《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 王炳轩 – 编程、写报告** **120L022109 李世轩 – 编程、写报告** **120L022116 袁琪博 – 写报告** **120L022130 徐哲涵 – 写报告**	
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

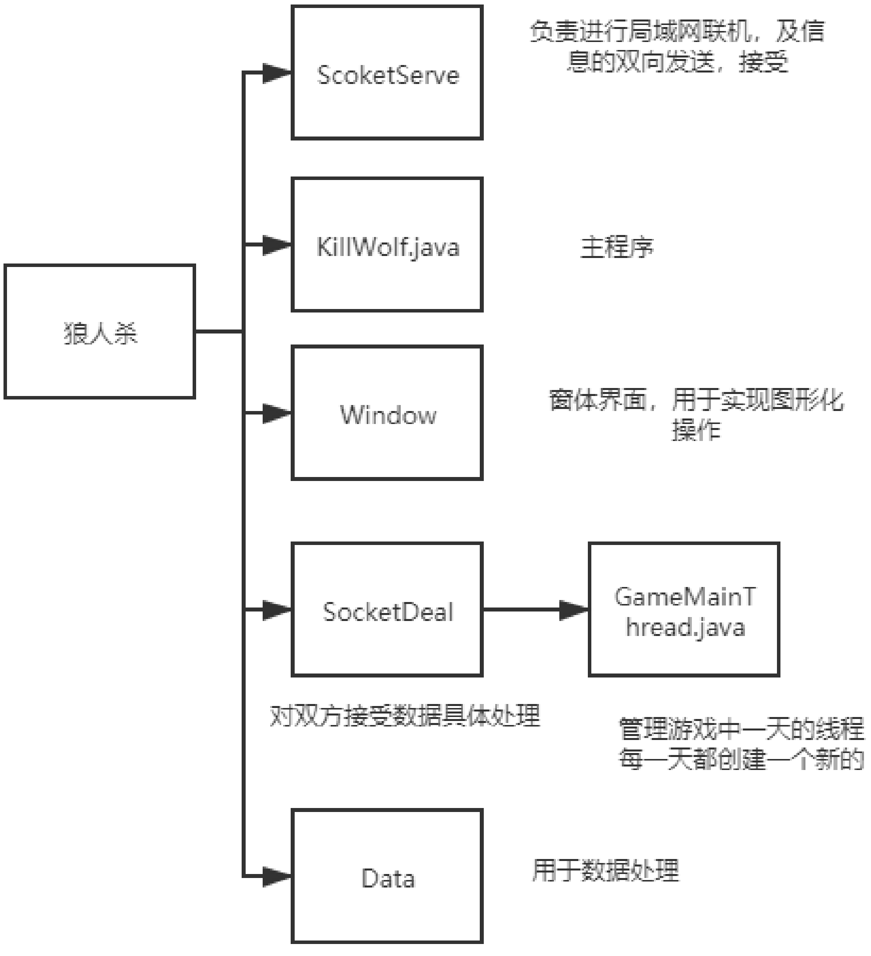
均为 String 型

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：


狼人杀
ScoketServe
负责进行局域网联机，及信息的双向发送，接受
KillWolf.java
主程序
Window
窗体界面，用于实现图形化操作
SocketDeal
对双方接受数据具体处理
GameMainThread.java
管理游戏中一天的线程 每一天都创建一个新的
Data
用于数据处理

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	ReceiveID	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		

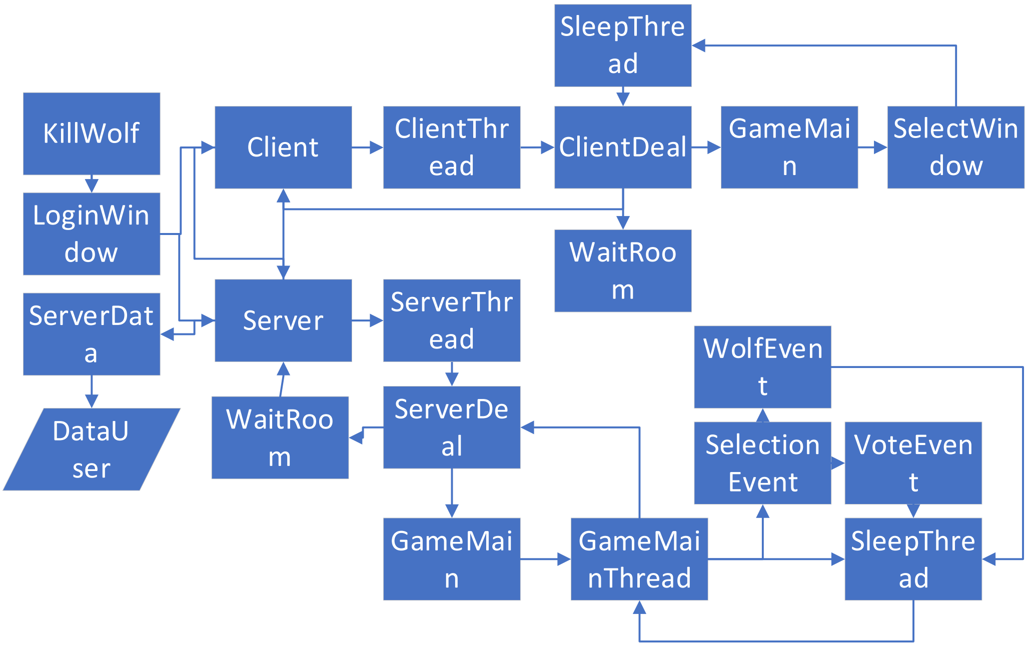
序号	成员函数	函数功能	函数参数	函数返回值
1	run			void

（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：


SleepThread
KillWolf
Client
ClientThread
ClientDeal
GameMain
SelectWindow
LoginWindow
WaitRoom
ServerData
Server
ServerThread
WolfEvent
DataUser
WaitRoom
ServerDeal
SelectionEvent
VoteEvent
GameMain
GameMainThread
SleepThread

3. 数据结构

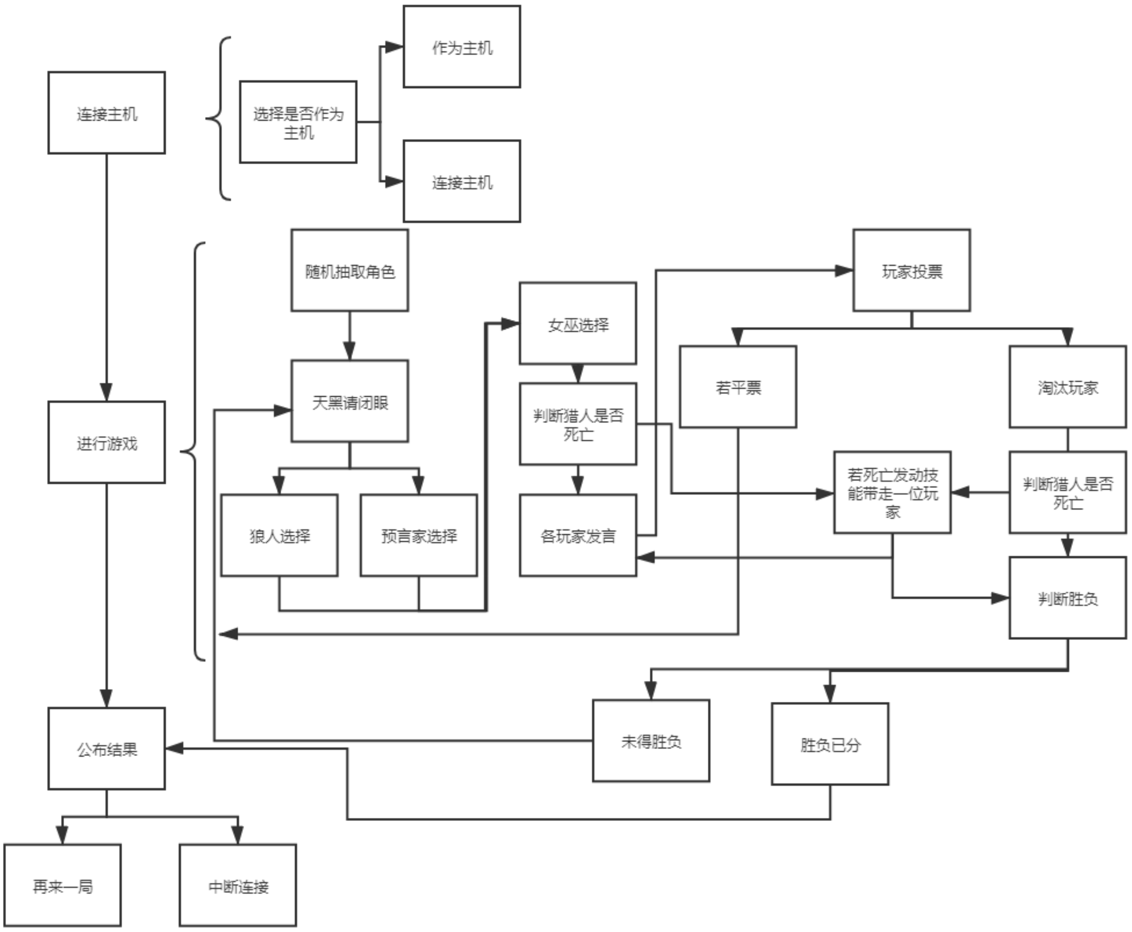
整型数组、对象数组、容器类

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：


连接主机
选择是否作为主机
作为主机
连接主机
随机抽取角色
天黑请闭眼
狼人选择
预言家选择
女巫选择
判断猎人是否死亡
各玩家发言
玩家投票
若平票
淘汰玩家
若死亡发动技能带走一位玩家
判断猎人是否死亡
判断胜负
进行游戏
未得胜负
胜负已分
公布结果
再来一局
中断连接

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。

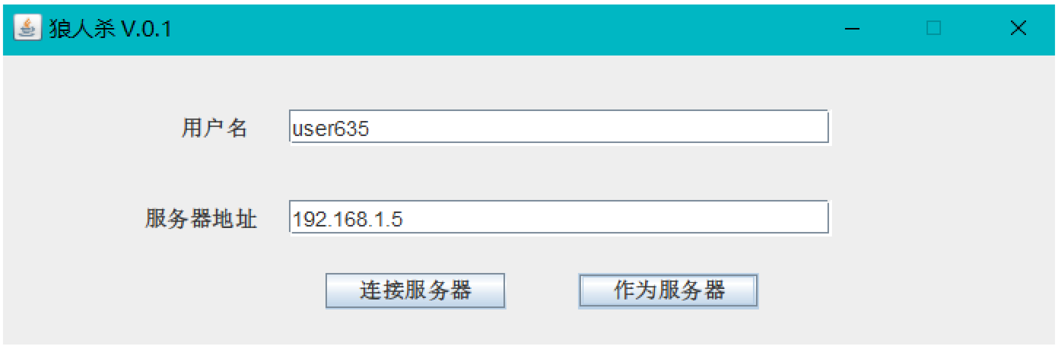
原因：对 swing 中组件使用问题。

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：

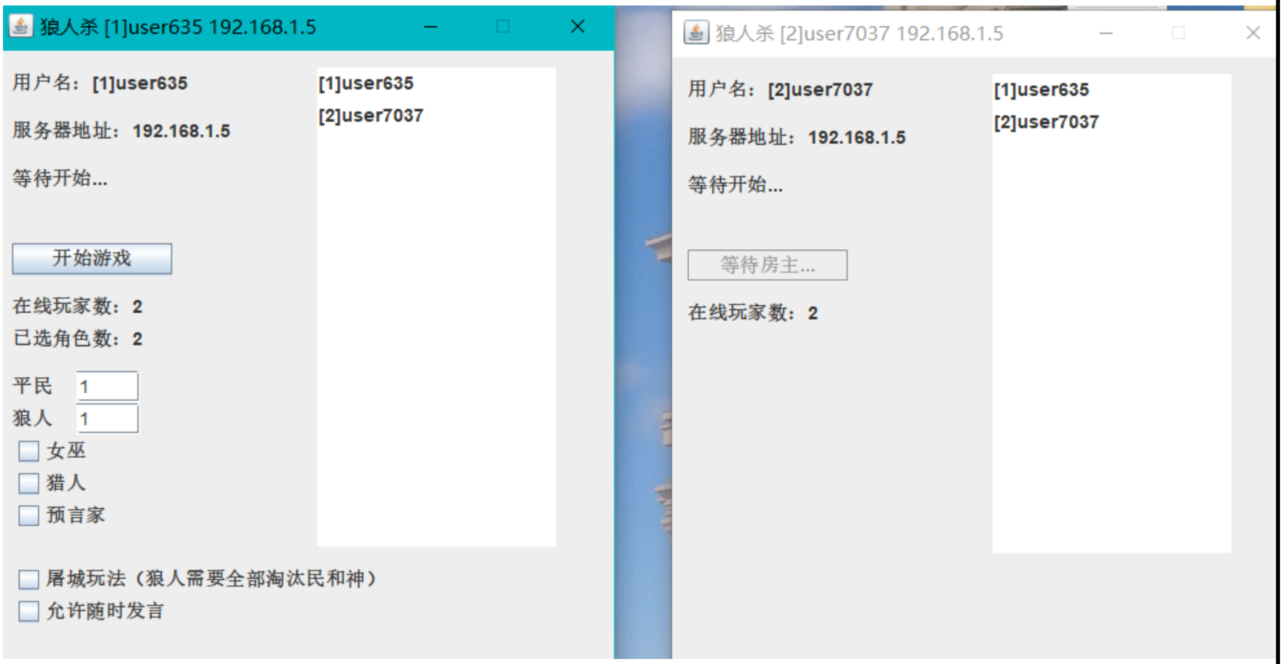
测试用例 1：


狼人杀 V.0.1
连接服务器
作为服务器

用户名 user635

服务器地址 192.168.1.5

测试用例 2：


狼人杀 [1]user635 192.168.1.5
开始游戏
狼人杀 [2]user7037 192.168.1.5
等待房主...

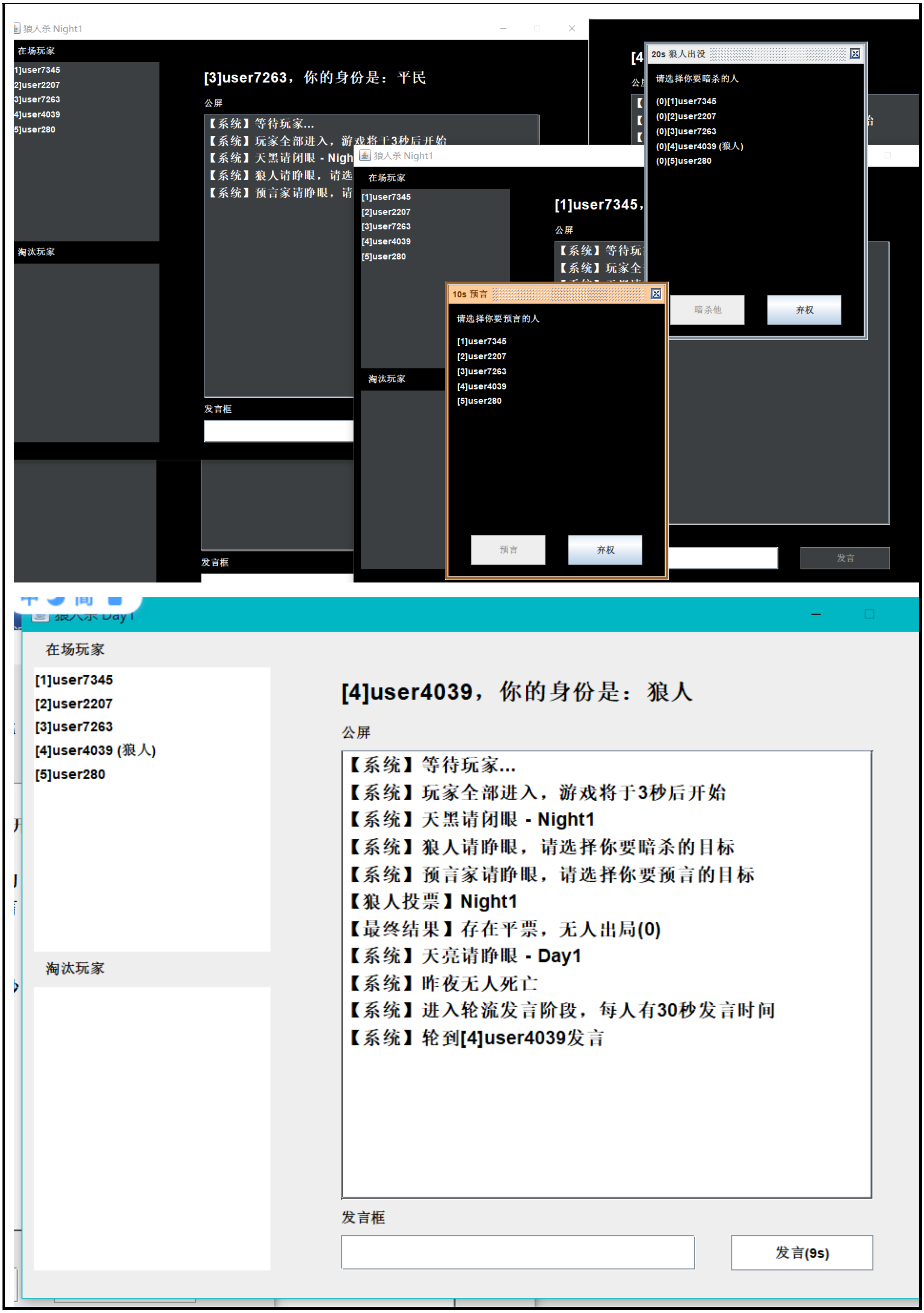
用户名：[1]user635
服务器地址：192.168.1.5
等待开始...
在线玩家数：2
已选角色数：2
平民 1
狼人 1
□ 女巫
□ 猎人
□ 预言家
□ 屠城玩法（狼人需要全部淘汰民和神）
□ 允许随时发言

[1]user635
[2]user7037

用户名：[2]user7037
服务器地址：192.168.1.5
等待开始...
在线玩家数：2

[1]user635
[2]user7037

测试用例 3：


狼人杀 Night1
在场玩家
1]user7345
2]user2207
3]user7263
4]user4039
5]user280
淘汰玩家
公屏
发言框
狼人杀 Night1
在场玩家
[1]user7345
[2]user2207
[3]user7263
[4]user4039
[5]user280
淘汰玩家
公屏
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
狼人杀 Day1
在场玩家
[1]user7345
[2]user2207
[3]user7263
[4]user4039 (狼人)
[5]user280
淘汰玩家
公屏
发言框
发言(9s)

[3]user7263，你的身份是：平民

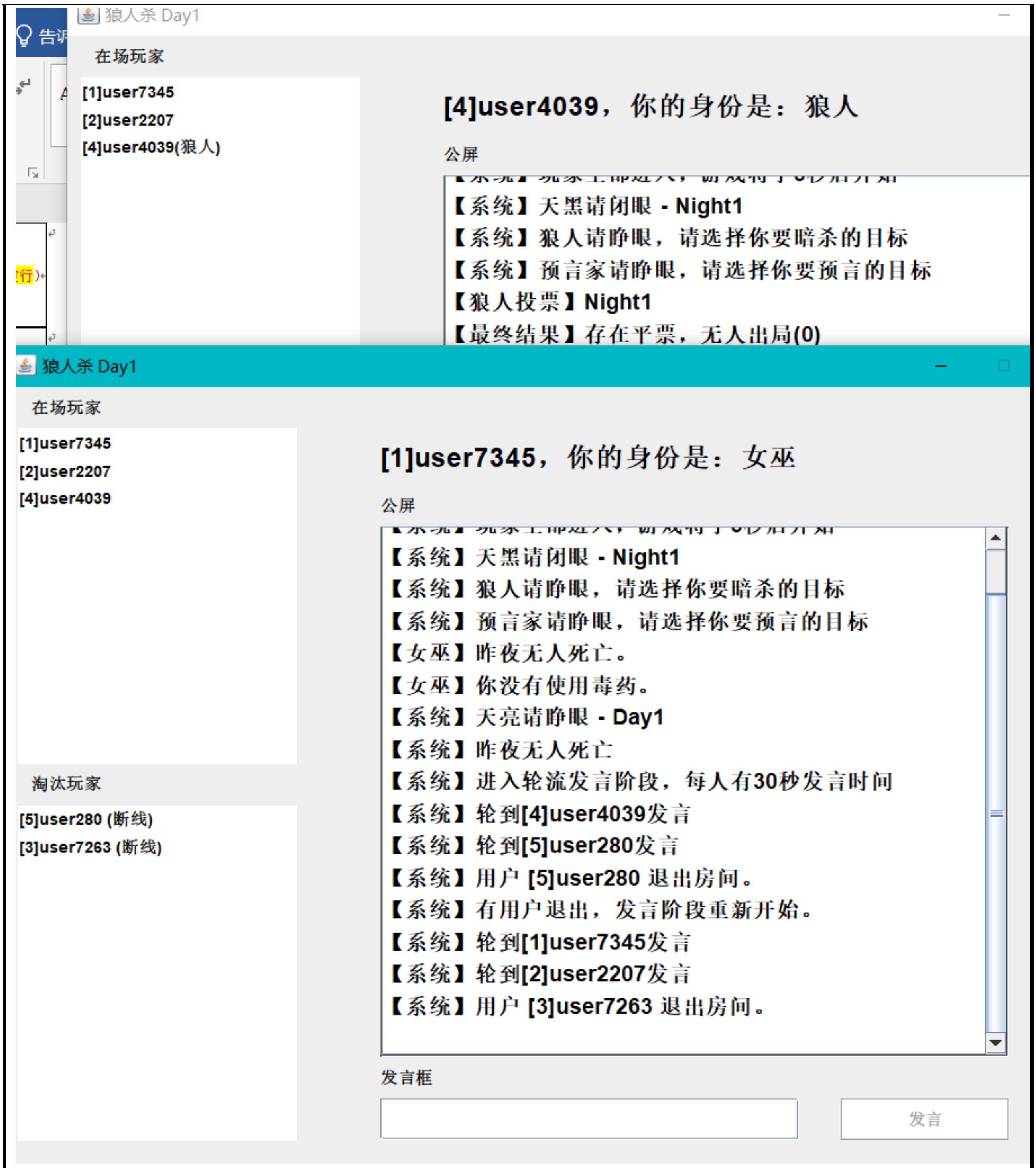
【系统】等待玩家...
【系统】玩家全部进入，游戏将于3秒后开始
【系统】天黑请闭眼 - Nigh
【系统】狼人请睁眼，请选
【系统】预言家请睁眼，请

[1]user7345，

【系统】等待玩
【系统】玩家全

[4]user4039，你的身份是：狼人

【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



[1]user7345
[2]user2207
[4]user4039(狼人)

[4]user4039，你的身份是：狼人

【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)

[1]user7345
[2]user2207
[4]user4039

[5]user280 (断线)
[3]user7263 (断线)

[1]user7345，你的身份是：女巫

【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。

程序源代码

```
//文件名 KillWolf.java
package KillWolf;
import java.net.Socket;

import java.util.Scanner;

import javax.swing.JFrame;

import KillWolf.Data.DataUser;
import KillWolf.Data.SelectionEvent;
import KillWolf.Data.SelectionEventCallBack;
import KillWolf.KillWolf.serverData;
import KillWolf.SocketDeal.GameMainThread;
import KillWolf.Window.GameMain;
import KillWolf.Window.LoginWindow;
import KillWolf.Window.SelectWindow;
import KillWolf.Window.WaitRoom;
import SocketServe.*;


public class KillWolf {
    public class serverData {
        public static String StartSpeakName = null;
        //public static int startSpeakUserArrayId;
        public static String nowSpeakUser;
        //public static int nowSpeakUserArrayId;
        public static String deathUser = null;      //狼人杀死/女巫解救
        public static String deathUser2 = null; //女巫毒死      为**NONE**表示无人死亡
        public static String deathUser3 = null; //猎人杀死
        public static String deathUser4 = null; //投票出局
        public static GameMainThread GameMainThread = null;
        public static int Day = 0;
        public static boolean night = false;
        public static boolean playMode = false; //true 表示屠城玩法，false 表示屠边玩法
        public static boolean freeSpeak = false;
        public static DataUser ServerUsers[] = new DataUser[50];
        public static int UsersLen = 0;
        public static int UsersRealLen = 0;
        public static SelectionEvent Selection[] = new SelectionEvent[200];
        public static int SelectionLen = 0;
        public static DataUser[] getUsersWithID(String identify) {
            int[] a = new int[ServerUsers.length];
            int n = 0;
            for(int i=0;i< UsersLen;i++) {
```

```
            if(ServerUsers[i] != null)
            if(ServerUsers[i].identify.equals(identify)) {
                a[n] = i;
                n++;
            }
        }
        DataUser[] b = new DataUser[n];
        for(int i=0;i<n;i++) b[i] = ServerUsers[a[i]];
        return b;
    }
    public static String[] ArrayDatatoName(DataUser[] d) {
        int[] a = new int[d.length];
        int n = 0;
        for(int i=0;i<d.length;i++) {
            if(d[i] != null)
            if(d[i].live) {
                a[n] = i;
                n++;
            }
        }
        String[] b = new String[n];
        for(int i=0;i<n;i++) {
            b[i] = d[a[i]].name;
        }
        return b;
    }
    public static Object[] getLivingUsers(boolean onlyName) {
        int[] a = new int[ServerUsers.length];
        int n = 0;
        for(int i=0;i<UsersLen;i++) {
            if(ServerUsers[i] != null)
            if(ServerUsers[i].live) {
                a[n] = i;
                n++;
            }
        }
        String[] b = new String[n];
        DataUser[] c = new DataUser[n];
        for(int i=0;i<n;i++) {
            if(onlyName) b[i] = ServerUsers[a[i]].name;
            else c[i] = ServerUsers[a[i]];
            System.out.println("LIVING : "+ServerUsers[a[i]].name);
        }
        if(onlyName) return b;
        else return c;
    }
    public static boolean isAllWolfDied() {
        DataUser[] b = getUsersWithID("狼人");
        for(int i=0;i<b.length;i++) if(b[i].live) return false;
        return true;
    }
    public static boolean isAllPeoDied() {
        DataUser[] b = getUsersWithID("平民");
        for(int i=0;i<b.length;i++) if(b[i].live) return false;
        return true;
    }
    public static boolean isAllGodDied() {
        DataUser[] b = getUsersWithID("猎人");
        for(int i=0;i<b.length;i++) if(b[i].live) return false;
        b = getUsersWithID("预言家");
        for(int i=0;i<b.length;i++) if(b[i].live) return false;
        b = getUsersWithID("女巫");
        for(int i=0;i<b.length;i++) if(b[i].live) return false;
        return true;
    }
    public static String whoWin() {
        for (int i=0;i<serverData.UsersLen;i++){
            if(serverData.ServerUsers[i]!=null){
                System.out.println(serverData.ServerUsers[i].name +" "
  +serverData.ServerUsers[i].identify+" "+serverData.ServerUsers[i].live);
            }
        }
        if(playMode) {
            if(isAllWolfDied()) return "平民";
            if(isAllPeoDied() && isAllGodDied()) return "狼人";
            return "无";
        }else {
            if(isAllWolfDied()) return "平民";
            if(isAllPeoDied() || isAllGodDied()) return "狼人";
            return "无";
        }
    }
    public static DataUser getUser(String name) {
        for(int i=0;i<UsersLen;i++) {
            if(ServerUsers[i] != null)
            if(ServerUsers[i].name.equals(name)) {
                return ServerUsers[i];
            }
				}
				return null;
			}

		}

		public static sServer server;
		public static sClient client;
		public static boolean freeSpeak = false;
		public static boolean isServer = false;
		public static String ServerIp;
		public static String Mip;
		public static String UserName;
		public static DataUser Users[] = new DataUser[50];
		public static int UsersLen = 0;
		public static int UsersRealLen = 0;
		public static LoginWindow loginwindow;
		public static WaitRoom WaitRoom;
		public static GameMain GameMain;
		public static SelectWindow SelectWindow;
		public static void main(String[] args) {
			//Scanner sn = new Scanner(System.in);
			Mip = sIP.getIP();

			loginwindow = new LoginWindow();
			loginwindow.reset(Mip);

			System.out.println("Server on IP: "+Mip);

		}


		public static void goWaitRoom() {


			loginwindow.window.setVisible(false);
			WaitRoom = new WaitRoom();
			WaitRoom.reset(Mip,UserName,isServer);
		}
		public static void goGameMain() {
			WaitRoom.window.setVisible(false);
			GameMain = new GameMain();
			GameMain.reset();

```

```
        }
}
// CallBack.java
package SocketServe;
/*
 *      author: wbx
 */
import java.net.Socket;

public interface CallBack {
        public void run(String text,Socket sk);
}
// sClient.java
package SocketServe;
/*
 *      author: wbx
 */
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class sClient {
        public int PORT = 15648;
        public boolean Log = true;
        public Socket sk = null;
        public CallBack cb = null;
        void wlog(String log) {
                if(Log) System.out.println(log);
        }
        public boolean close() {
                try {
                        sk.close();
                }catch(Exception e) {
                        e.printStackTrace();
                        return false;
                }
                return true;
        }
        public boolean send(String Text)
        {
                try{
                        PrintStream ps = new PrintStream(sk.getOutputStream());
                        ps.println(Text);
                        ps.flush();
                //if(Text.equals("END")) {
                //      sk.close();
                //}else {
                this.wlog(sk.getLocalPort()+" Client Send: "+Text);
                //}
            }
            catch(Exception e){
                e.printStackTrace();
                return false;
            }
            return true;
        }
        public boolean start(String ServerIp,CallBack cb) //= "127.0.0.1")
        {
            this.cb = cb;
            try{
                sk = new Socket(ServerIp,PORT);
            }
            catch(Exception e){
                e.printStackTrace();
                return false;
            }
            this.wlog("Connecting Server on "+ServerIp+","+PORT);
            try {
                ClientThread st = new ClientThread(sk);
                st.start();
            }catch(Exception e) {
                e.printStackTrace();
                return false;
            }
            this.wlog("Server is success connected on "+ServerIp+","+PORT);
            return true;
        }
        //�ͻ����◌̈���
        class ClientThread extends Thread
        {
            Socket sk;
            public ClientThread(Socket sk){
                this.sk= sk;
            }
            public void run() {
                try {
                    Scanner sn = new Scanner(sk.getInputStream());
                    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){
                e.printStackTrace();
                return "";
            }
            return MIP;
        }
        /**

        * 获取本地真正的 IP 地址，即获得有线或者无线 WiFi 地址。

        * 过滤虚拟机、蓝牙等地址

        * @author yins

        * @date 2018 年 8 月 12 日  下午 9:53:58

        */

        public class GetRealLocalIP {
        /**

        * 获取本地真正的 IP 地址，即获得有线或者无线 WiFi 地址。

        * 过滤虚拟机、蓝牙等地址

        * @author yins

        * @date 2018 年 8 月 12 日下午 9:56:35

        * @return

        */

        public static String getRealIP() {
        try {
        Enumeration<NetworkInterface> allNetInterfaces = NetworkInterface.getNetworkInterfaces();

        while (allNetInterfaces.hasMoreElements()) {
        NetworkInterface netInterface = (NetworkInterface) allNetInterfaces.nextElement();

        // 去除回环接口，子接口，未运行和接口

        if (netInterface.isLoopback() || netInterface.isVirtual()|| !netInterface.isUp())continue;
        if (!netInterface.getDisplayName().contains("Intel")&&
            !netInterface.getDisplayName().contains("Realtek")) continue;
```

```
Enumeration<InetAddress> addresses = netInterface.getInetAddresses();
System.out.println(netInterface.getDisplayName());

while (addresses.hasMoreElements()) {
InetAddress ip = addresses.nextElement();
if (ip != null) {
// ipv4

if (ip instanceof Inet4Address) {
System.out.println("ipv4 = " + ip.getHostAddress());

return ip.getHostAddress();

}

}

}

break;

}

} catch (SocketException e) {
System.err.println("Error when getting host ip address"

+ e.getMessage());

}

return null;

}

}
/*

  * This method is used to get all ip addresses from the network interfaces.
  * network interfaces: eth0, wlan0, l0, vmnet1, vmnet8

public static void getAllIpAddress() {
        try {
                //get all network interface
                Enumeration<NetworkInterface>                    allNetworkInterfaces                    =
NetworkInterface.getNetworkInterfaces();
            NetworkInterface networkInterface = null;

            //check if there are more than one network interface
            while (allNetworkInterfaces.hasMoreElements()) {
                //get next network interface
                networkInterface = allNetworkInterfaces.nextElement();
                //output interface's name
                //System.out.println("network interface: " + networkInterface.getDisplayName());

                                //get all ip address that bound to this network interface
                Enumeration<InetAddress> allInetAddress = networkInterface.getInetAddresses();

                InetAddress ipAddress = null;

                //check if there are more than one ip addresses
                //band to one network interface
                while (allInetAddress.hasMoreElements()) {
                    //get next ip address
                    ipAddress = allInetAddress.nextElement();
                    if (ipAddress != null && ipAddress instanceof Inet4Address) {

                        System.out.println("ip address: " + ipAddress.getHostAddress());
                    }
                }
            }

        } catch (SocketException e) {
            e.printStackTrace();
        }
    }//end method getAllIpAddress*/
}
//sSever.java
package SocketServe;
/*
*     author: wbx
*/
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class sServer {
    public int PORT = 15648;
    public boolean Log = true;
public ServerSocket ss = null;
public Socket[] users = new Socket[50];
public int usersLen = 0;
public CallBack cb = null;
void wlog(String log) {
    if(Log) System.out.println(log);
}
public boolean close() {
    try {
        ss.close();
    }catch(Exception e) {
        e.printStackTrace();
        return false;
    }
    return true;
}
public boolean sendto(String Text,Socket sk) {
    if(sk == null) return false;
    PrintStream ps;
    try {
        ps = new PrintStream(sk.getOutputStream());
        ps.println(Text);
        ps.flush();
        this.wlog("Send to "+sk.getPort()+" : "+Text);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        return false;
    }

    return true;
}
public boolean send(String Text) {
    try{
        for(int i=0;i<usersLen;i++) {
            Socket sk = users[i];
            if(sk == null) continue;
            PrintStream ps = new PrintStream(sk.getOutputStream());
            ps.println(Text);
            ps.flush();
            this.wlog("Send "+(i+1)+"/"+usersLen+" to "+sk.getPort()+" : "+Text+"####");
            Thread.sleep(10);
        }
    }
```

```
            catch(Exception e){
                e.printStackTrace();
                return false;
            }
            return true;
        }
        public boolean start(CallBack cb) //= "127.0.0.1")
        {
            this.cb = cb;
            try {
                ss = new ServerSocket(PORT);
                ServerThreadAC st = new ServerThreadAC();
                st.start();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                return false;
            }
            return true;
        }
        //�������¿ɔ��◌̈���
        public class ServerThreadAC extends Thread
        {
            //String ServerIp;
            int id = 0;
            boolean Log = true;
            //public ServerThreadAC(String ServerIp){
            //      this.ServerIp = ServerIp;
            //}
            public void run() {
                try {
                    if(Log) System.out.println("Server is open on "+PORT);
                    //��h��whileѫ�������ᶇ��Ö����ɔ��"�����
                    while(true){
                        Socket sk = ss.accept();//������������Ö�ڵ�����
                        ServerThread st = new ServerThread(sk);
                        st.start();
                    }
                }catch(Exception e) {
                    e.printStackTrace();
                }
            }
        }
        //�������ɔ��◌̈���
```

```
    public class ServerThread extends Thread
    {
        Socket sk;
        int id = 0;
        boolean Log = true;
        public ServerThread(Socket sk){
            this.sk= sk;
        }
        public void run() {
            try {
                users[usersLen] = sk;
                this.id = usersLen;
                usersLen++;
                if(Log) System.out.println("New User in "+sk.getPort());
                cb.run("New User: "+sk.getPort(),sk);
                Scanner sn = new Scanner(sk.getInputStream());
                while(true) {
                    if(sn.hasNextLine()) {
                        String line = sn.nextLine();//br.readLine();
                        //System.out.println("Server Receive from "+sk.getPort()+": "+line);
                        cb.run(line,sk);
                    }else {
                        try {
                                sk.sendUrgentData(0);
                            } catch (IOException e) {
                                System.out.println("User "+sk.getPort()+" has disconnected.");
                                cb.run("User Disconnect "+sk.getPort(),sk);
                                users[this.id] = null;
                                break;
                            }


                    }
                }
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        }
    }
}

//DataUser.java
package KillWolf.Data;

```

```
import java.net.Socket;

public class DataUser {
    public Socket sk;
    public int port;
    public String name;
    public boolean live;
    public String identify;
    public boolean hasConfirmID; //被预言家预言了，仅在预言家的客户端可以使用
    public boolean havePoison = false;
    public boolean haveAntidote = false;
}
//SelectCallBack.java
package KillWolf.Data;

public interface SelectCallBack {
    public void run(String Selection);
}
// SelectionEvent.java
package KillWolf.Data;

import KillWolf.KillWolf;
import SocketServe.sServer;

public class SelectionEvent {
    public static SelectionEvent Selection[] = new SelectionEvent[200];
    public static int SelectionLen = 0;
    public static void UserQuit(String name) {
        for(SelectionEvent i : Selection) {
            if(i!=null) {
                if(i.open) {
                    for(int j=0;j<i.openusers.length;j++) {
                        if(i.openusers[j].equals(name)) {
                            if(i.target[j] == null) {
                                i.target[j] = "**NONE**";
                                i.isFinish();
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
    public boolean open;

    public String[] users; //要选的目标人
    public int[] count;

    public String[] openusers; //需要选人的人
    public String[] target;

    public String ReceiveID;
    public int timeLimit;
    public SelectionEventCallBack cb;
    public sServer server;
    public SelectionEvent(String ReceiveID,String[] users,int timeLimit,SelectionEventCallBack cb) {
        this.users = new String[users.length];
        this.count = new int[users.length];
        this.cb = cb;
        this.ReceiveID = ReceiveID;
        this.timeLimit= timeLimit;
        for(int i=0;i<users.length;i++)
            if(users[i] != null) this.users[i] = users[i];
            else    this.users[i] = "**NONE**";
    }
    public void update(String user,String toUser) {
        for(int i=0;i<users.length;i++) {
            if(users[i].equals(user)) {
                for( int j=0;j<openusers.length;j++) {
                    if(openusers[j].equals(user)) {
                        target[j] = toUser;
                    }
                }
            }
            if(users[i].equals(toUser))        count[i]++;
        }
        System.out.println("Finish ? " + isFinish());
    }
    public boolean isFinish() {
        System.out.println("###### IS FINISH ? " + target.length);
        for(String i : target) {
            if(i == null) {
                //System.out.println("########################");
                return false;
            }
        }
        closeSelection(this.server,false);
        return true;
    }
    public int[] getResult() {
        return count;
    }
    public boolean closeSelection(sServer server,boolean run) {
        //if(run) if(this.cb != null) this.cb.run(ReceiveID, count, users, target);
        System.out.println("CLOSESELECTION");
        if(this.open) {
            this.open = false;
            for(int i=0;i<openusers.length;i++){
                if(openusers[i]!=null){
                    DataUser user = KillWolf.serverData.getUser(openusers[i]);
                    if(user!=null)
                    server.sendto("CLOSESELECTION"+ReceiveID,user.sk);
                }
            }
            //没选的默认弃权
            if(this.cb != null) this.cb.run(ReceiveID, count, users, target, openusers);
        }
        return true;
    }
    public void openSeletion(sServer server,DataUser du[],int RealLen,
String WindowTitle,String ButtonTitle,String Tip,boolean isDark) {
        this.open = true;
        this.server = server;
        Selection[SelectionLen] = this;
        SelectionLen++;



        String STR = "";
        for(String i : users) {
            STR = STR +"\t"+ i;
        }
        openusers = new String[RealLen];
        this.target = new String[RealLen];
        int j = 0;
        for(DataUser i : du) {
            if(i != null) {
                openusers[j] = i.name;
                server.sendto("OPENSELETION"+timeLimit+"\t\t"+ReceiveID+"\t"+
STR+"\t\t"+WindowTitle+"\t\t"+Tip+"\t\t"+ButtonTitle+"\t\t"+ (isDark?"DARK":"LIGHT"),i.sk);
 //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;
```

```

public class ClientDeal implements CallBack {
    //public SelectWindow SelectWindow;
    public void run(String text,Socket sk){
        System.out.println(sk.getLocalPort()+" Client Receive: "+text+"####");
        if(text.startsWith("ENTERSUCCESS"))
        {
            KillWolf.UserName = text.substring(12);
            KillWolf.loginwindow.LoginButton.setEnabled(false);
            KillWolf.loginwindow.userName.setEnabled(false);
            KillWolf.goWaitRoom();
        }else if(text.startsWith("ENTERFAIL")){
            JOptionPane.showMessageDialog(null, text.substring(9));
            /*try {
                sk.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }*/
        }else if(text.startsWith("NEWWAITUSERLIST")){
            String users[] = text.substring(15).split("\t");
            //KillWolf.WaitRoom.setList
            for(int i=0;i<users.length;i++) {
                KillWolf.Users[i]= new DataUser();
                KillWolf.Users[i].name = users[i];
            }
            KillWolf.UsersRealLen = KillWolf.UsersLen = users.length;
            //KillWolf.WaitRoom.UserList.setListData(users);
            KillWolf.WaitRoom.setUserCount(users);

            System.out.println(sk.getLocalPort()+" USERS: " + KillWolf.UserName+"\t"
                                    +KillWolf.UsersRealLen);
        }else if(text.startsWith("Disconnect from Server")){
            if(KillWolf.GameMain != null) {
                KillWolf.GameMain.getFrame().setAlwaysOnTop(true);
                JOptionPane.showMessageDialog(KillWolf.GameMain.getFrame(),
                   "服务器连接丢失");
            }else {
                JOptionPane.showMessageDialog(null, "服务器连接丢失");
            }
            //JOptionPane.setDefaultLocale(null);
            System.exit(0);
        }else if(text.startsWith("USERQUIT")){
            String user = text.substring(8),STR = "";
        for(int i=0;i<KillWolf.Users.length;i++) {
            if(KillWolf.Users[i] != null)
                if(KillWolf.Users[i].name.equals(user)) {
                    KillWolf.Users[i] = null;
                }else {
                    STR = STR + "\t" +KillWolf.Users[i].name;
                }
        }
        String users[] = STR.substring(1).split("\t");
        //KillWolf.WaitRoom.UserList.setListData();
        KillWolf.UsersRealLen--;
        //KillWolf.WaitRoom.settitle("玩家数量："+ KillWolf.UsersRealLen)//+"  总数："
                  +KillWolf.UsersLen);
        KillWolf.WaitRoom.setUserCount(users);

        if(KillWolf.GameMain == null) return;

        int usdi=0,temp=1;
        for(int i=0;i<KillWolf.GameMain.UserOutListData.length;i++){
            if(KillWolf.GameMain.UserOutListData[i].startsWith(user+" ")){
                temp=0;
                KillWolf.GameMain.UserOutListData[i] =
                         KillWolf.GameMain.UserOutListData[i] + "(断线)";
                break;
            }//else usod[i]=KillWolf.GameMain.UserOutListData[i];
        }

        String usod[]=new String[KillWolf.GameMain.UserOutListData.length+temp];
        for(int i=0;i<KillWolf.GameMain.UserOutListData.length;i++){
            usod[i] = KillWolf.GameMain.UserOutListData[i];
        }

        if(temp==1){
            String usda[]=new String[KillWolf.GameMain.UserListData.length-1],
                    usd[]=new String[KillWolf.GameMain.UserListData.length-1],
                    usdt[]=new String[KillWolf.GameMain.UserListData.length-1];
            usod[usod.length-1] = user+" (断线)";
            for(int i=0;i<KillWolf.GameMain.UserListData.length;i++) {
                if(KillWolf.GameMain.UserListData[i].equals(user))
                {
                    usod[usod.length-1] = usod[usod.length-1] + KillWolf.GameMain.UserListTipData[i];
                }else{
                    usd[usdi]=KillWolf.GameMain.UserListData[i];
```

```
                            usdt[usdi]=KillWolf.GameMain.UserListTipData[i];
                            usda[usdi]=usd[usdi]+usdt[usdi];
                            usdi++;
                        }
                    }
                    KillWolf.GameMain.UserListData = usd;
                    KillWolf.GameMain.UserListTipData = usdt;
                    KillWolf.GameMain.getUserList().setListData(usda);
                }
                KillWolf.GameMain.UserOutListData = usod;
                KillWolf.GameMain.getOutUserList().setListData(usod);
                KillWolf.GameMain.addToScreen("【系统】用户 "+user+" 退出房间。");
                //KillWolf.UsersLen = KillWolf.Users.length;
            }else if(text.startsWith("GAMESTART")) {
                String[] STR = text.substring(9).split("#u#");
                DataUser[] du = new DataUser[STR.length];
                if(STR[0].equals("TRUE")){
                    KillWolf.freeSpeak = true;
                }else{
                    KillWolf.freeSpeak = false;
                }
                for(int i=0;i<STR.length-1;i++) {
                    du[i] = new DataUser();
                    String[] p = STR[i+1].split("\t");

                    du[i].name = p[0];
                    du[i].identify = p[1];
                }
                KillWolf.Users = du;
                KillWolf.goGameMain();
            }else if(text.startsWith("SELECT")) {
                if(KillWolf.SelectWindow != null) KillWolf.SelectWindow.SELECTInfo(text.substring(6));
            }else if(text.startsWith("OPENSELETION")) {
                String a[] = (text.substring(12).split("\t\t")) ;
                JFrame fra = null;
                if(KillWolf.GameMain!=null) fra = KillWolf.GameMain.getFrame();
                KillWolf.SelectWindow = new SelectWindow(a[3],true,a[1],true,fra);
                if(a[6].equals("DARK"))KillWolf.SelectWindow.setColorMode(true);
                String[] b = a[2].split("\t");//KillWolf.GameMain.UserListData;

                String myId = KillWolf.GameMain.userSelf.identify;
                String[] tip = new String[b.length];
                if(myId.equals("狼人")) {//|| myId.equals("预言家")
                    for (int j = 0; j < b.length; j++) {
```

```
                    for (int i = 0; i < KillWolf.UsersLen; i++) {
                        if (KillWolf.Users[i] != null) {
                            if (KillWolf.Users[i].name.equals(b[j])) {
                                if(myId.equals("狼人")){
                                    if(KillWolf.Users[i].identify.equals("狼人")){
                                        tip[j] = "("+KillWolf.Users[i].identify+")";
                                    }
                                }/*else if(myId.equals("预言家")&& KillWolf.Users[i].hasConfirmID){
                                    {
                                        if(KillWolf.Users[i].identify.equals("狼人")){
    “)";
                                        }
                                    }

                                }*/else{
                                    tip[j] = "";
                                }
                            }
                        }
                    }
                }

                KillWolf.SelectWindow.start(a[4], a[5], b, true,null, Integer.parseInt(a[0]),tip);
            }else if(text.startsWith("CLOSESELECTION")) {
                KillWolf.SelectWindow.getSelectionAndClose(text.substring(14));
            }else if(text.startsWith("MESSAGE")) {
                if(text.substring(7).startsWith("【系统】天黑请闭眼")){
                    KillWolf.GameMain.setDark();
                    KillWolf.GameMain.getFrame().setTitle("狼人杀  "+text.substring(19));
                    //天黑请闭眼  - Night
                }
                if(text.substring(7).startsWith("【系统】天亮请睁眼")){
                    KillWolf.GameMain.getFrame().setTitle("狼人杀  "+text.substring(19));
                    KillWolf.GameMain.setWhite();
                }
                KillWolf.GameMain.addToScreen(text.substring(7));
            }else if(text.startsWith("YUYANJIASTART")){
                if(!KillWolf.GameMain.userSelf.identify.equals("预言家"))return;

                KillWolf.SelectWindow = new SelectWindow("预言",false,"预言家",true,KillWolf.GameMain.getFrame());
                KillWolf.SelectWindow.setColorMode(true);
```

```
//String[] users=new String[]
SelectCallBack cb = Selection -> {
    System.out.println("SelectFinal:" + Selection);
    //System.out.println("预言："+st);
    if(Selection.equals("**NONE**")) {
        KillWolf.GameMain.addToScreen("【预言】你没有进行预言。");
        KillWolf.SelectWindow.getSelectionAndClose("预言家");
        KillWolf.client.send("YUYANEND");
        return;
    }
    String str = "";
    for(int i=0;i<KillWolf.UsersLen;i++){
        if(KillWolf.Users[i] == null) continue;
        if(KillWolf.Users[i].name.equals(Selection)){
            for(int j=0;j<KillWolf.GameMain.UserListData.length;j++){
                System.out.println("#################List:" + KillWolf.GameMain.UserListData[j]+"#");
                if(KillWolf.GameMain.UserListData[j].equals(Selection)){
                    str = (KillWolf.Users[i].identify.equals("狼人")?"坏人":"好人");
                    KillWolf.GameMain.UserListTipData[j] = "(" + str + ")";
                    str = "预言："+KillWolf.Users[i].name+"是"+str;
                    KillWolf.SelectWindow.CenterLabel.setText(str);
                    KillWolf.SelectWindow.CenterLabel.setForeground(Color.yellow);
                    break;
                }
            }

        }
    }
    String [] usn = KillWolf.GameMain.UserListData;
    String [] usnt = KillWolf.GameMain.UserListTipData;
    String [] usna = new String[usn.length];
    for (int i=0;i<usn.length;i++) {
        usna[i]=usn[i]+" "+usnt[i];
    }
    KillWolf.GameMain.getUserList().setListData(usna);
    KillWolf.SelectWindow.getSelectionAndClose("预言家");
    KillWolf.GameMain.addToScreen("【预言】"+str.substring(3));
    KillWolf.client.send("YUYANEND");
    //JOptionPane.showMessageDialog(KillWolf.GameMain.getFrame(),str);

};
KillWolf.SelectWindow.start("请选择你要预言的人", "预言", KillWolf.GameMain.UserListData,true,cb,20,KillWolf.GameMain.UserListTipData);
```

```
            SleepThread sth = new SleepThread("预言家");
            sth.start();

            //System.out.println("预言："+st);
            //if(st.equals("**NONE**")) {
        //          KillWolf.GameMain.addToScreen("【预言】你没有进行预言。");
            //}
        }
        else if(text.startsWith("HUNTERSTART")){
            if(!KillWolf.GameMain.userSelf.identify.equals("猎人"))return;
            int res=JOptionPane.showConfirmDialog(KillWolf.GameMain.getFrame(), "你已死亡，是否使用技能？", "消息", JOptionPane.YES_NO_OPTION);
            if(res==JOptionPane.YES_OPTION){
                if(text.substring(11).startsWith("LAST"))KillWolf.client.send("HUNTSTARTLASTYES");
                else KillWolf.client.send("HUNTSTARTYES");
                System.out.println("选择是后执行的代码");    //点击“是”后执行这个代码块
            }else{
                if(text.substring(11).startsWith("LAST"))KillWolf.client.send("HUNTSTARTLASTYES");
                else KillWolf.client.send("HUNTSTARTNO");
                System.out.println("选择否后执行的代码");    //点击“否”后执行这个代码块
                return;
            }

        }else if(text.startsWith("HUNTSELECT")){
            if(!KillWolf.GameMain.userSelf.identify.equals("猎人"))return;

            KillWolf.SelectWindow = new SelectWindow("开枪",false,"猎人技能",true,KillWolf.GameMain.getFrame());
            KillWolf.SelectWindow.setColorMode(true);

            SelectCallBack cb = Selection -> {
                System.out.println("SelectFinal:" + Selection);
                //System.out.println("预言："+st);
                if(Selection.equals("**NONE**")) {
                    KillWolf.GameMain.addToScreen("【猎人】你没有开枪。");
                    KillWolf.SelectWindow.getSelectionAndClose("猎人技能");
                    if(text.substring(10).startsWith("LAST"))KillWolf.client.send("HUNTSELELASTNO");
                    else KillWolf.client.send("HUNTSELENO");
                    return;
                }
                KillWolf.GameMain.addToScreen("【猎人】你对"+Selection+"进行射击。");
                KillWolf.SelectWindow.getSelectionAndClose("猎人技能");
                if(text.substring(10).startsWith("LAST"))KillWolf.client.send("HUNTSELELASTYES"+KillWolf.UserName+"\t"+Selection);
                else KillWolf.client.send("HUNTSELEYES"+KillWolf.UserName+"\t"+Selection);
            };
            KillWolf.SelectWindow.start("你要对谁开枪？", "开枪", KillWolf.GameMain.UserListData,true,cb,20,null);
            SleepThread sth = new SleepThread("猎人技能");
            sth.start();
        }else if(text.equals("FORBIDSPEAK")){
            KillWolf.GameMain.forbidSubmit();
        }else if(text.equals("ENABLESPEAK")){
            KillWolf.GameMain.ableSubmit();
        }else if(text.equals("ENABLESPEAKEND")){
            KillWolf.freeSpeak = true;
            KillWolf.GameMain.ableSubmit();


            String usda[]=new String[KillWolf.GameMain.UserListData.length];
            for(int i=0;i<KillWolf.GameMain.UserListData.length;i++) {
                for(int j=0;j<KillWolf.UsersLen;j++){
                    if(KillWolf.Users[j]==null)continue;
                    if(KillWolf.Users[j].name.equals(KillWolf.GameMain.UserListData[i])){
                        usda[i]=KillWolf.GameMain.UserListData[i]+" ("+KillWolf.Users[j].identify+")";
                    }
                }
            }
            KillWolf.GameMain.getUserList().setListData(usda);

        }else if(text.equals("ENABLESPEAKAC")){
            KillWolf.GameMain.ableSubmit();
            KillWolf.GameMain.startSpeakClock();
        }else if(text.startsWith("WIZARDSTART")) {
            if(!KillWolf.GameMain.userSelf.identify.equals("女巫"))return;
            String name = text.substring(11);
            if(KillWolf.GameMain.userSelf.haveAntidote) {
                if(name.equals("**NONE**")) {
                    KillWolf.client.send("WIZARDANTINO");
                    return;
                }
                int res=JOptionPane.showConfirmDialog(KillWolf.GameMain.getFrame(), "昨夜"+name+"死亡，是否使用解药？", "消息", JOptionPane.YES_NO_OPTION);
        if(res==JOptionPane.YES_OPTION){
            KillWolf.GameMain.userSelf.haveAntidote = false;
            KillWolf.client.send("WIZARDANTIYES");
            System.out.println("选择是后执行的代码");      //点击“是”后执行这个代码块
        }else{
            KillWolf.client.send("WIZARDANTINO");
            System.out.println("选择否后执行的代码");      //点击“否”后执行这个代码块
            return;
        }
    }else {
        KillWolf.client.send("WIZARDANTINONE");
    }
}else if(text.startsWith("WIZARDPOS")) {
    if(!KillWolf.GameMain.userSelf.identify.equals("女巫"))return;

    KillWolf.SelectWindow = new SelectWindow("毒药",false,"女巫毒药",true,KillWolf.GameMain.getFrame());
    KillWolf.SelectWindow.setColorMode(true);
    //String[] users=new String[]
    SelectCallBack cb = Selection -> {
        System.out.println("SelectFinal:" + Selection);
        //System.out.println("预言："+st);
        if(Selection.equals("**NONE**")) {
            KillWolf.GameMain.addToScreen("【女巫】你没有使用毒药。");
            KillWolf.SelectWindow.getSelectionAndClose("女巫毒药");
            KillWolf.client.send("WIZARDPOSNO");
            return;
        }
        KillWolf.GameMain.addToScreen("【女巫】你对"+Selection+"使用了毒药。");
        KillWolf.SelectWindow.getSelectionAndClose("女巫毒药");
        KillWolf.client.send("WIZARDPOSYES"+Selection);
        //JOptionPane.showMessageDialog(KillWolf.GameMain.getFrame(),str);
    };
    KillWolf.SelectWindow.start("你要对谁使用毒药？", "使用", KillWolf.GameMain.UserListData,true,cb,20,KillWolf.GameMain.UserListTipData);
    SleepThread sth = new SleepThread("女巫毒药");
    sth.start();
}else if(text.startsWith("USERDIED")){
    String user = text.substring(8),STR = "";
    /*for(int i=0;i<KillWolf.Users.length;i++) {
        if(KillWolf.Users[i] != null)
            if(KillWolf.Users[i].name.equals(user)) {
                KillWolf.Users[i].live = false;
            }else {
                  //STR = STR + "\t" +KillWolf.Users[i].name;
              }
      }
      String users[] = STR.substring(1).split("\t");
      *///KillWolf.WaitRoom.UserList.setListData();
      //KillWolf.UsersRealLen--;
      //KillWolf.WaitRoom.settitle("玩家数量："+ KillWolf.UsersRealLen)//+"总数："+KillWolf.UsersLen);
      //KillWolf.WaitRoom.setUserCount(users);

      //if(KillWolf.GameMain == null) return;
      int pp=0;
      for(;pp<KillWolf.UsersLen;pp++){
          if(KillWolf.Users[pp]==null)continue;
          if(KillWolf.Users[pp].name.equals(user)){
              pp=-1;
              break;
          }
      }
      if(pp!=-1) return;

      String usda[]=new String[KillWolf.GameMain.UserListData.length-1],
              usd[]=new String[KillWolf.GameMain.UserListData.length-1],
              usdt[]=new String[KillWolf.GameMain.UserListData.length-1],
              usod[]=new String[KillWolf.GameMain.UserOutListData.length+1];
      int usdi=0;
      for(int i=0;i<KillWolf.GameMain.UserOutListData.length;i++){
          usod[i]=KillWolf.GameMain.UserOutListData[i];
      }
      usod[usod.length-1] = user+" (淘汰)";
      if(user.equals(KillWolf.UserName)){
KillWolf.GameMain.getIdentity().setText(KillWolf.UserName+"("+KillWolf.GameMain.userSelf.identify+")，你已被淘汰");
      }
      for(int i=0;i<KillWolf.GameMain.UserListData.length;i++) {
          if(KillWolf.GameMain.UserListData[i].equals(user))
          {
              usod[usod.length-1] = usod[usod.length-1] + KillWolf.GameMain.UserListTipData[i];
          }else{
              usd[usdi]=KillWolf.GameMain.UserListData[i];
              usdt[usdi]=KillWolf.GameMain.UserListTipData[i];
              usda[usdi]=usd[usdi]+usdt[usdi];
```

```
                usdi++;
            }
        }

        KillWolf.GameMain.UserListData = usd;
        KillWolf.GameMain.UserListTipData = usdt;
        KillWolf.GameMain.UserOutListData = usod;
        KillWolf.GameMain.getUserList().setListData(usda);
        KillWolf.GameMain.getOutUserList().setListData(usod);
//      System.out.println(text);
//      String[] result = text.substring(8).split("\t");
//      String[] users = new String[KillWolf.GameMain.UserListData.length- result.length];
//      String[] outUsers = new String[KillWolf.GameMain.UserOutListData.length+result.length];
//      for(int j=0;j< result.length;j++){
//          if(result[j].equals(KillWolf.GameMain.userSelf.name))KillWolf.GameMain.userSelf.live=false;
//          for(int k=0;k< KillWolf.GameMain.UserListData.length;k++){
//              if(KillWolf.GameMain.UserListData[k].startsWith(result[j])){
//                  result[j]=KillWolf.GameMain.UserListData[k];
//                  KillWolf.GameMain.UserListData[k]=null;
//              }
//          }
//      }
//      int i=0;
//      for (;i<KillWolf.GameMain.UserOutListData.length;i++){
//          outUsers[i] = KillWolf.GameMain.UserOutListData[i];
//      }
//      int j=i;
//      for(;i<outUsers.length;i++){
//          outUsers[i]=result[i-j]+"(淘汰)";
//      }
//      KillWolf.GameMain.UserOutListData=outUsers;
//      KillWolf.GameMain.getOutUserList().setListData(outUsers);
//      int i2 = 0;
//      for(int i1 = 0;i1< KillWolf.GameMain.UserListData.length;i1++){
//          if(KillWolf.GameMain.UserListData[i1]!=null&& !KillWolf.GameMain.UserListData[i1].equals("**NONE**")){
//              users[i2++]=KillWolf.GameMain.UserListData[i1];
//          }
//      }
//      KillWolf.GameMain.UserListData = users;
//      KillWolf.GameMain.getUserList().setListData(users);
```

```
        }



            //    +timeLimit+"\t\t"+ReceiveID+"\t"+STR,du[j].sk); //RECEIVEID \t\t user1 \t user2

    }

}


class SleepThread extends Thread{
    String sss;
    public SleepThread(String a) {
        this.sss = a;
    }
    public void run() {
        try {
            Thread.sleep(20*1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        KillWolf.SelectWindow.getSelectionAndClose(sss);
    }
}
// GameMainThread.java
package KillWolf.SocketDeal;

import KillWolf.KillWolf;
import KillWolf.Data.DataUser;
import KillWolf.Data.SelectionEvent;
import KillWolf.Window.GameMain;

public class GameMainThread extends Thread{
    public boolean YuYanCompleted = false;
    public boolean WizardCompleted = false;
    public void run() {
        KillWolf.serverData.deathUser = "**NONE**";
        KillWolf.serverData.deathUser2 = "**NONE**";
        KillWolf.serverData.deathUser3 =   "**NONE**";
        KillWolf.serverData.deathUser4 =   "**NONE**";
        gameNight();
    }
    public void gameNight(){
        if(KillWolf.serverData.GameMainThread.isWin()) return;

        KillWolf.serverData.Day++;
        KillWolf.serverData.night = true;


        KillWolf.server.send("MESSAGE【系统】天黑请闭眼 - Night"+KillWolf.serverData.Day);
        KillWolf.server.send("FORBIDSPEAK");

        //KillWolf.server.send("ENABLESPEAK");

        String[] allUserName = (String[])KillWolf.serverData.getLivingUsers(true);
        //给狼人弹窗
        DataUser[] wolfUser0 = KillWolf.serverData.getUsersWithID("狼人");
        DataUser[] wolfUser = new DataUser[wolfUser0.length];
        int wi = 0;
        for(int k=0;k<wolfUser0.length;k++) {
            if(wolfUser0[k].live) {
                wolfUser[wi] = wolfUser0[k];
                wi++;
            }
        }
        //String[] wolfUserName = KillWolf.serverData.ArrayDatatoName(wolfUser);

        try {
            Thread.sleep(2*1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        KillWolf.server.send("MESSAGE【系统】狼人请睁眼，请选择你要暗杀的目标");
        KillWolf.serverData.Selection[KillWolf.serverData.SelectionLen] = new SelectionEvent(Integer.toString(KillWolf.serverData.SelectionLen),allUserName,30,new WolfSelect());
        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);
        KillWolf.serverData.SelectionLen++;
        //预言家弹窗
        KillWolf.server.send("MESSAGE【系统】预言家请睁眼，请选择你要预言的目标");
        DataUser d[] = KillWolf.serverData.getUsersWithID("预言家");
        if (d.length == 1 && d[0].live) {

            YuYanCompleted = false;
            KillWolf.server.sendto("YUYANJIASTART",d[0].sk);
        }else YuYanCompleted = true;

        //女巫弹窗在狼人结束才开始，此处先置为 false
        WizardCompleted = false;


        //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。
        try {
            Thread.sleep(20*1000);
            YuYanCompleted = true;//等 20 秒，预言一定已经完成了
            gameDay();
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void gameWizard(String diedUser) {
        DataUser d[] = KillWolf.serverData.getUsersWithID("女巫");
        //System.out.println(null.length);
        if (d.length == 1    && d[0].live) {
            WizardCompleted = false;
            KillWolf.server.sendto("WIZARDSTART"+diedUser,d[0].sk);
        }else {
            if(!diedUser.equals("**NONE**")) {
                KillWolf.serverData.getUser(diedUser).live = false;
            }
            WizardCompleted = true;
            KillWolf.serverData.GameMainThread.gameDay();
        }
    }
    boolean isRunGameDay = false;
    public void gameDay() {
        if(isRunGameDay) return;
        if(!YuYanCompleted || !WizardCompleted) return;
        //如果都完成了则继续
        String death = KillWolf.serverData.deathUser;
        String death2 = KillWolf.serverData.deathUser2;
        isRunGameDay = true;

        KillWolf.server.send("MESSAGE【系统】天亮请睁眼 - Day"+KillWolf.serverData.Day);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        String death3 = KillWolf.serverData.deathUser3;
        if(death2.equals("**NONE**") && death.equals("**NONE**")) {
            KillWolf.server.send("MESSAGE【系统】昨夜无人死亡");
        }else {
            if(!death.equals("**NONE**")){
                DataUser u = KillWolf.serverData.getUser(death);
                if(u!=null) u.live =false;
                KillWolf.server.send("USERDIED"+death);
            }
            if(!death2.equals("**NONE**")){
                DataUser u = KillWolf.serverData.getUser(death2);
                if(u!=null) u.live =false;
                KillWolf.server.send("USERDIED"+death2);
            }
            if(!death3.equals("**NONE**")){
                DataUser u = KillWolf.serverData.getUser(death3);
                if(u!=null) u.live =false;
                KillWolf.server.send("USERDIED"+death3);
            }
//KillWolf.server.send("USERDIED"+(death.equals("**NONE**")?"":death)+(death2.equals("**NONE**")?"":("\t"+death2))+(death3.equals("**NONE**")?"":("\t"+death3)));
            if(death2.equals("**NONE**"))KillWolf.server.send("MESSAGE【系统】昨夜"+(death.equals("**NONE**")?"":death)+(death3.equals("**NONE**")?"":("、"+death3))+"死亡");
            else KillWolf.server.send("MESSAGE【系统】昨夜"+(death.equals("**NONE**")?"":death) +(death2.equals("**NONE**")?"":("、"+death2))+(death3.equals("**NONE**")?"":("、"+death3))+"死亡");
        }

        if(KillWolf.serverData.GameMainThread.isWin())return;

        //判断猎人是否存活，若死亡，向猎人弹窗，使用技能
        DataUser user = KillWolf.serverData.getUser(death);
        DataUser user2 = KillWolf.serverData.getUser(death2);
        if(user == null && user2 == null) dayWork();
        else if(user != null && user.identify.equals("猎人")){KillWolf.server.sendto("HUNTERSTART",user.sk);}
        else if(user2 != null && user2.identify.equals("猎人")){KillWolf.server.sendto("HUNTERSTART",user2.sk);}
        else dayWork();

        //向所有玩家弹窗，投票

        //向公屏输出，投票结果，若平票，则再次投票

        //再次判断猎人是否存活，若死亡，向猎人弹窗

//        }
        //KillWolf.server.send("MESSAGE【系统】"+whoWin+"胜，将在 5 秒后退出该窗口");
        //重置游戏
    }
    class TimeTha extends Thread{
        public void run() {
            try {
                Thread.sleep(1000*120);
                gameVote();
            }catch(Exception e) {
                e.printStackTrace();
            }
        }
    }
    public void dayWork (){
        String death = KillWolf.serverData.deathUser;
        String death2 = KillWolf.serverData.deathUser2;
        //String death3 = KillWolf.serverData.deathUser3;
        String firstdeath = "";


        //}else if(death2.equals("**NONE**") && !death.equals("**NONE**")){
//            //KillWolf.server.send("MESSAGE【系统】昨夜"+death+"死亡");
//            firstdeath = death;
//        }else if(death2.equals("**NONE**") && death.equals("**NONE**")) {
//            //KillWolf.server.send("MESSAGE【系统】昨夜无人死亡");
//        }
//        else if(!death2.equals("**NONE**") && !death.equals("**NONE**")){
//            firstdeath = death;
//            //KillWolf.server.send("MESSAGE【系统】昨夜"+death+"、"+death2+"死亡");
//        }
        if(!death.equals("**NONE**") ){
            firstdeath = death;
        }else if(!death2.equals("**NONE**") ){
            firstdeath = death2;
        }

        if(KillWolf.serverData.GameMainThread.isWin())return;
        //判断是否获胜

```

```
//玩家轮流发言
if(KillWolf.serverData.freeSpeak){
    KillWolf.server.send("MESSAGE【系统】现在有 120 秒的交流时间");
    KillWolf.server.send("ENABLESPEAKAC");
    TimeTha spt = new TimeTha();
    spt.start();
    return;
}
KillWolf.server.send("MESSAGE【系统】进入轮流发言阶段，每人有 30 秒发言时间");
try {
    Thread.sleep(2000);
} catch (InterruptedException e) {
    e.printStackTrace();
}
DataUser user = null;//(DataUser[]) KillWolf.serverData.getUser(firstdeath);

if(firstdeath.equals("")) for(int j=0;j<KillWolf.serverData.UsersLen;j++){
    if(KillWolf.serverData.ServerUsers[j] != null)
        if(KillWolf.serverData.ServerUsers[j].live) firstdeath = KillWolf.serverData.ServerUsers[j].name;
}
for(int i = 0;i<KillWolf.serverData.UsersLen;i++){
    if(KillWolf.serverData.ServerUsers[i]==null) continue;

    if(firstdeath.equals(KillWolf.serverData.ServerUsers[i].name)){
        int j=i+1;
        for(;j<KillWolf.serverData.UsersLen;j++){
            if(KillWolf.serverData.ServerUsers[j] != null)
                if(KillWolf.serverData.ServerUsers[j].live) user = KillWolf.serverData.ServerUsers[j];
        }
        if(user == null) for(j=0;j<i;j++){
            if(KillWolf.serverData.ServerUsers[j] != null)
                if(KillWolf.serverData.ServerUsers[j].live) user = KillWolf.serverData.ServerUsers[j];
        }
        if(user == null) {
            KillWolf.serverData.GameMainThread.gameNext();
            System.out.println("ERROR !!!!!!!!!!!!!");
            return;
        }
        KillWolf.serverData.StartSpeakName = user.name;
        KillWolf.serverData.nowSpeakUser = user.name;
        KillWolf.server.send("MESSAGE【系统】轮到"+user.name+"发言");
```

```
                KillWolf.server.sendto("ENABLESPEAKAC",user.sk);

                SpeakThread spt = new SpeakThread(user.name,KillWolf.serverData.Day);
                spt.start();
                break;
            }
            //if(i==0)

        }
    }

    public void gameVote() {
        KillWolf.server.send("MESSAGE【系统】发言结束，现在开始进行投票");

        //投票弹窗
        String[] allUserName = (String[])KillWolf.serverData.getLivingUsers(true);
        DataUser[] allUser = (DataUser[]) KillWolf.serverData.getLivingUsers(false);
        //String[] wolfUserName = KillWolf.serverData.ArrayDatatoName(wolfUser);

        KillWolf.serverData.Selection[KillWolf.serverData.SelectionLen] = new SelectionEvent(Integer.toString(KillWolf.serverData.SelectionLen),allUserName,20,new VoteSelection());
        KillWolf.serverData.Selection[KillWolf.serverData.SelectionLen].openSeletion(KillWolf.server, allUser, allUser.length,"开始投票","投他","请选择",false);
        KillWolf.serverData.SelectionLen++;

    }
    public void gameHunt(String death) {

        DataUser user = KillWolf.serverData.getUser(death);
        if (user != null && user.identify.equals("猎人")) {
            KillWolf.server.sendto("HUNTERSTARTLAST", user.sk);
        }
    }
    public void gameNext(){
        if(!isWin()){
            (KillWolf.serverData.GameMainThread = new GameMainThread()).start();
        }
    }
    public boolean isWin(){
        String whoWin = KillWolf.serverData.whoWin();
        if(!(whoWin.equals("无"))){
            KillWolf.server.send("MESSAGE【系统】游戏结束，"+whoWin+"胜");

            gameEnd();
            return true;
            //KillWolf.server.send("MESSAGE【系统】将在 5 秒后关闭窗口");
        }
        return false;
    }
    public void gameEnd(){
        KillWolf.server.send("MESSAGE【系统】游戏统计");
        DataUser[] d = KillWolf.serverData.ServerUsers;
        for(int i=0;i<KillWolf.serverData.UsersLen;i++){
            if(d[i]!=null){
                KillWolf.server.send("MESSAGE【系统】"+d[i].name+" "+d[i].identify+" "+(d[i].live?"存活":"淘汰"));
            }
        }
        KillWolf.serverData.freeSpeak = true;
        KillWolf.server.send("ENABLESPEAKEND");

    }
}
// ServerDeal.java
package KillWolf.SocketDeal;

import java.net.Socket;
import java.security.PublicKey;

import KillWolf.Data.SelectionEventCallBack;
import KillWolf.KillWolf;
import KillWolf.Data.DataUser;
import KillWolf.Data.SelectionEvent;
import KillWolf.KillWolf.serverData;
import SocketServe.CallBack;
import com.sun.source.tree.ReturnTree;

public class ServerDeal implements CallBack {
    public void run(String text,Socket sk){
        System.out.println(sk.getPort()+" Server Receive: "+text+"####");
        if(text.startsWith("ENTER")){
            if(KillWolf.GameMain!=null) {
                KillWolf.server.sendto("ENTERFAIL 游戏已经开始，请稍后再试",sk);
                return;
            }
            String name = text.substring(5);
            for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
                if(KillWolf.serverData.ServerUsers[i]!=null)
                if(name.equals(KillWolf.serverData.ServerUsers[i].name)) {
                    //System.out.println("NAME: "+name+" "+i);
                    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);
                    return;
                }
            }
            KillWolf.server.sendto("ENTERSUCCESS["+(KillWolf.serverData.UsersLen+1)+"]"+name,sk);
            KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen] = new DataUser();
            KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen].name="["+(KillWolf.serverData.UsersLen+1)+"]"+name;
            KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen].sk=sk;
            KillWolf.serverData.ServerUsers[KillWolf.serverData.UsersLen].port=sk.getPort();
            KillWolf.serverData.UsersLen++;
            KillWolf.serverData.UsersRealLen++;
            String STR = "";
            for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
                if(KillWolf.serverData.ServerUsers[i]!=null)
                STR = STR +'\t'+ KillWolf.serverData.ServerUsers[i].name;
            }
            KillWolf.server.send("NEWWAITUSERLIST"+STR.substring(1));
            return;
        }
        else if(text.equals("GETUSERS")) {
            String STR = "";
            for(DataUser i : KillWolf.serverData.ServerUsers) {
                if(i!=null)
                STR = STR +'\t'+ i.name;
            }
            KillWolf.server.send(STR.substring(1));
            return;
        }else if(text.startsWith("User Disconnect ")) {
            int port = Integer.parseInt(text.substring(16));
            for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
                if(KillWolf.serverData.ServerUsers[i]!=null)
                if(port == KillWolf.serverData.ServerUsers[i].port) {
                    System.out.println("NAME QUIT: "+KillWolf.serverData.ServerUsers[i].name);
                    KillWolf.server.send("USERQUIT"+KillWolf.serverData.ServerUsers[i].name);//KillWolf.server.sendto("ENTERFAIL",sk);
                    SelectionEvent.UserQuit(KillWolf.serverData.ServerUsers[i].name);
                    KillWolf.serverData.ServerUsers[i] = null;
            KillWolf.serverData.UsersRealLen--;
            //添加选择时用户退出
            return;
        }
    }


}else if(text.startsWith("SELECT")) {
    KillWolf.server.send(text);
}else if(text.startsWith("FINALSELECT")) {
    String s[] = text.substring(11).split("\t");
    SelectionEvent se = KillWolf.serverData.Selection[Integer.parseInt(s[0])];
    se.update(s[1],s[2]);
}else if(text.startsWith("GAMEREADY")){
    serverData.getUser(text.substring(9)).live = true;
    for(int i=0;i<serverData.UsersLen;i++){
        if(serverData.ServerUsers[i]!=null)
        if(!serverData.ServerUsers[i].live) return;
    }
    KillWolf.server.send("MESSAGE【系统】玩家全部进入，游戏将于 3 秒后开始");
    try {
        Thread.sleep(3000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    KillWolf.serverData.Day = 0;
    KillWolf.serverData.night = false;

    (KillWolf.serverData.GameMainThread = new GameMainThread()).start();
}else if(text.startsWith("USERSPEAK")){
    String[] submitMsg = text.substring(9).split("\t");
    KillWolf.server.send("MESSAGE【玩家】"+submitMsg[0]+"  ：  "+submitMsg[1]);
    if(!KillWolf.serverData.freeSpeak) nextSpeak(submitMsg[0],serverData.Day);

}else if(text.startsWith("YUYANEND")){
    KillWolf.serverData.GameMainThread.YuYanCompleted = true;
    KillWolf.serverData.GameMainThread.gameDay();
}else if(text.startsWith("WIZARDANTI")) {
    String result = text.substring(10);
    if(result.equals("YES")) {
        DataUser u = KillWolf.serverData.getUser(KillWolf.serverData.deathUser);
        if(u!=null) u.live = true;
        KillWolf.server.sendto("MESSAGE【女巫】你为"+KillWolf.serverData.deathUser+"使用了解药。", sk);
```

```
                    KillWolf.serverData.deathUser = "**NONE**";
                    KillWolf.serverData.GameMainThread.WizardCompleted = true;
                    KillWolf.serverData.GameMainThread.gameDay();
                }else if(result.equals("NO")){
                    if(!KillWolf.serverData.deathUser.equals("**NONE**")) {
                        DataUser u = KillWolf.serverData.getUser(KillWolf.serverData.deathUser);
                        if(u!=null) u.live = false;
                        KillWolf.server.sendto("MESSAGE【女巫】昨夜"+KillWolf.serverData.deathUser+"死亡，你没有使用解药。", sk);
                    }
                    else KillWolf.server.sendto("MESSAGE【女巫】昨夜无人死亡。", sk);
                    KillWolf.server.sendto("WIZARDPOS", sk);
                }else if(result.equals("NONE")){
                    if(!KillWolf.serverData.deathUser.equals("**NONE**")){
                        DataUser u = KillWolf.serverData.getUser(KillWolf.serverData.deathUser);
                        if(u!=null) u.live = false;
                    }
                    KillWolf.server.sendto("WIZARDPOS", sk);
                }
            }else if(text.startsWith("WIZARDPOS")) {
                String result = text.substring(9);
                KillWolf.serverData.deathUser2 = "**NONE**";
                if(result.startsWith("YES")) {
                    DataUser u = KillWolf.serverData.getUser(result.substring(3));
                    if(u!=null) u.live = false;
                    KillWolf.serverData.deathUser2 = result.substring(3);
                }else if(result.equals("NO")){

                }else if(result.equals("NONE")){

                }
                KillWolf.serverData.GameMainThread.WizardCompleted = true;
                KillWolf.serverData.GameMainThread.gameDay();
            }
            else if(text.startsWith("HUNTSTART")){
                String result = text.substring(9);
                if(result.equals("YES")){
                    KillWolf.server.sendto("HUNTSELECT", sk);
                }else if(result.equals("NO")){
                    KillWolf.server.sendto("MESSAGE【猎人】你已死亡，没有使用技能。",sk);
                    KillWolf.serverData.GameMainThread.dayWork();
                }else if(result.startsWith("LAST")){
                    if(result.substring(4).equals("YES")){
                        KillWolf.server.sendto("HUNTSELECTLAST", sk);
```

```
        }else if(result.substring(4).equals("NO")) {
            KillWolf.server.sendto("MESSAGE【猎人】你已死亡，没有使用技能。", sk);
            serverData.GameMainThread.gameNext();
        }     //KillWolf.serverData.GameMainThread.dayWork();
    }
}else if(text.startsWith("HUNTSELE")){
    //System.out.println("猎人技能成功#########");
    String result = text.substring(8);
    if(result.startsWith("YES")) {
        String[] u = result.substring(3).split("\t");
        DataUser us = KillWolf.serverData.getUser(u[1]);
        if(us!=null) us.live = false;
        KillWolf.serverData.deathUser3 = u[1];
        System.out.println("猎人技能成功#########");
        KillWolf.server.send("MESSAGE【猎人】"+u[0]+"死前开枪杀死了"+serverData.deathUser3);
        KillWolf.server.send("USERDIED"+serverData.deathUser3);
        KillWolf.serverData.GameMainThread.dayWork();
    }
    else if(result.equals("NO")){
        KillWolf.serverData.GameMainThread.dayWork();
    }else if (result.startsWith("LAST")){
        result = result.substring(4);
        if(result.startsWith("YES")) {
            String[] u = result.substring(3).split("\t");
            DataUser us = KillWolf.serverData.getUser(u[1]);
            if(us!=null) us.live = false;
            //KillWolf.serverData.getUser(u[1]).live = false;
            KillWolf.serverData.deathUser3 = u[1];
            System.out.println("猎人技能成功#########");
            KillWolf.server.send("MESSAGE【猎人】"+u[0]+"死前开枪杀死了"+serverData.deathUser3);
            KillWolf.server.send("USERDIED"+serverData.deathUser3);
            //KillWolf.serverData.GameMainThread.dayWork();
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        else if(result.equals("NO")) {
        }
        serverData.GameMainThread.gameNext();
    }
```

```

        }

        //KillWolf.server.send("["+sk.getPort()+"]"+text);
    }
    public static boolean nextSpeak(String str,int Day){
        System.out.println("NOW SPEAK:"+KillWolf.serverData.nowSpeakUser+" CALLBACK:"+str);
        if(!KillWolf.serverData.nowSpeakUser.equals(str) || Day != serverData.Day) return false;
        else {
            DataUser[] liveUsers = (DataUser[]) KillWolf.serverData.getLivingUsers(false);
            String lastSpeak = str;
            if(serverData.StartSpeakName.equals("**VOTE**")) return true;
            for(int i = 0;i<liveUsers.length;i++){
                if(lastSpeak.equals(liveUsers[i].name)){
                    KillWolf.server.sendto("FORBIDSPEAK",liveUsers[i].sk);

                    if(i==liveUsers.length-1) i=-1;
                    i=i+1;

                    System.out.println("CHANGE  SPEAK  "+KillWolf.serverData.nowSpeakUser+" TO "+liveUsers[i].name);
                    if(liveUsers[i].name.equals(serverData.StartSpeakName)){
                        //一轮发言结束
                        serverData.StartSpeakName = "**VOTE**";
                        KillWolf.serverData.GameMainThread.gameVote();
                        return true;
                    }
                    KillWolf.serverData.nowSpeakUser = liveUsers[i].name;
                    KillWolf.server.send("MESSAGE【系统】轮到"+liveUsers[i].name+"发言");
                    KillWolf.server.sendto("ENABLESPEAKAC",liveUsers[i].sk);
                    SpeakThread spt = new SpeakThread(liveUsers[i].name, serverData.Day);
                    spt.start();
                    return true;
                }
                //if(i==0)

            }
            //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。
            KillWolf.server.send("MESSAGE【系统】有用户退出，发言阶段重新开始。");
            for(int j=0;j<KillWolf.serverData.UsersLen;j++){
                if(KillWolf.serverData.ServerUsers[j] != null)
                    if(KillWolf.serverData.ServerUsers[j].live) {
                        serverData.StartSpeakName = KillWolf.serverData.ServerUsers[j].name;
                        KillWolf.serverData.nowSpeakUser = KillWolf.serverData.ServerUsers[j].name;
                        KillWolf.server.send("MESSAGE 【系统】轮到"+KillWolf.serverData.ServerUsers[j].name+"发言");
                        KillWolf.server.sendto("ENABLESPEAKAC",KillWolf.serverData.ServerUsers[j].sk);
                        SpeakThread spt = new SpeakThread(KillWolf.serverData.ServerUsers[j].name, serverData.Day);
                        spt.start();
                        break;
                    }
            }
            return true;
        }
    }


}

class SpeakThread extends Thread{
    String user;
    int Day;
    public SpeakThread (String user,int Day){
        this.user = user;
        this.Day = Day;
    }
    @Override
    public void run() {
        try {
            Thread.sleep(30*1000);
        }catch (InterruptedException e){
            e.printStackTrace();
        }
        ServerDeal.nextSpeak(user,Day);
    }


}
class WolfSelect implements SelectionEventCallBack {
    public void run(String ReceiveID,int[] count,String users[],String target[],String[] openusers) {
        String name = "**NONE**",sameName = "**NONE**";
        int max = 0,samemax = 0;
        System.out.println("####狼人投票--------");
        String STR = "Night"+KillWolf.serverData.Day;
        for(int i=0;i<users.length;i++){
            if(max < count[i]){
                max = count[i];
                name = users[i];
            }else if(max == count[i]) {
                sameName = users[i];
                samemax = max;
            }
            System.out.println("#"+users[i]+" "+count[i]);
            if(count[i] == 0) continue;
            STR = STR +"\n"+ "("+count[i]+")"+users[i];
            boolean first = true;
            for(int j=0;j<target.length;j++) {
                if(target[j] != null)
                if(target[j].equals(users[i])) {
                    if(first) {
                        first = false;
                        STR = STR + " <- "+openusers[j];
                    }else STR = STR +"、"+openusers[j];
                }
            }
        }
        System.out.println(name + "--------"+ max);
//        KillWolf.serverData.getUser(name).live = false;
        String tip ="";
        if(name == "**NONE**") {
            name = "存在平票，无人出局";
            KillWolf.serverData.deathUser="**NONE**";
        }else if(samemax == max){
            name = "存在平票，无人出局";
            KillWolf.serverData.deathUser="**NONE**";
        }else {
            tip = "被狼人暗杀";
            KillWolf.serverData.deathUser=name;
            //KillWolf.serverData.getUser(KillWolf.serverData.deathUser).live = false;
        }
        DataUser[] wu = KillWolf.serverData.getUsersWithID("狼人");
        String[] s = STR.split("\n");
        for(int i=0;i<wu.length;i++) {
            for(int j=0;j<s.length;j++) {
                KillWolf.server.sendto("MESSAGE【狼人投票】"+s[j],wu[i].sk);
            }
            KillWolf.server.sendto("MESSAGE【最终结果】"+name+"("+max+")"+tip,wu[i].sk);
        }
        KillWolf.serverData.GameMainThread.gameWizard(KillWolf.serverData.deathUser);
    }
}
class VoteSelection implements SelectionEventCallBack {

    public void run(String ReceiveID,int[] count,String users[],String target[],String[] openusers) {
        String name = "**NONE**",sameName = "**NONE**";
        int max = 0,samemax = 0;
        System.out.println("####投票--------");
        String STR = "投票结束";
        for(int i=0;i<users.length;i++){
            if(max < count[i]){
                max = count[i];
                name = users[i];
            }else if(max == count[i]) {
                sameName = users[i];
                samemax = max;
            }
            System.out.println("#"+users[i]+" "+count[i]);
            if(count[i] == 0) continue;
            STR = STR +"\n"+ "("+count[i]+")"+users[i];
            boolean first = true;
            for(int j=0;j<target.length;j++) {
                if(target[j] != null)
                if(target[j].equals(users[i])) {
                    if(first) {
                        first = false;
                        STR = STR + " <- "+openusers[j];
                    }else STR = STR +"、"+openusers[j];
                }
            }
        }
        System.out.println(name + "--------"+ max);
//      KillWolf.serverData.getUser(name).live = false;
        String tip ="";
        if(name == "**NONE**") {
            name = "存在平票，无人出局";
            KillWolf.serverData.deathUser4="**NONE**";
        }else if(samemax == max){
            name = "存在平票，无人出局";
            KillWolf.serverData.deathUser4="**NONE**";
        }else {
            tip = "投票出局";
            KillWolf.serverData.deathUser4=name;
            DataUser u =KillWolf.serverData.getUser(KillWolf.serverData.deathUser4);
            if(u!=null) u.live = false;
        }

//        System.out.println("-------------------"+STR);
//        DataUser[] wu = KillWolf.serverData.getUsersWithID("狼人");
        //DataUser[] wu = new DataUser[users.length];
//        if(KillWolf.serverData.getLivingUsers(false) instanceof DataUser[])
//            wu = (DataUser[]) KillWolf.serverData.getLivingUsers(false);
        String id = "";
        //参与本场投票的人
        for (int i =0;i<users.length;i++){
            //wu[i]=KillWolf.serverData.getUser(users[i]);
            if(users[i].equals(name))
            {
                DataUser u = KillWolf.serverData.getUser(users[i]);
                if(u!=null)     id = u.identify;
            }
        }
        String[] s = STR.split("\n");
        //for(int i=0;i<wu.length;i++) {
            for(int j=0;j<s.length;j++) {
                KillWolf.server.send("MESSAGE【投票】"+s[j]);
                //KillWolf.server.sendto();
            }
            //if("".equals(id))
            //else KillWolf.server.sendto("MESSAGE【最终结果】"+name+"("+max+")"+tip+"，他是"+id,wu[i].sk);
        //}
        try {
            Thread.sleep(2*1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        KillWolf.server.send("MESSAGE【最终结果】"+name+"("+max+")"+tip);
        if(!"存在平票，无人出局".equals(name))KillWolf.server.send("USERDIED"+name);

        if(serverData.GameMainThread.isWin()) return;

        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        if(id.equals("猎人"))KillWolf.serverData.GameMainThread.gameHunt(name);
        else{
            serverData.GameMainThread.gameNext();
        }

    }
}
// GameMain.java
package KillWolf.Window;

import KillWolf.Data.DataUser;
import KillWolf.KillWolf;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class GameMain {
//  public static void main(String[] args) {
//      GameMain game = new GameMain();
//      game.reset();
//  }
    public boolean freeSpeak = false;
    private JFrame frame = new JFrame("开始游戏");
    private JPanel panel = new JPanel();
    //玩家列表
    private JLabel userListLabel = new JLabel("在场玩家");
    private JList<String> userList = null;
    //出局玩家列表
    private JLabel outUserListLabel = new JLabel("淘汰玩家");
    private JList<String> OutUserList = null;
    //
    private JLabel identity = null;
    //公屏
    private JLabel screenStr = new JLabel("公屏");
    private JTextArea screen = new JTextArea();
    private JScrollPane jsp = null;
//  //投票按钮
//  private JButton vote = new JButton("投票");
    //发言框
    private JLabel submitBoxStr = new JLabel("发言框");
    private JTextField  submitStr = new JTextField();
    //发言按钮
    private JButton submit = new JButton("发言");
    public String[] UserListData ;
    public String[] UserListTipData ;
    public String[] UserOutListData ;
    public DataUser userSelf=null;
    public void reset(){
        //本地玩家信息初始化

        for(DataUser user : KillWolf.Users){
            if(user != null) {
                if (user.identify.equals("女巫")) {
                    user.haveAntidote = true;
                    user.havePoison = true;
                }
                user.live = true;
            }
        }
        //设置窗体参数
        frame.setSize(800,600);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);
        frame.setResizable(false);
        frame.setVisible(false);
        frame.setLocationRelativeTo(null);
        //在场玩家列表设置
        userListLabel.setBounds(20,5,100,20);
        frame.add(userListLabel);
        String[] items1=new String[10];
        userList=new JList(items1);     //创建 JList
        userList.setBounds(10, 30, 200, 240);
        userList.setFixedCellHeight(20);
        frame.add(userList);
        //淘汰玩家列表设置
        outUserListLabel.setBounds(20,275,100,20);
        frame.add(outUserListLabel);
        String[] items2=new String[10];
        OutUserList=new JList(items2);     //创建 JList
        OutUserList.setBounds(10, 300, 200, 240);
        OutUserList.setFixedCellHeight(20);
        frame.add(OutUserList);


        //顶端字体设置
        DataUser[] users= KillWolf.Users;
        String[] usn = new String[KillWolf.UsersRealLen];
        String[] usna = new String[KillWolf.UsersRealLen];
        String[] usnt = new String[KillWolf.UsersRealLen];
        int n = 0;
        for (int i=0;i<KillWolf.UsersLen;i++) {
            DataUser user = users[i];
            if(user == null) continue;
            if(user.name.equals(KillWolf.UserName)){
                userSelf=user;
            }
            usn[n]= user.name;
            //System.out.println("name:" + user.name);
            usna[n]=usn[n];
            usnt[n] = "";
            n++;
        }
        n=0;
        if(userSelf.identify.equals("狼人")){
            for (int i=0;i<KillWolf.UsersLen;i++) {
                DataUser user = users[i];
                if(user == null) continue;
                if(user.identify.equals("狼人")) {
                    usnt[n] = "(狼人)";
                }
                usna[n]=usn[n]+" "+usnt[n];
                n++;
            }
        }
        userList.setListData(usna);
        UserListData = usn;
        UserListTipData = usnt;
        UserOutListData = new String[0];

        identity = new JLabel(userSelf.name +"，你的身份是：" +userSelf.identify); ;
        identity.setBounds(270,0,450,100);
        identity.setFont(new Font(Font.DIALOG, Font.BOLD, 20));
        frame.add(identity);

        //公屏设置
        panel.setBounds(270,100,450,380);
        panel.setLayout(null);
        jsp = new JScrollPane(screen);
        jsp.setBounds(0,0,450,380);

        screen.setBounds(0,0,450,380);
        screen.setFont(new Font(Font.DIALOG,Font.BOLD,16));
        screen.setEditable(false);
        screen.setLineWrap(true);          //激活自动换行功能
        screen.setWrapStyleWord(true);            // 激活断行不断字功能
        screenStr.setBounds(270,60,100,50);

        jsp.setViewportView(screen);
//        screen.setBackground(Color.black);
//        jsp.setBackground(Color.BLUE);
//        panel.setBackground(Color.green);
        //jsp.setOpaque(false);
        jsp.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
        panel.add(jsp);
        frame.add(panel);
        frame.add(screenStr);

        submitBoxStr.setBounds(270,470,100,50);
        submitStr.setBounds(270,510,300,30);
        submitStr.setFont(new Font(Font.DIALOG,Font.BOLD,15));
        //submitStr.setText("请发言");
        submitStr.setColumns(3);
        frame.add(submitBoxStr);
        frame.add(submitStr);

        submit.setBounds(600,510,120,30);
        submit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                submit();
            }
        });
        frame.add(submit);

        this.freeSpeak = KillWolf.freeSpeak;
        ableSubmit();

        frame.setVisible(true);
        KillWolf.client.send("GAMEREADY"+KillWolf.UserName);
        addToScreen("【系统】等待玩家...");
    }
    public void forbidSubmit(){
        if(freeSpeak)return;
        ClockRun = false;
        submit.setText("发言");
        this.submit.setEnabled(false);
    }
    public void ableSubmit(){
        //if(freeSpeak)return;
        this.submit.setEnabled(true);
    }
    public boolean ClockRun = false;
    class TimeTh extends Thread{
        int time;
        public TimeTh(int time) {
            this.time = time-1;
        }
        public void run() {
            try {
                submit.setText("发言("+time+"s)");
                Thread.sleep(1000);
                while(time>1 && ClockRun) {
                    time--;
                    submit.setText("发言("+time+"s)");
                    Thread.sleep(1000);

                }
                submit.setText("发言");
                forbidSubmit();
            }catch(Exception e) {
                e.printStackTrace();
            }
        }
    }
    public void startSpeakClock(){
        if(freeSpeak){
            TimeTh tt = new TimeTh(120);
            ClockRun = true;
            tt.start();
            return;
        }
        TimeTh tt = new TimeTh(30);
        ClockRun = true;
        tt.start();
    }
    public void submit(){
        System.out.println(submitStr.getText());
        if(submitStr != null && (submitStr.getText() != null && !"".equals(submitStr.getText()))){
            KillWolf.client.send("USERSPEAK"+userSelf.name+"\t"+submitStr.getText());
            submitStr.setText("");
            if(!KillWolf.freeSpeak){
                this.submit.setEnabled(false);
                ClockRun = false;
                submit.setText("发言");
            }

        }

    }
    public void addToScreen(String str){
        //添加问本到公屏
        screen.append(str+"\r\n");
        //显示最新内容
        screen.setCaretPosition(screen.getText().length());
    }
    public   void setDark(){
        Color c = new Color(-12828863);
        if(userList != null) {
            userList.setBackground(c);
            userList.setForeground(Color.white);
        }
        if(OutUserList != null) {
            OutUserList.setBackground(c);
            OutUserList.setForeground(Color.white);
        }
        if(jsp != null){
            jsp.setBackground(c);
            jsp.setForeground(Color.white);
        }
        if(screen != null){
            screen.setBackground(c);
            screen.setForeground(Color.white);
        }
        if(submitStr != null){
            submit.setBackground(c);
            submit.setForeground(Color.white);
        }
        if(userListLabel != null)userListLabel.setForeground(Color.white);
        if(outUserListLabel != null)outUserListLabel.setForeground(Color.white);
        if(screenStr != null)screenStr.setForeground(Color.white);
        if(submitBoxStr != null)submitBoxStr.setForeground(Color.white);
        if(identity != null)identity.setForeground(Color.white);
        if(frame != null) frame.getContentPane().setBackground(Color.black);
```

```

//          if(CenterLabel != null) CenterLabel.setForeground(Color.white);
    }
    public   void setWhite(){
        if(userList != null) {
            userList.setBackground(Color.white);
            userList.setForeground(Color.black);
        }
        if(OutUserList != null) {
            OutUserList.setBackground(Color.white);
            OutUserList.setForeground(Color.black);
        }
        if(screen != null){
            screen.setBackground(Color.white);
            screen.setForeground(Color.black);
        }
//          if(jsp != null){
//              jsp.setBackground(Color.white);
//              jsp.setForeground(Color.black);
//          }
        if(submitStr != null){
            submit.setBackground(Color.white);
            submit.setForeground(Color.black);
        }
        if(userListLabel != null)userListLabel.setForeground(Color.black);
        if(outUserListLabel != null)outUserListLabel.setForeground(Color.black);
        if(screenStr != null)screenStr.setForeground(Color.black);
        if(submitBoxStr != null)submitBoxStr.setForeground(Color.black);
        if(identity != null)identity.setForeground(Color.black);
        if(frame != null) frame.getContentPane().setBackground(new Color(-1118482));

//          if(CenterLabel != null) CenterLabel.setForeground(Color.white);
    }

    public JFrame getFrame() {
        return frame;
    }

    public void setFrame(JFrame frame) {
        this.frame = frame;
    }

    public JLabel getUserListLabel() {
        return userListLabel;
```

```
	}

	public void setUserListLabel(JLabel userListLabel) {
		this.userListLabel = userListLabel;
	}

	public JList<String> getUserList() {
		return userList;
	}

	public void setUserList(JList<String> userList) {
		this.userList = userList;
	}

	public JLabel getOutUserListLabel() {
		return outUserListLabel;
	}

	public void setOutUserListLabel(JLabel outUserListLabel) {
		this.outUserListLabel = outUserListLabel;
	}

	public JList<String> getOutUserList() {
		return OutUserList;
	}

	public void setOutUserList(JList<String> outUserList) {
		OutUserList = outUserList;
	}

	public JLabel getIdentity() {
		return identity;
	}

	public void setIdentity(JLabel identity) {
		this.identity = identity;
	}

	public JLabel getScreenStr() {
		return screenStr;
	}

	public void setScreenStr(JLabel screenStr) {
		this.screenStr = screenStr;
```

```
    }

    public JTextArea getScreen() {
        return screen;
    }

    public void setScreen(JTextArea screen) {
        this.screen = screen;
    }

    public JLabel getSubmitBoxStr() {
        return submitBoxStr;
    }

    public void setSubmitBoxStr(JLabel submitBoxStr) {
        this.submitBoxStr = submitBoxStr;
    }

    public JTextField getSubmitStr() {
        return submitStr;
    }

    public void setSubmitStr(JTextField submitStr) {
        this.submitStr = submitStr;
    }

    public JButton getSubmit() {
        return submit;
    }

    public void setSubmit(JButton submit) {
        this.submit = submit;
    }

    public String[] getUserListData() {
        return UserListData;
    }

    public void setUserListData(String[] userListData) {
        UserListData = userListData;
    }

    public String[] getUserOutListData() {
        return UserOutListData;
    }

    public void setUserOutListData(String[] userOutListData) {
        UserOutListData = userOutListData;
    }
}
// LoginWindow.java
package KillWolf.Window;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Random;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import KillWolf.KillWolf;
import KillWolf.SocketDeal.ClientDeal;
import KillWolf.SocketDeal.ServerDeal;
import SocketServe.sClient;
import SocketServe.sServer;

public class LoginWindow {
    public JButton ServerButton = null;
    public JButton LoginButton = null;
    public JTextField userName = null;
    public JTextField ServerIp = null;
    public JFrame window = null;
    public String MIP = null;
    public void reset(String ip) {
        window = new JFrame("狼人杀  V.0.1");
        window.setVisible(false);
        window.setSize(600, 200);//设置大小
        window.setLocationRelativeTo(null);//设置居中
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//设置可关闭
        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）
        window.setResizable(false);//设置窗口不可拉伸改变大小
        //设置用户名标签
        JLabel username_label =new JLabel("用户名");
        username_label.setBounds(100,30,100,20);
        window.add(username_label);
        //设置文本框
        userName = new JTextField();
        userName.setBounds(160, 30, 300, 20);
        Random r = new Random();
        //r.nextInt();
        r.setSeed(System.currentTimeMillis());
        userName.setText("user"+r.nextInt(10000));
        window.add(userName);
        //设置标签
        JLabel ServerIp_label =new JLabel("服务器地址");
        ServerIp_label.setBounds(80,80,100,20);
        window.add(ServerIp_label);
        //设置文本框
        ServerIp = new JTextField();
        ServerIp.setBounds(160, 80, 300, 20);
        ServerIp.setText(ip);
        this.MIP = ip;
        window.add(ServerIp);
        //JPasswordField pwd=new JPasswordField();//隐藏密码
        //pwd.setBounds(150, 200, 300, 50);
        //window.add(pwd);
        //设置按钮
        LoginButton = new JButton("连接服务器");
        //Login.setEnabled(false);
        LoginButton.setBounds(180, 120, 100, 20);
        window.add(LoginButton);
        ServerButton = new JButton("作为服务器");
        //Login.setEnabled(false);
        ServerButton.setBounds(320, 120, 100, 20);
        window.add(ServerButton);
        //设置作为主机动作
        ButtonAction ServerAction = new ButtonAction();
        ServerButton.addActionListener(ServerAction);
        //设置连接服务器动作
        LinkAction LinkAction = new LinkAction();
        LoginButton.addActionListener(LinkAction);
        window.setVisible(true);//设置面板可见
    }
    class ButtonAction implements ActionListener{
        public void actionPerformed(ActionEvent event){
            if(!setName()) return;
            ServerIp.setText(MIP);
            KillWolf.server = new sServer();
            ServerDeal cb = new ServerDeal();
        if(!KillWolf.server.start(cb)){
            JOptionPane.showMessageDialog(null, "端口被占用，服务器启动失败！ ");
            return;
        }
        if(!createClient(MIP)){
            JOptionPane.showMessageDialog(null, "客户端启动失败！ ");
            return;
        }
        //window.setTitle(ServerIp.getText());
        KillWolf.ServerIp = MIP;
        KillWolf.isServer = true;

        ServerIp.setEnabled(false);
        ServerButton.setEnabled(false);
        KillWolf.client.send("ENTER"+KillWolf.UserName);

        //window.setTitle(ServerIp.getText());
    }
}
class LinkAction implements ActionListener{
    public void actionPerformed(ActionEvent event){
        if(!setName()) return;
        if(KillWolf.ServerIp == null) {
            if(!createClient(ServerIp.getText())){
                JOptionPane.showMessageDialog(null, "客户端启动失败!可能是服务器未连接成功。");
                return;
            }
            KillWolf.ServerIp = ServerIp.getText();
            KillWolf.isServer = false;
        }

        ServerIp.setEnabled(false);
        ServerButton.setEnabled(false);
        KillWolf.client.send("ENTER"+KillWolf.UserName);
    }
}
public boolean createClient(String IP) {
    KillWolf.client = new sClient();
    ClientDeal cb = new ClientDeal();
    //System.out.println("Connecting Server on "+IP+"...");
    return KillWolf.client.start(IP,cb);
}
public boolean setName() {
        if( userName.getText().equals("")) {
            JOptionPane.showMessageDialog(null, "请输入用户名");
            return false;
        }
        KillWolf.UserName = userName.getText();
        return true;

    }
}
// SelectWindow.java
package KillWolf.Window;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JRootPane;
import javax.swing.ListSelectionModel;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import KillWolf.KillWolf;
import KillWolf.Data.SelectCallBack;


/*
    构造类：String name,boolean ReceiveFromServer,String ReceiveID,boolean DisableParent,JFrame ParentWindow;
    方法：设置标题（可用于显示剩余时间） setTitle(title)
        启动 start(String tip,String ButtonTitle,String[] users, boolean canClose,SelectCallBack cb)
        关闭 getSelectionAndClose()
    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

    for example:
        SelectWindow = new SelectWindow("选择",true,"123",true,loginwindow.window);
        String[] b = {"a","b","c",UserName};
        SelectWindow.start("请选择", "投票", b, true,cb);

        cb 另需构造
```

```
*/


public class SelectWindow {
    public boolean isRunCB = false;
    public String ReceiveID;
    public boolean ReceiveFromServer;
    public boolean DisableParent;
    public JFrame ParentWindow;
    public JFrame window;
    public JLabel CenterLabel;
    public JList<String> List;
    public String name;
    public JButton ConfirmButton;
    public JButton CloseButton;
    public String[] originUsers;
    public String[] usersTip;
    public String[] targetUsers;
    public int[] selectCount;
    public String LastSelect = "**NONE**";
    public boolean isSendFinal = false;
    public SelectWindow(String name,boolean ReceiveFromServer,String ReceiveID,boolean DisableParent,JFrame ParentWindow) {
        this.ReceiveFromServer = ReceiveFromServer;
        this.ReceiveID = ReceiveID;
        this.DisableParent = DisableParent;
        this.ParentWindow = ParentWindow;
        this.name = name;
    }
    public boolean DarkMode = false;
    public SelectCallBack cb;
    public void setColorMode(boolean dark) {
        this.DarkMode = dark;
        if(dark) {
            if(List != null) {
                List.setBackground(Color.black);
                List.setForeground(Color.white);
            }
            if(window != null) window.getContentPane().setBackground(Color.black);
            if(CenterLabel != null) CenterLabel.setForeground(Color.white);
            //if()
        }
    }
    public void start(String tip,String ButtonTitle,String[] users, boolean canClose,SelectCallBack cb, int
TimeLimit, String[] usertip) {
        this.cb = cb;
        if(usertip==null) {
            usertip = new String[users.length];
        }
        this.usersTip = usertip;
        window = new JFrame(name);
        window.setVisible(false);
        window.setSize(300, 400);//设置大小
        window.setLocationRelativeTo(null);//设置居中
        window.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);//设置可关闭
        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）
        window.setResizable(false);//设置窗口不可拉伸改变大小
        window.setAlwaysOnTop(true);
        window.setUndecorated(true);                                    //不显示标题栏,最大化,最小化,退出按钮
        window.getRootPane().setWindowDecorationStyle(JRootPane.WARNING_DIALOG);//    使frame 只剩下标题栏
        CenterLabel =new JLabel(tip);
        CenterLabel.setBounds(10,10,260,20);
        window.add(CenterLabel);
        users = users.clone();
        originUsers = new String[users.length];
        targetUsers = new String[users.length];
        for(int i=0;i<users.length;i++) {
            originUsers[i] = users[i];
            targetUsers[i] = "";
            if(usersTip[i]==null) usersTip[i]="";
            if(ReceiveFromServer) users[i] = "(0)"+users[i]+" "+usersTip[i];
            else users[i] = users[i]+" "+usersTip[i];

        }
        selectCount = new int[users.length];
        List = new JList(users);        //创建 JList
        List.setBounds(10, 40, 260, 260);
        List.setFixedCellHeight(20);
        List.addListSelectionListener(new ListSelectionHandler());
        List.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        window.add(List);

        ConfirmButton = new JButton(ButtonTitle);
        ConfirmButton.setEnabled(false);
        ConfirmButton.setBounds(30, 310, 100, 40);
        ConfirmButton.addActionListener(new ButtonAction("confirm"));
        window.add(ConfirmButton);


        CloseButton = new JButton("弃权");
        if(canClose) {
            CloseButton.setEnabled(true);
        }else {
            CloseButton.setEnabled(false);
        }
        CloseButton.setBounds(160, 310, 100, 40);
        CloseButton.addActionListener(new ButtonAction("cancel"));
        window.add(CloseButton);


        if(this.DisableParent && this.ParentWindow != null) this.ParentWindow.setEnabled(false);

        TimeTh at = new TimeTh(TimeLimit);
        at.start();

        setColorMode(this.DarkMode);

        window.setVisible(true);
    }
    class TimeTh extends Thread{
        int time;
        public TimeTh(int time) {
            this.time = time-1;
        }
        public void run() {
            try {
                window.setTitle(time+"s "+name);
                while(time>0) {
                    Thread.sleep(1000);
                    time--;
                    window.setTitle(time+"s "+name);
                }
                window.setTitle(name);
                ConfirmButton.setEnabled(false);
                CloseButton.setEnabled(false);
            }catch(Exception e) {
                e.printStackTrace();
            }
        }
    }
```

```
public void setTitle(String Title) {
    window.setTitle(name + Title);
}
public void SELECTInfo(String ServerText) {
    if(!ReceiveFromServer) return;
    String[] c = ServerText.split("\t");
    // c[0] ReceiveID ,c[1] 发出者 ,c[2] 目标者, c[3] 原始目标（被取消）
    String[] nu = new String[this.originUsers.length];

    if(c[0].equals(this.ReceiveID)) {

        for(int i=0;i<this.originUsers.length;i++) {
            //nu[i] = this.originUsers[i];

            if(this.originUsers[i].equals(c[1])) {
                if(c[2].equals("**NONE**")) targetUsers[i] = " 弃权";
                else targetUsers[i] = " -> " + c[2];
            }
            if(this.originUsers[i].equals(c[2])) {
                selectCount[i]++;
                System.out.println(this.originUsers[i]+" +1 => "+ selectCount[i]);
            }
            if(this.originUsers[i].equals(c[3])) {
                selectCount[i]--;
                System.out.println(this.originUsers[i]+" -1 => "+ selectCount[i]);
            }
            System.out.println(this.originUsers[i]+" "+c[2]+" "+ selectCount[i]);
            //System.out.println("LIST0: "+nu[i]);
            //System.out.println("LISTc: "+selectCount[i]);
        }
        for(int i=0;i<nu.length;i++) {
            nu[i] = "(" +selectCount[i] + ")" +originUsers[i]+" "+usersTip[i]+ targetUsers[i];
            //System.out.println("LIST: "+nu[i]);
        }
        List.setListData(nu);
    }

}
class ListSelectionHandler implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent e) {
        if(e.getValueIsAdjusting()) {
            String t;
            if(List.getSelectedIndex() == -1) {
                //t = "**NONE**";
                return;
            }
        else t = originUsers[List.getSelectedIndex()];
        System.out.printf("Select: %s\n",t);
        ConfirmButton.setEnabled(true);

        String a = LastSelect;
        if(List.getSelectedIndex() == -1)
            LastSelect = "**NONE**";
        else LastSelect = originUsers[List.getSelectedIndex()];

        if(ReceiveFromServer) KillWolf.client.send("SELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+t+"\t"+a);
    }

    //output.append("LeadSelectionIndex is " + lsm.getLeadSelectionIndex() + "\n");


  }
}
class ButtonAction implements ActionListener{//开始游戏
    String ID;
    public ButtonAction(String ID) {
        this.ID = ID;
    }
    public void actionPerformed(ActionEvent event){
        isSendFinal = true;
        List.setEnabled(false);
        CloseButton.setEnabled(false);
        ConfirmButton.setEnabled(false);
        isRunCB = true;
        if(ID.equals("confirm")) {
            //List.setEnabled(false);
            if(ReceiveFromServer) {
                KillWolf.client.send("FINALSELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+LastSelect);
            }
            if(cb != null) cb.run(LastSelect);
        }else {
            //取消
            if(ReceiveFromServer) {
                KillWolf.client.send("SELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+"**NONE**"+"\t"+LastSelect);
```

```
                    KillWolf.client.send("FINALSELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+"**NONE**");
                }
                if(cb != null) cb.run("**NONE**");
            }



        }
    }
    public String getSelectionAndClose(String ReceiveID) {
        if(!ReceiveID.equals(this.ReceiveID)) return "";
        boolean a = isRunCB;
        isRunCB = true;
        window.setVisible(false);
        window.setEnabled(false);
        if(!a && cb != null) {
            cb.run(LastSelect);
        }
        if(!isSendFinal && ReceiveFromServer){
            isSendFinal = true;
            KillWolf.client.send("FINALSELECT"+ReceiveID+"\t"+KillWolf.UserName+"\t"+LastSelect);
        }
        if(DisableParent && ParentWindow != null) ParentWindow.setEnabled(true);

        return LastSelect;
    }
}
// WaitRoom.java
package KillWolf.Window;

import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.event.*;
import javax.swing.text.BadLocationException;
import javax.swing.text.Document;

import KillWolf.KillWolf;
import KillWolf.Data.DataUser;
import KillWolf.SocketDeal.ClientDeal;

import java.awt.*;
import java.awt.event.*;
import java.util.Random;

import SocketServe.*;

public class WaitRoom {
    public JButton LoginButton = null;
    public JTextField peoCount = null;
    public JTextField wolfCount = null;
    public JCheckBox wizardOption;
    public JCheckBox hunterOption;
    public JCheckBox yuyanjiaOption;
    public JCheckBox winmodeOption;
    public JCheckBox speakOption;
    //public JCheckBox optionD;
    public JList UserList;
    public JFrame window = null;
    public JLabel Userlenlabel;
    public JLabel Charlenlabel;
    public String MIP = null;
    public int CharLen = 0;
    public int peoLen = 1;
    public int wolfLen = 1;
    public int wizardLen = 0;
    public int hunterLen = 0;
    public int yuyanjiaLen = 0;
    public void reset(String ip,String name,boolean isServer) {
        KillWolf.serverData.freeSpeak = false;
        KillWolf.serverData.playMode = false;

        window = new JFrame("狼人杀  "+name+" "+ip);
        window.setVisible(false);
        window.setSize(400, 420);//设置大小
        window.setLocationRelativeTo(null);//设置居中
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);//设置可关闭
        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）
        window.setResizable(false);//设置窗口不可拉伸改变大小
        //设置用户名标签
        JLabel username_label =new JLabel("用户名："+name);
        username_label.setBounds(10,10,200,20);
        window.add(username_label);
        //设置标签
        JLabel ServerIp_label =new JLabel("服务器地址："+KillWolf.ServerIp);
        ServerIp_label.setBounds(10,40,200,20);
        window.add(ServerIp_label);
        //设置标签
        Userlenlabel =new JLabel("在线玩家数："+KillWolf.UsersRealLen);
        Userlenlabel.setBounds(10,150,200,20);
        window.add(Userlenlabel);
        JLabel label2 =new JLabel("等待开始...");
        label2.setBounds(10,70,200,20);
        window.add(label2);
        this.MIP = ip;
        //设置按钮

        if(isServer) {
            LoginButton = new JButton("开始游戏");
            //LoginButton.setEnabled(false);
        }else {
            LoginButton = new JButton("等待房主开始");
            LoginButton.setEnabled(false);
        }
        LoginButton.setBounds(10, 120, 100, 20);
        window.add(LoginButton);

        LinkAction LinkAction = new LinkAction();
        LoginButton.addActionListener(LinkAction);


        String[] items=new String[10];
        UserList=new JList(items);      //创建 JList
        UserList.setBounds(200, 10, 150, 300);
        UserList.setFixedCellHeight(20);
        window.add(UserList);

        if(KillWolf.isServer) {
            //设置文本框
            Charlenlabel =new JLabel("已选角色数："+CharLen);
            Charlenlabel.setBounds(10,170,200,20);
            window.add(Charlenlabel);

            JLabel Peolenlabel =new JLabel("平民");
            Peolenlabel.setBounds(10,200,40,20);
            window.add(Peolenlabel);
            peoCount = new JTextField();
            peoCount.setBounds(50,200,40,20);
            peoCount.setText(Integer.toString(peoLen));
              window.add(peoCount);

```

```
      JLabel Wolflenlabel =new JLabel("狼人");
      Wolflenlabel.setBounds(10,220,40,20);
    window.add(Wolflenlabel);
    wolfCount = new JTextField();
    wolfCount.setBounds(50,220,40,20);
    wolfCount.setText(Integer.toString(wolfLen));
      window.add(wolfCount);


      Document dt = wolfCount.getDocument();
    dt.addDocumentListener(new TextListener("wolf"));
    wolfCount.addKeyListener(new CountKeyListener());


    Document dt2 = peoCount.getDocument();
    dt2.addDocumentListener(new TextListener("people"));
    peoCount.addKeyListener(new CountKeyListener());

      //TextOnValueChanged WAL = new TextOnValueChanged();
      //wolfCount.addActionListener(WAL);
      //peoCount.addActionListener(WAL);

      wizardOption = new JCheckBox("女巫");
      wizardOption.setBounds(10,240,80,20);
    hunterOption = new JCheckBox("猎人");
    hunterOption.setBounds(10,260,80,20);
    yuyanjiaOption = new JCheckBox("预言家");
    yuyanjiaOption.setBounds(10,280,100,20);
    //optionD = new JCheckBox("D.");


    wizardOption.addItemListener(new CheckValueChanged("wizard"));
    hunterOption.addItemListener(new CheckValueChanged("hunter"));
    yuyanjiaOption.addItemListener(new CheckValueChanged("yuyanjia"));

    window.add(wizardOption);
    window.add(hunterOption);
    window.add(yuyanjiaOption);


    winmodeOption = new JCheckBox("屠城玩法（狼人需要全部淘汰民和神）");
      winmodeOption.setBounds(10,320,300,20);
    speakOption = new JCheckBox("允许随时发言");
    speakOption.setBounds(10,340,100,20);
```

```

            winmodeOption.addItemListener(new CheckValueChanged("获胜玩法"));
            speakOption.addItemListener(new CheckValueChanged("发言"));

            window.add(winmodeOption);
            window.add(speakOption);

            updateCharCount(false);
        }
        window.setVisible(true);//设置面板可见


        //run(new ListTest(),250,375);

    }
    public void setUserCount(String users[]) {
        Userlenlabel.setText("在线玩家数："+KillWolf.UsersRealLen);
        UserList.setListData(users);
        UserList.setFixedCellHeight(20);

        updateCharCount(false);
        //window.setTitle(Title);
    }
    class LinkAction implements ActionListener{//开始游戏
        public void actionPerformed(ActionEvent event){
            if(updateCharCount(true)) {
                LoginButton.setEnabled(false);
                Random r = new Random();
                //r.nextInt();
                r.setSeed(System.currentTimeMillis());
                int n;
                if(wizardOption.isSelected()) {
                    do{n = r.nextInt(KillWolf.serverData.UsersLen);}
                    while(KillWolf.serverData.ServerUsers[n] == null || KillWolf.serverData.ServerUsers[n].identify != null);
                    KillWolf.serverData.ServerUsers[n].identify = "女巫";
                }
                if(hunterOption.isSelected()) {
                    do{n = r.nextInt(KillWolf.serverData.UsersLen);}
                    while(KillWolf.serverData.ServerUsers[n] == null || KillWolf.serverData.ServerUsers[n].identify != null);
                    KillWolf.serverData.ServerUsers[n].identify = "猎人";
                }
                if(yuyanjiaOption.isSelected()) {
                    do{n = r.nextInt(KillWolf.serverData.UsersLen);}
                    while(KillWolf.serverData.ServerUsers[n] == null || KillWolf.serverData.ServerUsers[n].identify != null);
                    KillWolf.serverData.ServerUsers[n].identify = "预言家";
                }
                for(int i=0;i<wolfLen;i++){
                    do{n = r.nextInt(KillWolf.serverData.UsersLen);}
                    while(KillWolf.serverData.ServerUsers[n] == null || KillWolf.serverData.ServerUsers[n].identify != null);
                    KillWolf.serverData.ServerUsers[n].identify = "狼人";
                }

                for(int i=0;i<KillWolf.serverData.UsersLen;i++){
                    if(KillWolf.serverData.ServerUsers[i] != null && KillWolf.serverData.ServerUsers[i].identify == null) {
                        KillWolf.serverData.ServerUsers[i].identify = "平民";
                    }
                }


                String STR = "";
                for(int i=0;i<KillWolf.serverData.UsersLen;i++) {
                    DataUser p = KillWolf.serverData.ServerUsers[i];
                    if(p!=null)STR = STR + "#u#" + p.name+"\t"+p.identify;
                }
                System.out.print(STR);
                KillWolf.server.send("GAMESTART"+(KillWolf.serverData.freeSpeak?"TRUE":"FALSE")+STR);
            }
        }
    }

    class CheckValueChanged implements ItemListener{
        String ID;
        public CheckValueChanged(String ID) {
            this.ID = ID;
        }

        public void itemStateChanged(ItemEvent e) {
            JCheckBox jcb = (JCheckBox) e.getItem();// 将得到的事件强制转化为 JCheckBox 类
            if (jcb.isSelected()) {// 推断是否被选择
                if(this.ID.equals("wizard")) wizardLen = 1;
                else if(this.ID.equals("hunter")) hunterLen = 1;
                else if(this.ID.equals("yuyanjia")) yuyanjiaLen = 1;
            else if(this.ID.equals("发言")) KillWolf.serverData.freeSpeak = true;
            else if(this.ID.equals("获胜玩法")) KillWolf.serverData.playMode = true;
        } else {
            if(this.ID.equals("wizard")) wizardLen = 0;
            else if(this.ID.equals("hunter")) hunterLen = 0;
            else if(this.ID.equals("yuyanjia")) yuyanjiaLen = 0;
            else if(this.ID.equals("发言")) KillWolf.serverData.freeSpeak = false;
            else if(this.ID.equals("获胜玩法")) KillWolf.serverData.playMode = false;
        }
        updateCharCount(false);
    }
}
class TextListener implements DocumentListener{
    String ID;
    public TextListener(String ID) {
        this.ID = ID;
    }
    public void insertUpdate(DocumentEvent e) {
        //System.out.println("insertUpdate" + e.toString());
        changedUpdate(e);
    }
    public void removeUpdate(DocumentEvent e) {
        //System.out.println("removeUpdate"+e.toString());
        changedUpdate(e);
    }

    public void changedUpdate(DocumentEvent e) {

        try {
            int num = Integer.parseInt(e.getDocument().getText(0, e.getDocument().getLength()));
            if(ID.equals("people")) {
                peoLen = Math.abs(num);
            }else {
                wolfLen = Math.abs(num);
            }
            updateCharCount(false);

        }catch (BadLocationException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }catch(Exception e21) {
            updateCharCount(false);
        }


        }
}
public class CountKeyListener implements KeyListener {
        @Override
        public void keyTyped(KeyEvent e) {
        // TODO Auto-generated method stub
        int keyChar=e.getKeyChar();
        if (keyChar>=KeyEvent.VK_0 && keyChar<=KeyEvent.VK_9) {
        } else {
        e.consume();
        }
        }
        @Override
        public void keyPressed(KeyEvent e) {
        // TODO Auto-generated method stub
        }
        @Override
        public void keyReleased(KeyEvent e) {
        // TODO Auto-generated method stub
        }
        }
boolean updateCharCount(boolean isFinal) {
        CharLen = wolfLen+peoLen+wizardLen+hunterLen+yuyanjiaLen;
        if(Charlenlabel!=null) Charlenlabel.setText("已选角色数："+CharLen);
        if(!KillWolf.isServer) return true;
        if(wolfLen<1 && isFinal) {
                JOptionPane.showMessageDialog(null, "狼人角色数量至少为 1");
                return false;
        }

        if(CharLen == KillWolf.serverData.UsersRealLen) {

                LoginButton.setEnabled(true);
                LoginButton.setText("开始游戏");
                return true;
        }else {
                LoginButton.setEnabled(false);
                LoginButton.setText("角色不匹配");
                return false;
        }
}
//定义新类，实现 Exit 按钮的时间监听
/*class btnAction implements ActionListener
```

```
    { //接收事件
        public void actionPerformed(ActionEvent event)
        {
            Object object = event.getSource();
            if (object == JButton1)
                JButton1_actionPerformed(event);
        }
    } */
}
```

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