2897                    System.out.print(STR);
2898
    KillWolf.server.send("GAMESTART"+(KillWolf.serverData.freeSpeak?"TRUE":"FALSE")+STR);
2899                }
2900            }
2901        }
2902
2903        class CheckValueChanged implements ItemListener{
2904            String ID;
2905            public CheckValueChanged(String ID) {
2906                this.ID = ID;
2907            }
2908
2909            public void itemStateChanged(ItemEvent e) {
2910                JCheckBox jcb = (JCheckBox) e.getItem();// 将得到的事件强制转化为 JCheckBox 类
2911                if (jcb.isSelected()) {// 推断是否被选择
2912                    if(this.ID.equals("wizard")) wizardLen = 1;
2913                    else if(this.ID.equals("hunter")) hunterLen = 1;
2914                    else if(this.ID.equals("yuyanjia")) yuyanjiaLen = 1;
```

```
2915                    else if(this.ID.equals("发言")) KillWolf.serverData.freeSpeak = true;
2916                    else if(this.ID.equals("获胜玩法")) KillWolf.serverData.playMode = true;
2917                } else {
2918                    if(this.ID.equals("wizard")) wizardLen = 0;
2919                    else if(this.ID.equals("hunter")) hunterLen = 0;
2920                    else if(this.ID.equals("yuyanjia")) yuyanjiaLen = 0;
2921                    else if(this.ID.equals("发言")) KillWolf.serverData.freeSpeak = false;
2922                    else if(this.ID.equals("获胜玩法")) KillWolf.serverData.playMode = false;
2923                }
2924                updateCharCount(false);
2925            }
2926        }
2927        class TextListener implements DocumentListener{
2928            String ID;
2929            public TextListener(String ID) {
2930                this.ID = ID;
2931            }
2932            public void insertUpdate(DocumentEvent e) {
2933                //System.out.println("insertUpdate" + e.toString());
2934                changedUpdate(e);
2935            }
2936            public void removeUpdate(DocumentEvent e) {
2937                //System.out.println("removeUpdate"+e.toString());
2938                changedUpdate(e);
2939            }
2940
2941            public void changedUpdate(DocumentEvent e) {
2942
2943                try {
2944                    int num = Integer.parseInt(e.getDocument().getText(0, e.getDocument().getLength()));
2945                    if(ID.equals("people")) {
2946                        peoLen = Math.abs(num);
2947                    }else {
2948                        wolfLen = Math.abs(num);
2949                    }
2950                    updateCharCount(false);
2951
2952                }catch (BadLocationException e1) {
2953                    // TODO Auto-generated catch block
2954                    e1.printStackTrace();
2955                }catch(Exception e21) {
2956                    updateCharCount(false);
2957                }
2958
```

```
2959
2960            }
2961        }
2962        public class CountKeyListener implements KeyListener {
2963            @Override
2964            public void keyTyped(KeyEvent e) {
2965            // TODO Auto-generated method stub
2966            int keyChar=e.getKeyChar();
2967            if (keyChar>=KeyEvent.VK_0 && keyChar<=KeyEvent.VK_9) {
2968            } else {
2969            e.consume();
2970            }
2971            }
2972            @Override
2973            public void keyPressed(KeyEvent e) {
2974            // TODO Auto-generated method stub
2975            }
2976            @Override
2977            public void keyReleased(KeyEvent e) {
2978            // TODO Auto-generated method stub
2979            }
2980            }
2981        boolean updateCharCount(boolean isFinal) {
2982            CharLen = wolfLen+peoLen+wizardLen+hunterLen+yuyanjiaLen;
2983            if(Charlenlabel!=null) Charlenlabel.setText("已选角色数："+CharLen);
2984            if(!KillWolf.isServer) return true;
2985            if(wolfLen<1 && isFinal) {
2986                JOptionPane.showMessageDialog(null, "狼人角色数量至少为 1");
2987                return false;
2988            }
2989
2990            if(CharLen == KillWolf.serverData.UsersRealLen) {
2991
2992                LoginButton.setEnabled(true);
2993                LoginButton.setText("开始游戏");
2994                return true;
2995            }else {
2996                LoginButton.setEnabled(false);
2997                LoginButton.setText("角色不匹配");
2998                return false;
2999            }
3000        }
3001        //定义新类，实现 Exit 按钮的时间监听
3002        /*class btnAction implements ActionListener
```

```
3003      { //接收事件
3004            public void actionPerformed(ActionEvent event)
3005            {
3006                  Object object = event.getSource();
3007                  if (object == JButton1)
3008                        JButton1_actionPerformed(event);
3009            }
3010      } */
3011}
```

分析总结、收获和体会:

优点：

图形化页面、对所有用户的退出存在监测并实时应用到游戏中、联网游戏。

创新之处：

网络联机

不足之处：

图形页面没有图片、图标等

需要改进的地方：

图形页面没有图片、图标等

自查自纠：	是	否
程序是否有尚未解决的问题或 bug？		否
程序代码是否符合代码规范(对齐与缩进，有必要的注释)？	是	
是否按模块化要求进行了程序设计，系统功能是否完善？	**是**	
是否独立完成，未参考其他人的设计或代码？	是	

报告完成日期：2021-07-30